2013

Intercepting a Target with Sensor Swarms

Meyer-Nieberg, Silja

2013 46th Hawaii International Conference on System Sciences
http://hdl.handle.net/10945/43690

2013 46th Hawaii International Conference on System Sciences

Intercepting a Target with Sensor Swarms

Silja Meyer-Nieberg, Erik Kropat, Stefan Pickl
Department of Computer Science
Universität der Bundeswehr München
Werner-Heisenberg-Weg 39, 85577 Neubiberg, Germany
Email: name.surname@unibw.de

Alex Bordetsky
Department of Information Science
Naval Postgraduate School Monterey
CA 93943, U.S.A.
Email: abordets@nps.edu

Abstract

This paper introduces a new coordination method to intercept a mobile target in urban areas with a team of sensor platforms. The task is to intercept the target before it leaves the area. The approach combines algorithmic concepts from ant colony and particle swarm optimization in order to bias the search and to spread the team in the search area. The algorithms introduced are tested in simulation experiments on grids. The success probabilities measured are relatively high for most parameter combinations, and the target is intercepted in roughly half the simulation time on average. Furthermore, the experiments reveal robust behavior with regard to the parameter setting.

Keywords

swarms, pursuit-evasion, search coordination

1. Introduction

In recent years, wireless sensor networks (WSNs) have gained more and more importance – for instance in disaster relief missions, maritime interdiction operations, and in civil-military co-operations. The focus of the present paper lies on the use of a WSN, or a sensor swarm, in an urban environment for locating and intercepting a mobile target that is being tracked. The task is challenging since the situation is dynamic and uncertain: Measurements are imprecise leading to uncertainty in the present target position and possibly also the sensor positions are not known exactly. While these measurement errors may not be large, they may cause the assumption that the target is presently in one street while it is actually in another. Therefore the swarm of (autonomous) sensors must search several streets. In addition, there is always the possibility that one or more sensors malfunction delivering strongly erroneous signals or that information may be lost due to faulty communication. Furthermore, time is usually a very critical component.

This paper addresses the task of coordinating the search focusing on one important submechanism of a mobile sensor network. Other important tasks, e.g. data fusion or sensor measurement planning are not subject of the present paper. It treats the sensors as abstract mobile agents which are able to take measurements which are then combined for a common picture of a part of the area. To coordinate the movement of the sensors, the paper introduces a new stochastic approach inspired by ant colony optimization (ACO) [6] and particle swarm optimization (PSO) [7]. It is structured as follows: First, the scenario considered is introduced. Afterwards, a brief overview over pursuit-evasion games is provided. This is followed by a presentation of the approaches and a first experimental analysis.

2. A Tracking and Intercepting Scenario

The scenario is based on a occurrence during the CENETIX simulation experiments in 2011 which were conducted by an international research group lead by A. Bordetsky from the Naval Postgraduate School, Monterey, USA. A team tracked the vehicle of a suspected group of terrorists into the outskirts of a city and tried to intercept the target in the urban environment. However, due to an effect later termed "cyber distortion" [9], the position of the target was too imprecise for its being successfully located. This is the baseline for the following scenario:

A number of ground vehicles (the sensor platforms) are tracking a target that may or may not be aware of being tracked. The target has entered an urban environment and shall be intercepted by the tracking team while in the area. However, due to measurement uncertainties, the position of the target cannot be determined with the required accuracy so that the target may be in several streets with a positive probability. This leads to the question which strategies shall be applied by the sensor platforms in order to locate the target. The task is complicated by the movement of the target: The probabilities for its being on certain streets change over time when new measurements are made. It is therefore in general not sufficient to deploy the team or swarm members to certain locations in the urban environment. Instead, an active search is required covering the interesting part of the region without loosing the network connectivity.

3. Search and Pursuit-Evasion Games

The scenario above is best described as a search and pursuit-evasion game. Also there are some similarities to the area of terrain covering although the latter usually assumes stationary targets that must be located by one or more sensors. Search and pursuit-evasion games in general and on graphs have a long research tradition (see [4, 1] for surveys) dating back

1530-1605/12 $26.00 © 2012 IEEE
DOI 10.1109/HICSS.2013.281

to the 1970s, e.g. [12]. Aside from security related applications, pursuit-evasion games are also important for multi-robot systems in search-and-rescue missions. Finding an adversarial target represents a worst case for search-and-rescue and gives thus an estimate for the worst case performance of the systems [4].

Several subtypes exist which differ in the assumptions they make, e.g. the sensor model used (perfect/imperfect information), information about the environment (accurate map/no map beforehand), or how the environment is modeled. It should be noted that the situation leads in general to $\mathcal{NP}$-hard optimization problems. Often, it is assumed that the target is aware of being tracked and evades the pursuers [14], whereas other approaches follow a nearly opposite strategy and assume a random behavior of the target. Similarly, see [4], two main focuses can be identified. Either strategies are developed that maximize a worst-case performance often combined with an omniscient evader with infinite speed or probabilistic methods are applied which optimize a statistical performance indicator, e.g. expected minimal time to detection or the probability to detect the target. Various approaches have been introduced ranging from stochastic heuristics [8] to specialized branch-and-bound optimization [13]. In the following, a brief overview is provided to illustrate the broad range of subjects and methods.

A special case concerning a graph representation is edge searching in which a omniscient and arbitrarily fast target shall be detected by a party of searchers [10]. Kolling et al. [10, 11] developed several algorithms based on a graph representation introducing the graph-clear problem. This problem differs from the usual edge search in the procedure that is required to clear an edge or a vertex. In a graph-clear problem a vertex is considered cleared if it is blocked from all edges and searched by a third robot. The task is to create a schedule and to coordinate the search such that the graph is cleared by the least number of robots. In [11], it was shown that the graph-clear problem is $\mathcal{NP}$-hard in general.

Gerkey et al. [8] applied a stochastic parallel hill-climbing approach called Parish to a general robot coordination problem. They used heuristics to estimate the benefits and costs of action plans that may involve more than one robot. Since the heuristics give only an estimate of the "true" value, they coupled it with a stochastic plan selection to overcome local optima.

Vidal et al. [14] investigated a probabilistic pursuit-evasion game with heterogeneous multi-robot teams and a randomly moving evader. Instead of assuming that the environment is completely known, the robot team builds a map while searching for the evader. The authors considered two greedy strategies: a local-max policy where each pursuer moves towards the cell with the highest probability of containing the evader in a local neighborhood and a global-max policy which uses the entire map. For the global-max policy, the cells with the globally highest probability are determined as well as the desired positions of the pursuers. This information is then used to derive the navigation policy. The strategies were tested in computer and real-life simulations with the global-max leading to better results in the majority of experiments.

This paper represents a simple and fast approach to intercept a target for which uncertain position estimates exist. The situation differs from the scenarios in most literature in that the target must be located while it traverses a certain area. Similar to Vidal et al. [14] we consider local information, i.e., local in time and environment. However, we use a stochastic approach which counteracts to some extend the common problems of that approach. It should be noted that our approach does not explicitly plan ahead in contrast to [8]. Instead we bias the search by incorporating global information concerning the current most promising search region.

4. Sensor Platforms: Coordinating the Search

In a first approach, we use the following model: Let $\hat{\mathbf{x}}(t) \in \mathbb{R}^2$ be the estimate of the target position. In our scenario, we assume that the sensor error follows a multivariate normal distribution. The error defines an area of interest $\mathcal{E}(t)$ at time t in the $\mathbb{R}^2$-plane which includes all streets, crossroads and alleys where the probability of the target's current location exceeds a threshold value $\epsilon_C > 0$. Let $\mathcal{U} = (V_U, E_U)$ be an undirected connected graph representing the street map of the urban area. Each edge is associated with a weight (i.e., the length of the edge) $l : E_U \to \mathbb{R}_+$. The edges denote the streets, whereas the nodes or vertices stand for crossroads. At time t we denote with $p(e, t)$ the probability that the target is on the edge e. A sensor intercepts the target if both are on the same edge (including the nodes) at the same time. In other words, a sensor can observe the whole edge. If this is not the case, we introduce an artificial node and split the edge. We also assume that all sensors are omnidirectional.

A team of K sensor platforms (called swarm in the following) searches for the single mobile target z. The swarm and the target are assumed to travel with constant speed with the sensors' speed v_s greater than or equal to the target's speed v_z. A sensor platform therefore needs l/v_s time units to traverse an edge of length l. Nodes can be passed in negligible time.

In our model, we assume that the probability model for the target position is updated in discrete time steps by one or more sensors after new measurements arrive. In our first approach, we start with an unbounded communication radius leading to a common situational awareness. In order for the swarm to spread out, the current area of interest must be divided among the swarm members will. Several coordination methods can be applied for this task including e.g. auctioning. In this paper, we will present an approach based on particle repulsion.

For each mobile sensor platform, the search can be divided in at least two phases: A preliminary phase during which the sensor moves towards the region of interest and the actual pursuit and intercept phase when the sensor has entered the region. Only the latter is considered in the present paper since it provides a proof of concept. At present only one sensor platform type is considered, that is, the swarm is homogeneous. All sensor platforms are able to intercept the target and all

are given the same task of intercepting the target as soon as possible updating the probability model with other sensors while they move through the graph. The approach can be extended to heterogeneous swarms where team members are tasked with sampling information and others with intercepting the target.

4.1. Swarm Search: A Stochastic Approach

Here, we focus on the interception phase of the search assuming thus that the swarm is in the vicinity of the target. If a vehicle s_k has reached a node u, it has to choose the next edge. Our approach is inspired by stochastic (meta)heuristics [6]. The basic equation for the probability of choosing an edge e reads

$$p_e(s_k, t) \quad = \quad \frac{p(e,t)^\alpha}{\sum_{e'=(u,v')} p(e',t)^\alpha} \tag{1}$$

for $e = (u, v)$ with parameter $\alpha \geq 0$. Good values for α will have to be determined in experiments. One point for further investigation is whether the edge used by s_k to travel to u shall be considered or not. Its probability could be set to zero in order to induce a wider movement. The exception to this rule are of course cul-de-sacs which can be determined by counting the incident edges of node u. In that case, the probability of choosing edge e must one. We will investigate both possibilities in the experimental section. Please note that these decisions are made when a vehicle arrives at an intersection.

4.2. Follow the Best: Incorporating Global Information

Equation (1) equals a stochastic greedy search. Since greedy approaches seldom lead to good overall solutions – even when coupled with stochastic selection, we incorporate an additional bias for "promising search directions". This approach is inspired by particle swarm optimization where each particle has a tendency to move the current best position. The underlying assumption is that the current best region is the best estimate for the global best solution and that by moving towards it, a swarm member has a good chance to identify further promising regions. Particle swarm optimization (PSO) [7] has been introduced in the 1990s and is usually seen as one of the more efficient metaheuristics for continuous optimization. It propagates a swarm of particles through the search space by adjusting the velocity vectors of the particles.

In our case, we try to improve the selection probability of promising directions defined by their distance to the current "best edges". Let $p_{\max}(t) := \max_{e \in E_U} p(e,t)$ and $E_{\max}(t) := \{e \in E_U | p_{\max}(t) = p(e,t)\}$. This requires the concept of a distance between the current edge $e = (u, v)$ and $E_{\max}(t)$, for instance defined by computing the shortest path between node v and both vertices of all edges $e' \in E_{\max}(t)$ and taking the minimum as the distance $d(e, E_{\max})$.

Another way to proceed is e.g. to determine the Euclidian distance between v and all vertices in $E_{\max}$ and to take the minimal value as an estimate for the distance. We will consider this approach in the later work. Switching to Euclidian distances will speed up the computation considerably if the graph is large. However, the quality of the estimate may be low which could limit the usefulness.

It remains to determine a suitable transformation of the distance into a bias for the probabilistic decision. Smaller distances shall be preferred, larger distances shall decrease the probability. This is fulfilled e.g. by

$$f(e,t) = \frac{1}{1 + \mathrm{e}^{d(v, E_{\max}(t))}}. \tag{2}$$

The updated probability for choosing an edge e reads

$$p_e(s_k, t) \quad = \quad \frac{p(e,t)^\alpha f(e,t)^\beta}{\sum_{e'=(u,v')} p(e',t)^\alpha f(e',t)^\beta} \tag{3}$$

for $e = (u, v)$ with parameters α, $\beta \geq 0$ and function f, (2). Until now, the situation that more than one vehicle at a time traverses an edge has not been considered. Since simultaneous searches of edges waste scarce resources, the probability of a sensor platform's choosing edges in the vicinity of other sensors has to be decreased. This holds of course only in cases where the sensor platform has sufficient capabilities to search the edge or node on its own.

4.3. Charged Swarms: Spreading the Search

For spreading the swarm, the concept of particle repulsion is transferred from charged swarms [3]. Charged swarms were introduced in particle swarm optimization to increase the diversity in a swarm. One application area is for instance dynamic optimization. In charged swarms, a part of the swarm consists of charged particles which repel each other if they are getting to close. This counteracts the typical tendency of the swarm to converge towards a search point after a time and enables it to cover a broader search region. In particle swarm optimization, the extend of the repulsion depends on the load carried by the particles and the distance (vector) of the particles to each other. As soon as the distance of two particles is smaller a predefined radius, the particles are both repelled in opposite directions along the line defined by the distance vector – proportional to the charge and reciprocally proportional to the square of the distance. To prevent vast and sudden position changes, a minimal distance was introduced limiting the repulsion magnitude. We will apply a similar concept for the search strategy.

Particle repulsion requires a measure for distances between sensor platform positions and other nodes. To this end, we define the d_I-neighborhood: A vehicle s_k is said to be in the d_I-neighborhood of a node v if s_k can reach v in d_I time units. This concept requires the determination of shortest paths between either the end or the start vertex of the edge the vehicle is currently located in. If the distances are only determined when a swarm member is in a vertex, the reference

point is of course the opposite vertex of the edge. Again, a faster but inaccurate distance estimate can be given by the Euclidian distance which will underestimate the distance in metric graphs. However, in metric graphs the Euclidian distance may serve to limit the number of sensors for which the shortest path has to be determined. Whether this leads to a speed up depends on the size of the graph and the number of sensors and will be investigated in a follow-up paper.

Recall that the sensor s_k under consideration is located at node u and considers edge $e = (u, v)$. Similar to charged particle swarm optimization variants [3], we use two distances, a minimal distance $d_{\min}$ and a influence distance d_I. The minimal distance is introduced as a safeguard to avoid potential numerical problems. The equations read

$$\begin{aligned} r(e, t, s_k) &= \frac{1}{1 + \mathrm{e}^{-d(e,S,s_k))}} \text{ with} \\ d(e, S, s_k) &= \sum_{s_j \neq s_k} \tilde{d}(e, s_j) \end{aligned} \tag{4}$$

and

$$\tilde{d}(e, s_j) = \begin{cases} 0 & \text{if} \quad d(e, s_j) > d_I \\ d(e, s_j) & \text{if} \quad d_{\min} \leq d(e, s_j) \leq d_I \\ d_{\min} & \text{if} \quad d_{\min} \geq d(e, s_j) \end{cases} \tag{5}$$

As for (2), other functional dependencies are possible in (4) and will be investigated in further work. Finally, we arrive at

$$p_e(s_k, t) = \frac{p(e, t)^\alpha f(e, t)^\beta r(e, t, s_k)^\gamma}{\sum_{e'=(u,v')} p(e', t)^\alpha f(e', t)^\beta r(e', t, s_k)^\gamma} \tag{6}$$

for $e = (u, v)$ with parameters α, $\beta, \gamma \geq 0$. In the following, the algorithm which applies (6) is called *simple probabilistic search* (SPS). Suitable values for all parameters have to be determined experimentally. An important point for later investigations is the performance robustness with respect to parameter values.

4.4. Taking a Look at Ant Colony Optimization: Exploiting the Information

A second algorithm is inspired by ant colony system (ACS)[5] – a specific ant colony optimization algorithm. Ant colony optimization (ACO) takes its inspiration from mathematical models of search behavior of ant colonies and belongs to the class of search and optimization metaheuristics. It has been applied successfully to several combinatorial problems ranging from (quadratic) assignment to vehicle routing problems [6]. Ant colony system is usually considered one of the best ACO-methods with respect to solution quality and efficiency [6]. Normally, ACO algorithms employ a similar stochastic rule to Eq. (6). Ant colony system follows a different approach: For each ant (a sensor in our case), ACS chooses the basic stochastic decision rule with a given probability. Otherwise, it aggressively exploits the information and chooses

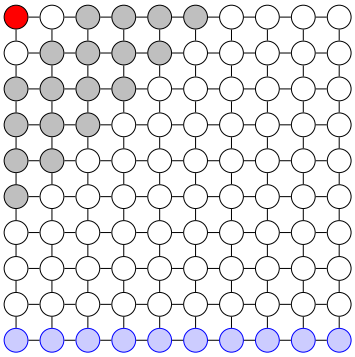


Figure 1. An exemplary $\mathbf{10 \times 10}$ graph (not used in the experiments). The target starts on the red node in the upper left corner. The gray nodes around the target indicate possible initial positions for the sensors if e.g. $\boldsymbol{di_{\max} = 4}$. The target chooses one of the nodes of the lower side as an exit point and follows then a shortest path.

the edge with the largest probability deterministically which in our case would translate to

$$e = \arg \max_{e'=(u,v')} p_{e'}(s_k, t). \tag{7}$$

The algorithm with probability p_{SPS} for choosing (6) and $1 - p_{SPS}$ for (7) will be referred to as *exploiting probabilistic search* (EPS).

5. First Experiments on Grids

For a proof of concept of the algorithms proposed the urban environment is represented as a $N \times N$ graph with equidistant nodes. The target enters over the node in the upper left corner of the graph and leaves the area over a randomly chosen node on the lower side following a shortest path (see Fig. 1). The target can only be intercepted while it remains in the area. The sensors are initialized on nodes relatively close to the target since we consider the pursuit and interception phase. For the initialization of the sensor positions, a minimal initial distance ($di_{\min}$) and a maximal distance ($di_{\max}$) are introduces. We allow two sensors to start on the same position.

It is assumed that the sensor can identify the target if they are either on the same node or edge or if the target is on one of the incident nodes of the sensor's edge. Both targets and sensors move from node to node in one time step. We simulate the uncertainty of the target position by a discrete probability distribution with n_c bins or classes with probabilities between $p_{\max}$ and $p_{\min}$ for all edges with at least one node within the influence radius ir of the target's current position. All other edges are given a small positive probability. The information is updated once the target has reached a new node. The radius remains constant in the simulation whereas the center of the distribution changes with the target's position. A simulation run ends when the target reaches its exit node. A run is called successful if the target is intercepted at least once by a sensor.

parameter	*significance*
N	no of nodes in a row/colum
n_S	no of sensors
$di_{\min}$	minimal initial distance to target
$di_{\max}$	minimal initial distance to target
ir	influence radius of prob. distribution
n_c	no of classes of discrete prob. distribution
$p_{\min}$	minimal edge probability for edges inside ir
$p_{\max}$	maximal edge probability
v_t	target speed
v_s	sensor speed
α	**exponent edge probability**
β	**exponent influence dist. to best edges**
γ	**exponent influence repulsion**
$d_{\min}$	**minimal repulsion distance**
$d_{\max}$	**maximal repulsion distance**
p_{SPS}	**prob. for choosing the SPS-rule (only EPS)**

Table 1. The parameter of the algorithms and the model variables. The parameter of the algorithms are set in bold face.

parameter	*default value*	*min*	*max*
N^*	15	15	20
n_S	3		
$di_{\min}$	2		
$di_{\max}$	6	5	6
$p_{\max}$	0.95		
$p_{\min}$	0.1		
n_c	5		
ir	5		
v_t	1		
v_s	1		
α^*	1	0	5
β^*	1	0	5
γ^*	1	0	5
$d_{\min}$	1		
$d_{\max}$	2		
p_{SPS}	0.1		

Table 2. The settings for the baseline runs. The influence of the parameters marked with $*$ will be investigated closer in the experimental section.

We consider the following performance measures for the algorithms. The first is the *success probability*, i.e., how often the target is intercepted by at least one sensor before it leaves the area

$$p_{succ} = \frac{\#\text{ successful runs}}{\#\text{ all runs}}. \quad (8)$$

For successful runs $\mathcal{SR}$, it is interesting to find out when the interception occurred. We will use *the expected relative interception time* (relative with respect to the time the target needed to reach the exit node)

$$\overline{T}_{ic}^{rel} = \frac{1}{|\mathcal{SR}|} \sum_{i \in \mathcal{SR}} \frac{\text{time to first intercept}(i)}{\text{target exit time}(i)}. \quad (9)$$

with $|\mathcal{SR}|$ the number of successful runs.

Since the difficulty of intercepting the target depends on the initial sensor positions and the choice of the exit point of the sensor, we consider the following experimental set-up. For each parameter configuration, we conduct 900 runs of the algorithms: 30 target exit points combined with 30 sensor initializations. Duplicates are allowed. For each of these combinations, we use 30 repeats because the algorithms themselves are also stochastic. Since the scenario states that the target is already being tracked by the sensors, we assume that they are close to the target but have had no visual contact. In other words, we set $di_{\min} = 2$ and $di_{\max} = ir/ir + 1$. Table 1 lists the variables of the model and the parameters of the algorithms. Please note that setting $\alpha = \beta = \gamma = 0$ equals a random search in the case of SPS with each edge having the same probability. EPS (as it is currently implemented) always takes the first edge with maximal probability and differs such from pure random search. Generally, experimental investigations on the influence of all parameters would be interesting. Due to space restrictions, we focus on a selection of the parameters we assume to have the greatest influence on the performance if neither swarm nor graph size is variable.

5.1. A First Look at the Approaches

Four algorithms with the baseline configuration given in Table 2 are considered

1) SPS: simple probabilistic search using Eq. (6),
2) SPS_NR: simple stochastic search without allowing immediate returns,
3) EPS: exploiting stochastic search using Eq. (6) with probability p_{SPS} and (7) with $1 - p_{SPS}$, and
4) EPS_NR: exploiting stochastic search without allowing immediate returns.

Comparing the results, we find that using EPS with a probability of 0.1 for the SPS-rule is apparently beneficial with respect to the success probability. In the case of EPS, success probabilities of $p_{succ} = 0.955$ (EPS) and $p_{succ} = 0.956$ (EPS_NR) can be measured compared to $p_{succ} = 0.912$ (SPS) and $p_{succ} = 0.872$ (SPS_NR).

The results also indicate that allowing immediate returns appears to have more influence on the SPS methods (in favor of allowance) whereas EPS appears more robust. There are no great differences concerning the expected mean relative time to first intercept. In the case of EPS, it reads $\overline{T}_{ic}^{rel} = 0.452$. If returns are not allowed, it lies lower at $\overline{T}_{ic}^{rel} = 0.429$ (EPS_NR). For SPS, we have $\overline{T}_{ic}^{rel} = 0.426$ and $\overline{T}_{ic}^{rel} = 0.411$ for SPS_NR.

Statistical tests (one-sided Mann-Whitney, $p = 0.01$) reject the zero hypothesis of equality of the means for returns allowed/not allowed EPS and SPS but it must be recalled that the data sets are large. The SPS methods appear to

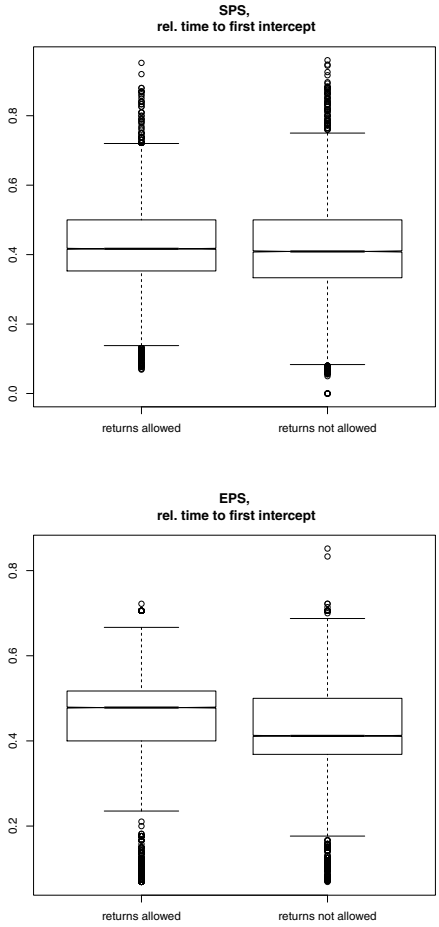


Figure 2. Box-plots of the relative time to first intercept for the basis scenario.

be slightly faster than the EPS approaches (see Figure 2). The box-plots show various outliers, in case of EPS very often towards small values. This could be due to initialization effects, however that would not explain why it appears more often for EPS. The findings so far may be affected by the size of the graph and further scenario parameters (initial distances, travel speeds) necessitating further experiments. Extrapolating from our findings, we gain the following hypotheses:

1) EPS/EPS_NR has a higher success probability compared to SPS/SPS_NR. This could be the effect of exploiting the information.
2) Allowing no immediate returns favors exploration and thus leads to a shorter time to the first intercept. However, this goes along with a lowered success probability for SPS.
3) The SPS variants find the target faster due to the increased exploration due to the stochastic nature of the SPS-rule (6). Since EPS_NR has an enforced exploration by disallowing immediate returns this effect could also be the cause for finding the target faster on average than EPS.

5.2. The Influence of the Exponents

The aim of this section is to investigate the influence of the parameters α, β, and γ. The main focus is to analyze the algorithms' robustness, i.e., whether an broader value range of these parameters exists for which acceptable results are obtained. This is crucial for practical use since otherwise a carefull fine-tuning is required. Optimal parameters are not determined since these are commonly very problem specific and their determination usually requires design-of-experiments techniques to explore the parameter space.

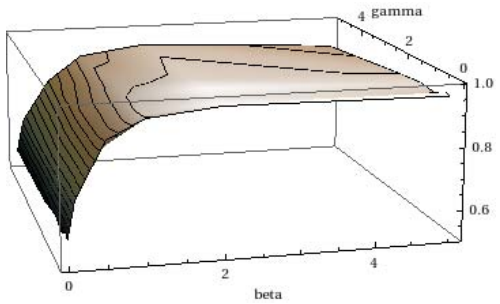
Please also note: In the case of metaheuristics, a rigorous theoretical determination of optimal parameters can usually be performed only for very simple scenarios or algorithms which limits the practical usefulness. Therefore, it is common practice to perform experimental investigations. The number of nodes in a row is set to $N = 20$ and $di_{\max} = ir = 5$. The influence distance is set to $d_{\max} = 1$ since for the scenario described larger values could open a path for the target which should be avoided. All other parameters are set to their default values.

5.2.1. Experimental Setup. We use a factorial design for all three parameters with values of $0, 0.1, 0.5, 1, 2, 5$ leading to 343 configurations for an algorithm. In the following we compare the results of the experiments for the two performance measures, the success probability and the expected relative time to the first intercept.

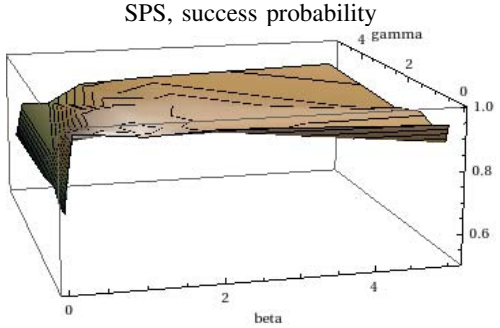
5.2.2. Success Probability. Tables 3 - 6 show exemplary the best and worst success probabilities grouped by β. Random search with $\alpha = \beta = \gamma = 0$ for SPS leads to a success probability of around 0.29. Other configurations with lower success probabilities (evader behavior, graph structure, and size) will be investigated in future research. It should be noted that some combinations lead to worse results: Spreading the swarm by using γ without directing the search with α and β apparently opens a path on the grid for the target.

A clear finding emerges: On a grid with size 20×20 and for three searchers, the parameter β which decides on the influence of the global information is an important factor. In the case of EPS, setting $\beta = 0$ may result in weak performance on the grid if α and γ are not chosen correctly. Otherwise, it is remarkably robust with worst success probabilities above 0.78 for $\beta > 0$. The same holds for EPS_NR, which also exhibits good performance if $\beta > 0$. The SPS variants appear more sensitive than EPS concerning the worst performances. While they often surpass EPS for the best success probability, a wrong parameter setting can affect them strongly. However, the experiments also showed that the SPS variants become robust for $\beta \geq 1$.

The differences between SPS- and EPS-variants with respect to the size of β could be explained by considering the different



SPS, success probability



EPS, success probability

Figure 3. 3D-plots for the success probability for $\boldsymbol{\alpha = 2}$.

search behavior. Due to the stronger stochastic influence, the SPS-approaches have a stronger inclination for exploration than EPS. Larger β-values focus the search towards the present target position so that SPS-variants generally benefit by an increase of the parameter. The three-dimensional plots, see Fig. 3 for $\alpha = 2$, show a nearly logistic growth curve with the increase flattening or even reversing slightly after $\beta = 2$. On the simple grid and with a target following the shortest path to its exit node, a strong focus on the global information appears favorable for SPS to conteract its strong exploration tendency to a certain point. The EPS variant which searches less stochastic is sooner influenced by the global information and thus works well with small β-values. As depicted in Table 3, EPS shows already a slight decline in the best and worst success probability for $\beta = 5$, indicating the beginning of a too strong bias towards the global information. However, it should be noted that the values (best/worst) are still above 0.88. These findings are affected by the uncertainty model applied. The edges with the largest probabilities are good estimates for the true target position. An investigation of different types of uncertainties is an interesting point for future research. Also the probability p_{SPS} for choosing the SPS-rule has an additional important influence. This will be investigated on more realistic graphs in future experiments.

Interestingly, the influence of α on the performance is less clear that for β. It has the strongest influence on the worst success probability. While the data does not show a clear trend in the case of the best success probability, the worst success probability generally increases with α. Strengthening the influence of the repulsion distances is generally not favorable in

β	*best success probability*	*worst success probability*
0	0.874 (α,γ)=(2,0)	0.031 (α,γ)=(0.0,5)
0.1	0.974 (α,γ)=(0,0)	0.835 (α,γ)=(5,1)
0.5	0.971 (α,γ)=(0.1,0)	0.853 (α,γ)=(5,5)
1	0.965 (α,γ)=(0.5,0)	0.877 (α,γ)=(5,5)
2	0.965 (α,γ)=(1,0)	0.869 (α,γ)=(5,5)
5	0.961 (α,γ)=(5,0)	0.833 (α,γ)=(1,5)

Table 3. Success probabilities for EPS grouped by the parameter $\boldsymbol{\beta}$.

the scenario. The worst success probabilities are encountered for $\gamma = 5$. This can traced back to the swarm size and the scenario conditions.

β	*best success probability*	*worst success probability*
0	0.731 (α,γ)=(5,0.5)	0.278 (α,γ)=(0,5)
0.1	0.818 (α,γ)=(5,1)	0.419 (α,γ)=(0,5)
0.5	0.909 (α,γ)=(5,0)	0.727 (α,γ)=(0,5)
1	0.959 (α,γ)=(5,0)	0.882 (α,γ)=(0,5)
2	0.972 (α,γ)=(0,0.5)	0.940 (α,γ)=(5,5)
5	0.974 (α,γ)=(1,0.5)	0.903 (α,γ)=(2,5)

Table 4. Success probabilities for SPS grouped by the parameter $\boldsymbol{\beta}$.

5.2.3. Expected Mean Relative Time to First Intercept. The expected mean relative time to the first intercept was also measured in the experiments. Table 7 - 10 show the best and the worst $\overline{T}_{ic}^{rel}$ grouped by β. The values do not differ much with the best interception times lying around 0.4. The variants which do not allow immediate returns are often faster than the original approaches and SPS can achieve better first interception times than EPS. Again, ultilizing global information speeds up the search. The EPS variants require only $\beta > 0$, whereas the SPS variants benefit from larger values of β as Fig. 4 illustrates for $\alpha = 2$.

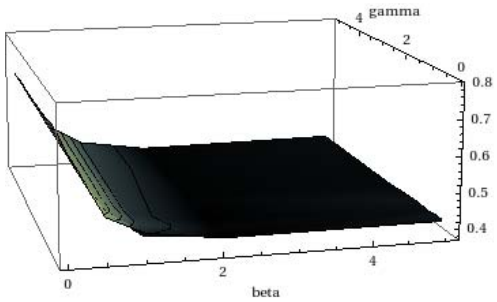
Interestingly, the experiments also hint at that choosing β too large can prolong the search – albeit not much for the parameter settings considered. The reasons for this must be investigated further but apparently while increased exploration lessens the chance of finding the target, it decreases the time in successful trials.

β	*best success probability*	*worst success probability*
0	0.637 (α,γ)=(5,0)	0.171 (α,γ)=(0,5)
0.1	0.974 (α,γ)=(0.1,0)	0.873 (α,γ)=(5,5)
0.5	0.976 (α,γ)=(2,0)	0.868 (α,γ)=(2,5)
1	0.980 (α,γ)=(0.1,0)	0.938 (α,γ)=(0.1,5)
2	0.974 (α,γ)=(0,0)	0.925 (α,γ)=(5,2)
5	0.974 (α,γ)=(0.5,0)	0.881 (α,γ)=(2,0.1)

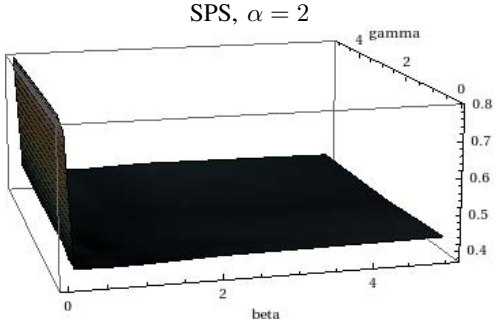
Table 5. Success probabilities for EPS_NR grouped by the parameter $\boldsymbol{\beta}$.

β	*best success probability*	*worst success probability*
0	0.609 (α,γ)=(5,5)	0.272 (α,γ)=(0,5)
0.1	0.723 (α,γ)=(5,0.1)	0.370 (α,γ)=(0,5)
0.5	0.882 (α,γ)=(5,0.1)	0.676 (α,γ)=(0,5)
1	0.941 (α,γ)=(5,0.1)	0.835 (α,γ)=(0.1,5)
2	0.968 (α,γ)=(0.5,0.5)	0.927 (α,γ)=(0,5)
5	0.969 (α,γ)=(0.5,0.1)	0.930 (α,γ)=(0.5,5)

Table 6. Success probabilities for SPS_NR grouped by the parameter β.



SPS, $\alpha = 2$



EPS, $\alpha = 2$

Figure 4. The relative time to first intercept.

6. Outlook

We have presented two new algorithms for a pursuit and interception game based on a occurrence during an international real-life experiment and provided a first experimental analysis. The task was to intercept a mobile target in an urban environment represented as a grid. The evader was assumed to behave deterministically – choosing an exit point and following a shortest-path. Once the target has exited the area, the opportunity for intercepting is lost. This situation differs from usual pursuit-evasion games which assume that the target remains in the area since all exit points can be closed. The evader's behavior is straightforward but since a shortest path is followed, the time for interception is relatively short. The algorithms combine concepts stemming from particle swarm optimization and ant colony optimization. Both algorithm types lead to good results and are quite robust with regard to the parameter setting.

Further work will consider more realistic and complicated graph topologies taken from urban street maps. Design of computer experiments methodology will be applied switching to space-filling designs [2] to provide an in-depth analysis of the algorithms. Also, sensor networks with a limited communication radius will be addressed considering at least two main approaches: One is to maintain the cohesion of the swarm similar to swarm approaches in underwater terrain covering. The other is to allow a wide spread of the swarm and thus different beliefs of the target position. The latter will increase the exploration behavior in contrast to the former. There are strong similarities to sparsely connected neighborhood structures for PSO which will be investigated by introducing personal and neighborhood estimations of the target position. The present paper did not made any assumptions on the probability model for the target position since it represents a proof of concept. However, the framework can be extended by adapting the probability model.

β	*best* $\overline{T}_{ic}^{rel}$	*worst* $\overline{T}_{ic}^{rel}$
0	0.100 (α,γ)=(0,5)	0.815 (α,γ)=(5,5)
0.1	0.412 (α,γ)=(0.1,0)	0.451 (α,γ)=(5,1)
0.5	0.410 (α,γ)=(0.1,0)	0.441 (α,γ)=(5,5)
1	0.417 (α,γ)=(0.5,0)	0.445 (α,γ)=(5,5)
2	0.422 (α,γ)=(0.5,0)	0.444 (α,γ)=(5,5)
5	0.430 (α,γ)=(0.1,0)	0.451 (α,γ)=(5,5)

Table 7. EPS: mean relative time to first intercept grouped by the parameter β.

β	*best* $\overline{T}_{ic}^{rel}$	*worst* $\overline{T}_{ic}^{rel}$
0	0.420 (α,γ)=(0,0)	0.727 (α,γ)=(5,5)
0.1	0.432 (α,γ)=(0,2)	0.679 (α,γ)=(5,5)
0.5	0.412 (α,γ)=(0,0.1)	0.529 (α,γ)=(5,5)
1	0.400 (α,γ)=(0.1,0)	0.452 (α,γ)=(5,5)
2	0.397 (α,γ)=(0,0.5)	0.423 (α,γ)=(5,5)
5	0.416 (α,γ)=(0.5,5)	0.427 (α,γ)=(5,0.1)

Table 8. SPS: mean relative time to first intercept grouped by the parameter β.

References

[1] Alspach, B. (2004). Searching and sweeping graphs: a brief survey. *Matematiche 59*(1-2), 5–37.

[2] Bartz-Beielstein, T., M. Chiarandini, and L. Paquete (Eds.) (2010). *Experimental Methods for the Analysis of Optimization Algorithms*. Springer.

[3] Blackwell, T. M. and P. J. Bentley (2002). Dynamic search with charged swarms. In *Proceedings of the Genetic and Evolutionary Computation Conference*, GECCO '02, San Francisco, CA, USA, pp. 19–26. Morgan Kaufmann Publishers Inc.

[4] Chung, T., G. Hollinger, and V. Isler (2011). Search and pursuit-evasion in mobile robotics. *Autonomous Robots 31*, 299 – 316.

[5] Dorigo, M. and L. Gambardella (1997). Ant colony system: a cooperative learning approach to the traveling

β	*best* $\overline{T}_{ic}^{rel}$	*worst* $\overline{T}_{ic}^{rel}$
0	0.512 (α,γ)=(0,1.0)	0.700 (α,γ)=(5,0.5)
0.1	0.395 (α,γ)=(0,0.5)	0.439 (α,γ)=(5,2)
0.5	0.408 (α,γ)=(0,0.5)	0.435 (α,γ)=(5,5)
1	0.408 (α,γ)=(2,2)	0.429 (α,γ)=(5,5)
2	0.414 (α,γ)=(0.5,0.1)	0.429 (α,γ)=(5,0.1)
5	0.418 (α,γ)=(0,1)	0.433 (α,γ)=(0.5,0.1)

Table 9. EPS_NR: mean relative time to first intercept grouped by the parameter β.

β	*best* $\overline{T}_{ic}^{rel}$	*worst* $\overline{T}_{ic}^{rel}$
0	0.457 (α,γ)=(0.1,0)	0.654 (α,γ)=(5,5)
0.1	0.447 (α,γ)=(0,0.1)	0.612 (α,γ)=(5,5)
0.5	0.409 (α,γ)=(0,2.0)	0.482 (α,γ)=(5,0.5)
1	0.392 (α,γ)=(0,0.5)	0.431 (α,γ)=(2,5)
2	0.392 (α,γ)=(0.5,2)	0.413 (α,γ)=(1,0.1)
5	0.404 (α,γ)=(0.5,2)	0.415 (α,γ)=(2,5)

Table 10. SPS_NR: mean relative time to first intercept grouped by the parameter β.

salesman problem. *Evolutionary Computation, IEEE Transactions on 1*(1), 53 – 66.

[6] Dorigo, M. and T. Stützle (2004). *Ant Colony Optimization*. MIT Press.

[7] Engelbrecht, A. P. (2005). *Fundamentals of Computational Swarm Intelligence*. Wiley.

[8] Gerkey, B. P., S. Thrun, and G. Gordon (2005). Parallel stochastic hill-climbing with small teams. In L. E. Parker et al. (Eds.), *Multi-Robot Systems: From Swarms to Intelligent Automata*, Volume III, pp. 65–77. Springer.

[9] Hudgens, B. and A. Bordetsky (2011). Through a blurred lens: cyber distortion in command and control. In *Proceedings Networking and Electronic Commerce Conference*.

[10] Kolling, A. (2009). *Multi-Robot Pursuit-Evasion*. Ph. D. thesis, University of California, Merced.

[11] Kolling, A. and S. Carpin (2010). Pursuit-evasion on trees by robot teams. *Trans. Rob. 26*(1), 32–47.

[12] Parsons, T. D. (1976). Pursuit-evasion in a graph. In Y. Alavi and D. Lick (Eds.), *Theory and Applications of Graphs*, Lecture Notes in Mathematics, pp. 426–441. Springer.

[13] Sato, H. and J. O. Royset (2010). Path optimization for the resource-constrained searcher. *Naval Research Logistics (NRL) 57*(5), 422 – 440.

[14] Vidal, R., O. Shakernia, H. Kim, D. Shim, and S. Sastry (2002). Probabilistic pursuit-evasion games: theory, implementation, and experimental evaluation. *Robotics and Automation, IEEE Transactions on 18*(5), 662 – 669.