View metadata, citation and similar papers at core.ac.uk

brought to you by CORE

provided by IntechOpen

We are IntechOpen,
the world's leading publisher of
Open Access books
Built by scientists, for scientists

4,800
Open access books available

122,000
International authors and editors

135M
Downloads

Our authors are among the

154
Countries delivered to

TOP 1%
most cited scientists

12.2%
Contributors from top 500 universities

WEB OF SCIENCE™
Selection of our books indexed in the Book Citation Index
in Web of Science™ Core Collection (BKCI)

Interested in publishing with us?
Contact book.department@intechopen.com

Numbers displayed above are based on latest data collected.
For more information visit www.intechopen.com

23

Modeling with Non-cooperative Agents: Destructive and Non-Destructive Search Algorithms for Randomly Located Objects

Dragos Calitoiu[1] and Dan Milici[2]
[1]*Carleton University*
[2]*Stefan cel Mare University*
[1]*Canada*
[2]*Romania*

The problem of plastic anti-personal mine detection is well known. These devices are typically 75mm in diameter and between 25mm to 50mm in thickness. Some types contain no metal, but their explosive (RDX, TNT) can be considered as a dielectric material with a dielectric constant between 3.5 and 4.0. This electromagnetic property enables a radar to identify mines. It is true that the radars operating "through the air" provide a high detection rate. However, in the case of operating "into the ground", the problems are significant and inherently reduce performance. Target identification is done in the presence of the air-ground interface, which usually produces a higher amplitude signal than an anti-personal mine Chignell (1998). This context can justify why the difficulties of detecting anti-personal mines are formidable. A family of search mobots controlled by a new algorithm proposed in this research could be a solution[1] for operating on the ground.

In this Chapter we[2] address the general question of what is the best strategy to search efficiently for randomly located objects (target sites). We propose a new agent based algorithm for searching in an unpredictable environment. The originality of our work consists in applying a non-cooperative strategy, namely the distributed Goore Game model, as opposed to applying the classical collaborative and competitive strategies, or individual strategies. This research covers both the destructive search and the non-destructive search. The first occurs when the agent visits the target only one time. The latter can be performed in either of the two cases - if the target becomes temporarily inactive or if it leaves the area.

The proposed algorithm has two versions: one when the agent can move with a step equal to unity and the other when the step of the agent follows a Levy flight distribution. The second version is inspired by the work of A.M. Reynolds *et al.* Reynolds (2006a;b; 2007; 2008a;b; 2009); Reynolds & Rhodes (2009); Rhodes & Reynolds (2007) where he braced the use of Levy processes when resources are sparsely distributed within unpredictable environments.

The Chapter is organized as follows. The Goore Game is presented in the next section. In Section 2 we introduce the terminology of the Learning Automata, methodology used for

[1] The main application of our proposed algorithm is anti-personal mine detection. However, this research can be extended to any type of exploration on ground or aerial using uninhabited aerial vehicle (on Earth or for conducting planetary science missions).
[2] The first author is also member of OPTIMOD Research Institute, Ottawa.

implementing the behavior of the players for the Goore Game. The motivation due to Levy flight for selecting the search step is described in Section 3. The proposed search algorithm and the results of the simulations are presented in Section 4 (for destructive search) and Section 5 (for non-destructive search). Finally, Section 6 concludes the Chapter.

1. Goore game

Goore Game is an example of self-organization and self-optimization game studied in the field of artificial intelligence. It was presented by Tsetlin in 1963 Tsetlin (1963) and analyzed in detail in Narendra & Thathachar (1989) and Thathachar & Arvind (1997). The informal formulation follows.

"*Imagine a large room containing N cubicles and a raised platform. One person (voter) sits in each cubicle and a Referee stands on the platform. The Referee conducts a series of voting rounds as follows. On each round the voters vote either "Yes" or "No" (the issue is unimportant) simultaneously and independently (they do not see each other) and the Referee counts the fraction, f, of "Yes" votes. The Referee has a unimodal performance criterion $G(f)$, which is optimized when the fraction of "Yes" votes is exactly f_0. The current voting round ends with the Referee awarding a dollar with probability $G(f)$ and assessing a dollar with probability $1 - G(f)$ to every voter independently. On the basis of their individual gains and losses, the N voters then decide, again independently, how to cast their votes on the next round. No matter how many players there are, after enough trials, the number of "Yes" votes will approximate Nf_0.*"

Each player plays solely in a greedy fashion, voting each time the way that seems to give the player the best payoff. This is somewhat unexpected. Greed affects outcomes in an unpredictable manner: a player does not attempt to predict the behavior of other players. Instead, each player performs by trial and error and simply preferentially repeats those actions that produce the best result for that player.

The essence of the Goore Game is a random walk that is strongly biased toward the global optimum. Some of the game's features Oommen et al. (1999) which render it both non-trivial and intriguing are:

- The game is a non-zero-sum game.
- Unlike the games traditionally studied in the AI literature (Chess, Checkers, etc.) the game is essentially a distributed game.
- The players of the game are ignorant of all of the parameters of the game. All they know is that they have to make a choice, for which they are either rewarded or penalized. They have no clue as to how many other players there are, how they are playing, or even of how/why they are rewarded/penalized.
- The stochastic function used to reward or penalize the players, after measuring their performance as a whole, can be completely arbitrary, as long as it is uni-modal.
- The game can achieve a globally optimal state with N-players without having to explicitly dictate the action to each player. The players self-organize and self-optimize based on the reward function.

The Goore Game can be representative for many real-life scenarios as recruitment of motor units Thathachar & Arvind (1997) (as working muscles) to perform a certain task, such as exerting a force to lift a weight. In this setting, each motor unit contributes either a fixed magnitude of force or none at all. Function of the specificity of the job, it will be mandatory to recruit the correct number of motor units. If there are more motor units than actually needed, this will exert more force than necessary. On contrast, if there are less motor units, they may

not be able to perform the task at all. The problem is one of employing the right number of working units to perform the task.
The game was initially studied in the general learning domain and was considered an interesting pathological game. Recently, it was applied in QoS (Quality of Service) support in wireless sensor networks Iyer & Kleinrock (2003), Chen & Varshney (2004), in controlling wings with integrated check-valve electrostatic actuators Ho et al. (2002), and in cooperative mobile robotics Cao et al. (1997), Tung & Kleinrock (1996).

2. Some fundamentals of learning automata

Due to the solid theoretical basis that has been established within the Learning Automata (LA) field during the last decades, and particularly due to the results concerning games, we decided to implement our searching algorithm using LA.
LA have been used Narendra & Thathachar (1989), Poznyak & Najim (1997), Lakshmivarahan (1981) to model biological learning systems and to find the optimal action that is offered by a Random Environment[3]. Learning is realized by interacting with the Environment and by processing its responses to the actions that are chosen, while gradually converging toward an ultimate goal. The environment evaluates the performance of LA and directs the learning process performed by the LA. Thus, the LA's overall performance is gradually improved. In this respect, the process of learning is based on a learning loop involving two entities: the Random Environment (RE) and the LA.
The RE offers to the automaton a set of possible actions $\{\alpha_1, \alpha_2, \dots, \alpha_r\}$ to choose from. The automaton chooses one of those actions, say α_i, which serves as an input to the RE. Since the RE is "aware" of the underlying penalty probability distribution of the system, depending on the *penalty probability* c_i corresponding to α_i, it produces a response (β) to the LA that can be a *reward* (typically denoted by the value $\beta = 0$), or a *penalty* (typically denoted by the value $\beta = 1$). The reward/penalty information (corresponding to the action) provided to the LA helps it to choose the subsequent action. By repeating the above process, through a series of Environment-Automaton interactions, the LA finally attempts to learn the *optimal* action from the Environment. The Learning process is presented in Figure 1.

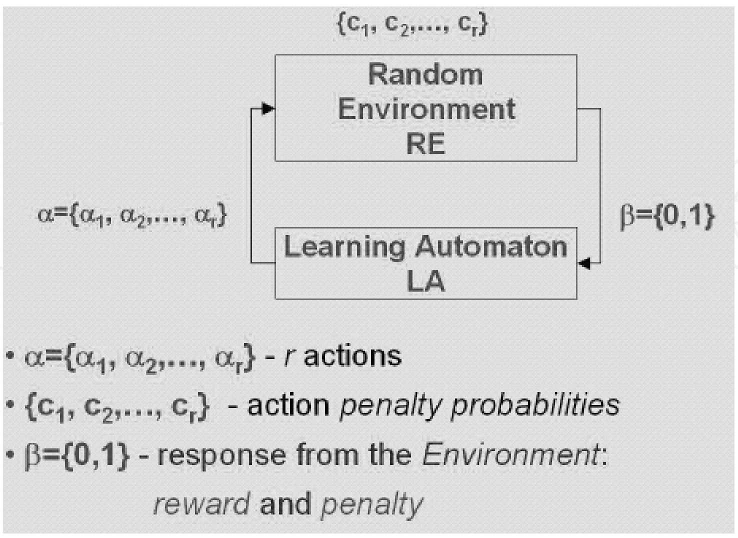

{c1, c2,..., cr}
Random Environment RE
α={α1, α2,..., αr}
β={0,1}
Learning Automaton LA
• α={α1, α2,..., αr} - r actions
• {c1, c2,..., cr} - action penalty probabilities
• β={0,1} - response from the Environment: reward and penalty

Fig. 1. The Learning process as a loop of interactions between the LA and Environment.

[3] The first author thanks Dr. John B. Oommen for introducing him in the field of Learning Automata.

2.1 The continuous linear reward-inaction scheme

When the Goore Game was first investigated, Tsetlin utilized his so-called Tsetlin automaton to solve it. Later, more research was done in the LA area and many families of LA proved to solve the Goore Game efficiently. In our research, we are using a very fast LA: the continuous Linear Reward-Inaction scheme (L_{RI}) that was introduced by Norman Norman (1968). This scheme is based on the principle that whenever the automaton receives a favorable response (i.e., a reward) from the environment, the action probabilities are updated, whereas if the automaton receives an unfavorable response (i.e., a penalty) from the environment, the action probabilities are unaltered.
The probability updating equations for this scheme are characterized by a parameter θ ($0 < \theta < 1$) and can be simplified to be as below Norman (1968):

$$\begin{array}{l} p_1(t+1) = p_1(t) + (1-\theta) \times (1 - p_1(t)) \\ \text{ if } \alpha(t) = \alpha_1, \text{ and } \beta(t) = 0 \\ p_1(t+1) = \theta \times p_1(t) \\ \text{ if } \alpha(t) = \alpha_2, \text{ and } \beta(t) = 0 \\ p_1(t+1) = p_1(t) \\ \text{ if } \alpha(t) = \alpha_1 \text{ or } \alpha_2, \text{ and } \beta(t) = 1 \end{array} \tag{1}$$

Note that if action α_i is chosen, and a reward is received, the probability $p_i(t)$ is increased, and the other probability $p_j(t)$ (i.e., $j \neq i$) is decreased. If either α_1 or α_2 is chosen, and a penalty is received, $P(t)$ is unaltered.
Equation (1) shows that the L_{RI} scheme has the vectors $[1,0]^T$ and $[0,1]^T$ as two absorbing states - one of which it converges to. Therefore, the convergence of the L_{RI} scheme is dependent on the nature of the initial conditions and probabilities.

3. Levy flight

The solution presented in this research has two components: the first one is motivated by the strategy involving distributed control; the second one is inspired by the moves made by the animal world. In the previous sections, we introduced the Goore Game and its implementation with LA. In this section, we present the inspiration from the animal world.
Animals move for various reasons: to search for sources of food that are not accessible in the immediate vicinity of the animal, to search for a mate, to avoid predators, to visit a watering hole or to search for a site on which to lay eggs Reynolds & Rhodes (2009). The complexity of the searching strategy depends on the knowledge about the environment. If the environment is unchanging or wholly predictable, animals may develop knowledge of where to locate resources and exploit that knowledge. However, where resource availability is unknown or unpredictable (and this is the scenario investigated in our research), animals have to conduct non-oriented searches with little or no prior knowledge of where resources are distributed. Consequently, the capability to locate resources efficiently will minimize the risk of starvation and potentially minimize exposure to competitors and predators.
The first suggestion that movement patterns of some biological organisms may have Levy flight characteristics came from Shlesinger and Klafter Shlesinger & Klafter (1986). These particular movements consist in random sequences of independent flight-segments whose lengths, l, are drawn from a probability distribution function, having a power-law tail

$$p(l) \approx l^{-\mu}, \tag{2}$$

where $1 < \mu < 3$.

Over recent years, theoretical and field studies provided evidence that many organisms adopt Levy flight[4] movement pattern when they are searching for resources. Honeybees (Apis mellifera), Drosophhila, aphid, microzooplankton, wandering albatross (Diomedea exulans) are a few examples Reynolds (2006a)-Reynolds (2008a), Benhamou (2008).

In general, individual random Levy-flight searching strategies are less efficient than an equidistant (Archimedian) spiral search. However, such a spiral search can only work if navigation and target detection are precise enough geometrically to ensure that all areas are explored and the intervening regions are not missed. In the scenario when the objective of the search is missed, there is no second chance of encountering it because the trajectory is an ever-expanding spiral. A.M. Reynolds mentioned in Reynolds (2008b) that adopting a spiral-search pattern would be disastrous if the navigation and detection systems are even slightly erroneous. A systematic spiral-searching strategy can only be used initially, when cumulative navigational errors are relatively small but should be abandoned in favour of a random looping-searching strategy at latter times.

For the completeness of our analysis, we present now the comparison between individual Brownian search and Levy flight search. For many years, the Brownian motion was the most used model for describing non-oriented animal movement Kareiva & Shigesda (1983)-Okubo & Levin (2002). An individual trajectory through space is regarded as being made up of a sequence of distinct, randomly oriented move step-lengths drawn from a Gaussian distribution. The main difference between the Brownian walk and the Levy flight is due to what (or who) decides the length of the movements[5]. In the Brownian walk, it is considered that the scale of the movement is defined by the organism; in contrast, in the Levy flight, the larger scale of the movement is determined by the distribution of the targets (food or prey). This difference explains why Levy flight is more flexible and suitable for the scenario when the animal has to adapt to the environmental changes.

The last conclusion encourages exploring the maximization of search efficiency. All of the above comments on searching strategies are presented in the context of individuals. However, we propose an algorithm that is collective and has a distributed control: it is able to coordinate independent searching agents, in a des-centralized fashion.

Until now, in addition to the individual models of search for food, the collective models used were developed in a collaborative and competitive manner (see Particle Swarm Optimization, where an entire swarm - flock of birds, school of fish, herd of insects - has a collaborative[6] effect of searching for food). To the best of our knowledge, this is the first research to present a search algorithm using non-cooperative players.

4. Destructive search

This section contains details regarding the algorithm and the corresponding results associated to the destructive search.

[4] Strictly speaking, the pattern identified should be named Levy walk because it contains continuous movements rather than discrete jumps. However, we are following the literature that uses Levy flight and Levy walk synonymously.

[5] The main consequence of this difference can be evaluated in swarms or flocks Cai et al. (2007); Viswanathan et al. (1996; 1999). In t steps, a Brownian-walker unit visits $t/\ln(t)$ new sites whereas a Levy flight unit visits t. However, in t steps, a swarm of N Brownian-walker units visit $t\ln(N/\ln(t))$ new sites whereas a swarm of N Levy flight units visit Nt.

[6] Examples of wolves making choices in how to search an area for food can be applied to optimize autonomous vehicles in search of data Plice et al. (2003).

4.1 The algorithm for the destructive search

The algorithm contains two types of actions that each LA can select: *Action 1* means "move", whereas *Action 2* means "stay". The "search" process can be added to each of these actions, namely we can have *Action 1* as "move" and "search", and *Action 2* as only "stay" or *Action 1* as only "move", and *Action 2* as "stay" and "search".
The formal procedure is presented below:

```
Initialization:
Randomly distribute $N$ agents is the area of radius $R_1$;
Randomly associated to each agent a probability $P_{i1}$;
Set $Flag\_i\_move$=0 for each agent;

FOR $j = 1$ TO $Nr\_iter$ DO:
Moving the agents:
For each agent $i$ DO:
Generate a random number $M_i$;
IF $M_i < P_{i1}$ THEN DO:
$Flag\_i\_move$=1;
Randomly rotate the agent;
Compute $D_{ij}$ the distance between the agent and the rest of $N - 1$ ;
IF (neighborhood is empty) AND ($D_{ij} < R_2$) AND (neighborhood is unvisited)
THEN (MOVE 1 Step);
END IF;
END IF;
END DO;

Feedback from Referee:
Compute $f = \sum(Flag\_i\_move)$ for all $N$ agents;
Compute $G(f)$;

Updating probabilities for agents
FOR each agent DO:
Generate a random number $S_i$;
IF ($S_i < G(f)$) THEN:
IF ($flag\_i\_move$ = 1) THEN increase $P_{i1}$
ELSE decrease $P_{i1}$;
END IF;
END IF;
END DO;
Set $Flag\_i\_move$=0 for each agent;

END DO;
```

Remarks:

- As we mentioned earlier, our proposed algorithm can be applied on two versions, function of how the resources are distributed in the environment.
  1. Version 1: the movement step is equal to unity.

2. Version 2: the movement step follows a Levy flight distribution described by $l \in [1;5]$ and $\mu = 2$).

- A comment can be made regarding the order of actions to be taken. From the decision point of view, if the condition $M_i < P_{i1}$ is satisfied, the agent will select *Action 1* (in our case, the agent is allowed to move). The agent will be rewarded by the Referee (in general, by the Environment) function of the decision taken. The second step after making the decision is to see if he has an empty space unvisited where to move AND if he satisfies the distance condition: his distance to the farthest neighbor agent must be less than R_2. If these conditions are satisfied, the agent can move on the field. We presented these aspects in detail to make it easier for the reader to understand the difference between the decision of selecting *Action 1* (to move) and the real process on the field. The probabilities are updated only function of the decision and not of the real move on the field. It is true that it is possible to test a new version of the algorithm, namely version 3, where the $Flag\_i\_move$ becomes equal to unity only after the movement step is completely done, namely (MOVE 1 STEP) AND ($Flag\_i\_move$). However, the distance condition is very powerful and will affect the convergence provided by the GG algorithm. The version 3 can succeed if the settings θ (the constant involved in updating probabilities), N, R_1 and R_2 are carefully selected.

4.2 Simulation results

In this subsection, we present four families of simulations: one obtained with Version 1 of the algorithm and three obtained with different settings of Version 2 of the algorithm (namely three different values for the length of the movement step). We depict only the graphical results obtained after enough iterations that are not allowed more movements, for each version of the proposed algorithm. However, we present in Table 1 the area covered by N=10 agents (players) in four settings: Version 1 and Version 2 with the same $\mu = 2$ and $l = 2$, $l = 3$ and $l = 4$. Each setting was run 25 simulations. In each of these settings, we used $N = 10$ agents, with $R_1 = 10$ and $R_2 = 40$. All the agents are L_{RI} with $\theta = 0.1$. The performance criterion $G(\cdot)$ used by the Referee is $G(x) = 0.9 \times e^{-\frac{(0.7-x)^2}{0.0625}}$, as presented in Figure 2. The reader can see that the maximum was covered from Version 1; the increase in the length of the movement decreases the total area covered by the agents.
In Figure 3 (left), we present the result corresponding to Version 1 of the algorithm, namely when the movement step is equal to unity. In Figures 3 (right) and 4 we present the results corresponding to Version 2 of the algorithm, namely when the movement step follows a Levy flight with $l = 2$ in the first graph, with $l = 3$ in the second graph, and $l = 4$ in the third graph. The exponent of the distribution is always $\mu = 2$. The reader can observe the size of the steps as being very close to the agents that cannot move. Many of them have a probability P_1 equal to unity. Thus, they can move from the point of view of the decision maker. However, the distance constraint will not allow them to realize the movement on the field.

5. Non-destructive search

This section contains details regarding the algorithm and the corresponding results associated to the non-destructive search.

5.1 The algorithm

As in previous section, the algorithm contains two types of actions that each LA can select: *Action 1* means “move”, whereas *Action 2* means “stay”. The “search” process can be added

Iteration	Version1	$l = 2$	$l = 3$	$l = 4$
1	218	516	406	381
2	196	337	535	455
3	185	584	370	428
4	147	608	471	526
5	132	490	569	412
6	190	510	366	524
7	134	511	411	368
8	180	703	625	534
9	184	433	452	429
10	197	582	462	331
11	138	443	638	486
12	129	569	556	280
13	332	389	441	510
14	127	450	435	441
15	165	525	502	400
16	168	300	540	538
17	218	302	521	394
18	235	422	364	504
19	217	430	588	385
20	225	372	417	597
21	252	441	474	544
22	247	501	391	346
23	130	432	624	373
24	111	568	405	430
25	113	428	383	360
Mean	182.80	473.84	477.84	439.04
Standard Deviation	52.98	97.71	86.90	79.96

Table 1. Destructive Search: The area covered by N=10 agents in four settings: Version 1 ($l = 1$) and Version 2 (with the same $\mu = 2$ and $l = 2$, $l = 3$ and $l = 4$). Each setting was run 25 simulations. The average and the standard deviation are presented.

to each of these actions, namely we can have *Action 1* as "move" and "search", and *Action 2* as only "stay" or *Action 1* as only "move", and *Action 2* as "stay" and "search".
The formal procedure is presented below:

```
**Initialization**:
Randomly distribute $N$ agents is the area of radius $R_1$;
Randomly associated to each agent a probability $P_{i1}$;
Set $Flag\_i\_move$=0 for each agent;

FOR $j = 1$ TO $Nr\_iter$ DO:
**Moving the agents:**
For each agent $i$ DO:
Generate a random number $M_i$;
IF $M_i < P_{i1}$ THEN DO:
```

Iteration	Version1	$l = 2$	$l = 3$	$l = 4$
1	876	822	716	613
2	848	757	766	589
3	1031	1017	745	623
4	1024	906	564	516
5	1021	993	752	796
6	766	834	782	528
7	928	750	572	606
8	1140	897	533	451
9	873	1006	882	374
10	1042	934	860	666
11	903	762	803	476
12	898	921	779	363
13	947	893	654	475
14	855	1086	857	434
15	1261	674	604	553
16	1062	950	643	697
17	1016	680	575	509
18	1112	831	727	610
19	983	955	1096	674
20	1178	499	716	365
21	1029	879	694	649
22	1011	958	740	573
23	908	987	619	594
24	1107	828	791	629
25	852	876	762	402
Mean	986.84	867.8	729.28	550.6
Standard Deviation	118.44	129.75	123.48	113.05

Table 2. Non-Destructive Search: The area covered by N=10 agents in four settings: Version 1 ($l = 1$) and Version 2 (with the same $\mu = 2$ and $l = 2$, $l = 3$ and $l = 4$). Each setting was run 25 simulations. The average and the standard deviation are presented.

```
$Flag\_i\_move$=1;
Randomly rotate the agent;
Compute $D_{ij}$ the distance between the agent and the rest of $N - 1$ ;
IF (neighborhood is empty) AND ($D_{ij} < R_2$)
THEN (MOVE 1 Step);
END IF;
END IF;
END DO;
```

Feedback from Referee:

```
Compute $f = \sum(Flag\_i\_move)$ for all $N$ agents;
Compute $G(f)$;
```

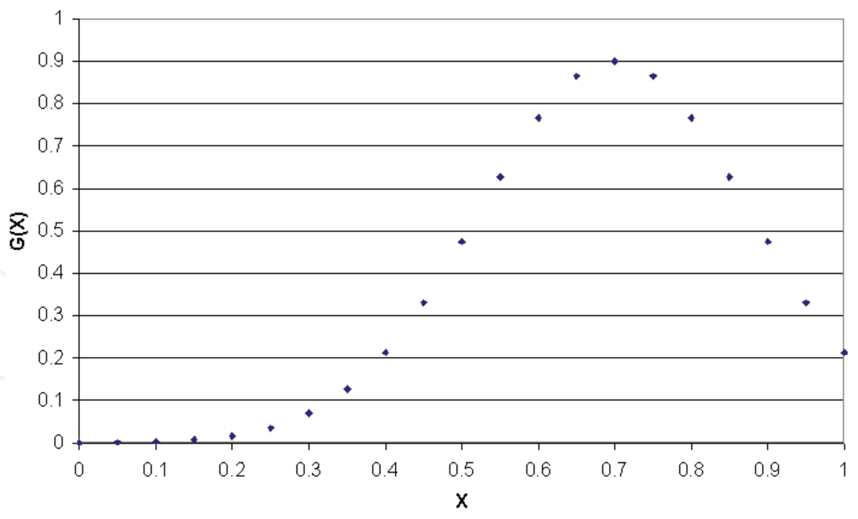


Fig. 2. The function $G(x) = 0.9 \times e^{-\frac{(0.7-x)^2}{0.0625}}$ has its maximum value for x=0.7.

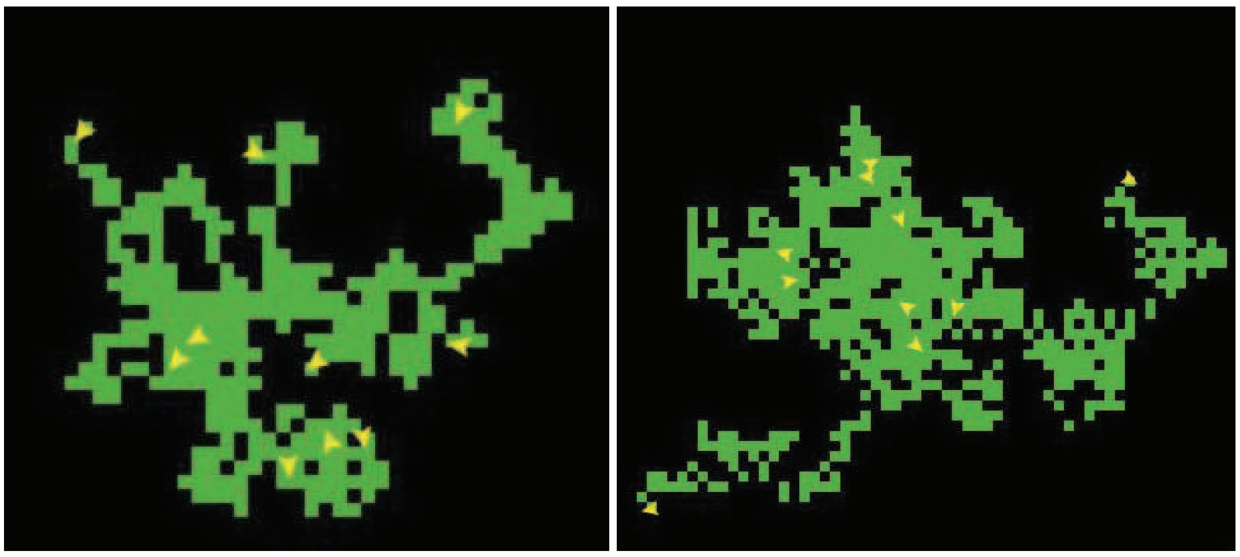


Fig. 3. Destructive Search: (Left)The simulation made with Version 1. (Right) The simulation made with Version 2: Levy flight with $\mu = 2$ and $l = 2$.

Updating probabilities for agents

```
FOR each agent DO:
Generate a random number $S_i$;
IF ($S_i < G(f)$) THEN:
IF ($flag\_i\_move$ = 1) THEN increase $P_{i1}$
ELSE decrease $P_{i1}$;
END IF;
END IF;
END DO;
Set $Flag\_i\_move$=0 for each agent;
```

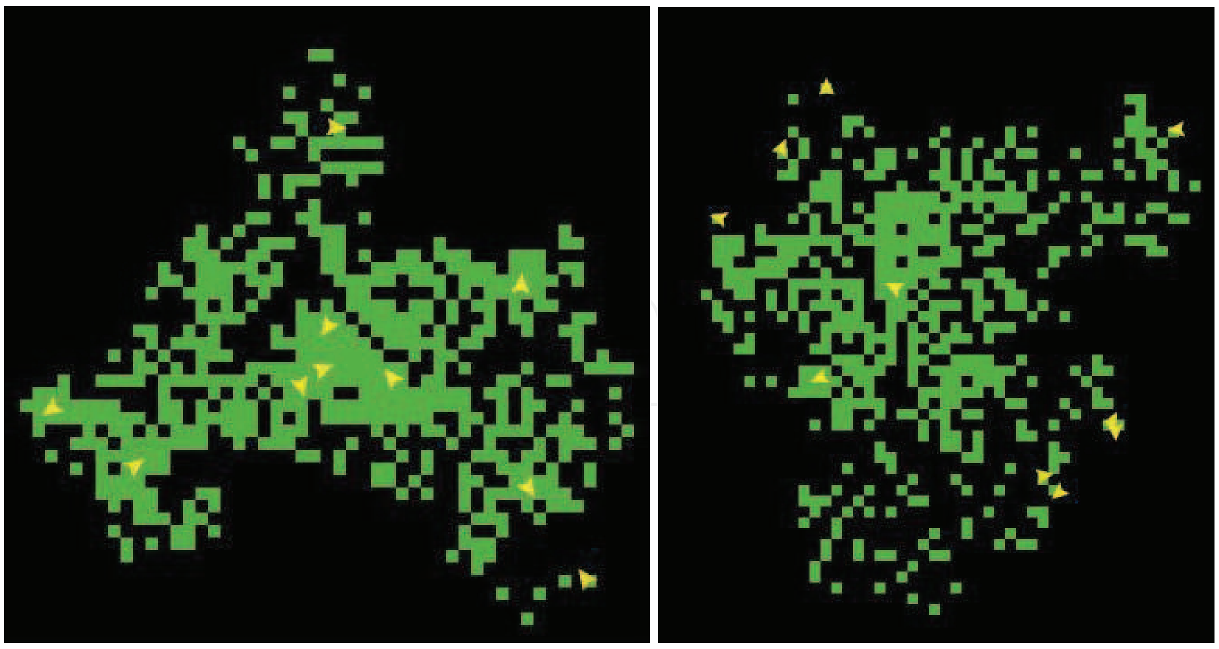


Fig. 4. Destructive Search: (Left) The simulation made with Version 2: Levy flight with $\mu = 2$ and $l = 3$. (Right) The simulation made with Version 2: Levy flight with $\mu = 2$ and $l = 4$.

END DO;

Remarks:

- As we mentioned earlier, our proposed algorithm can be applied on two versions, function of how the resources are distributed in the environment.
  1. Version 1: the movement step is equal to unity.
  2. Version 2: the movement step follows a Levy flight distribution described by $l \in [1;5]$ and $\mu = 2$).
- Again, a comment can be made regarding the order of actions to be taken. The steps are the same as in the previous section. However, the difference from the destructive search consist in the freedom to mode into an already visited empty space.

6. Simulation results

In this section, we present four families of simulations: one obtained with Version 1 of the algorithm and three obtained with different settings of Version 2 of the algorithm (namely three different values for the length of the movement step). We depict only the graphical results obtained after enough iterations that are not allowed more movements, for each version of the proposed algorithm. However, we present in Table 2 the area covered by N=10 agents (players) in four settings: Version 1 and Version 2 with the same $\mu = 2$ and $l = 2$, $l = 3$ and $l = 4$. Each setting was run 25 simulations. In each of these settings, we used $N = 10$ agents, with $R_1 = 10$ and $R_2 = 40$. All the agents are L_{RI} with $\theta = 0.1$. The performance criterion $G(\cdot)$ used by the Referee is $G(x) = 0.9 \times e^{-\frac{(0.7-x)^2}{0.0625}}$, as presented in the previous section. The reader can see that the maximum was covered from Version 1; the increase in the length of the movement decreases the total area covered by the agents.

In Figure 5 (left), we present the result corresponding to Version 1 of the algorithm, namely when the movement step is equal to unity. In Figures 5 (right) and 6 we present the results corresponding to Version 2 of the algorithm, namely when the movement step follows a Levy flight with $l = 2$ in the first graph, with $l = 3$ in the second graph, and $l = 4$ in the third graph. The exponent of the distribution is always $\mu = 2$. The reader can observe the size of the steps as being very close to the agents that cannot move. Many of them have a probability P_1 equal to unity. Thus, they can move from the point of view of the decision maker. However, the distance constraint will not allow them to realize the movement on the field.

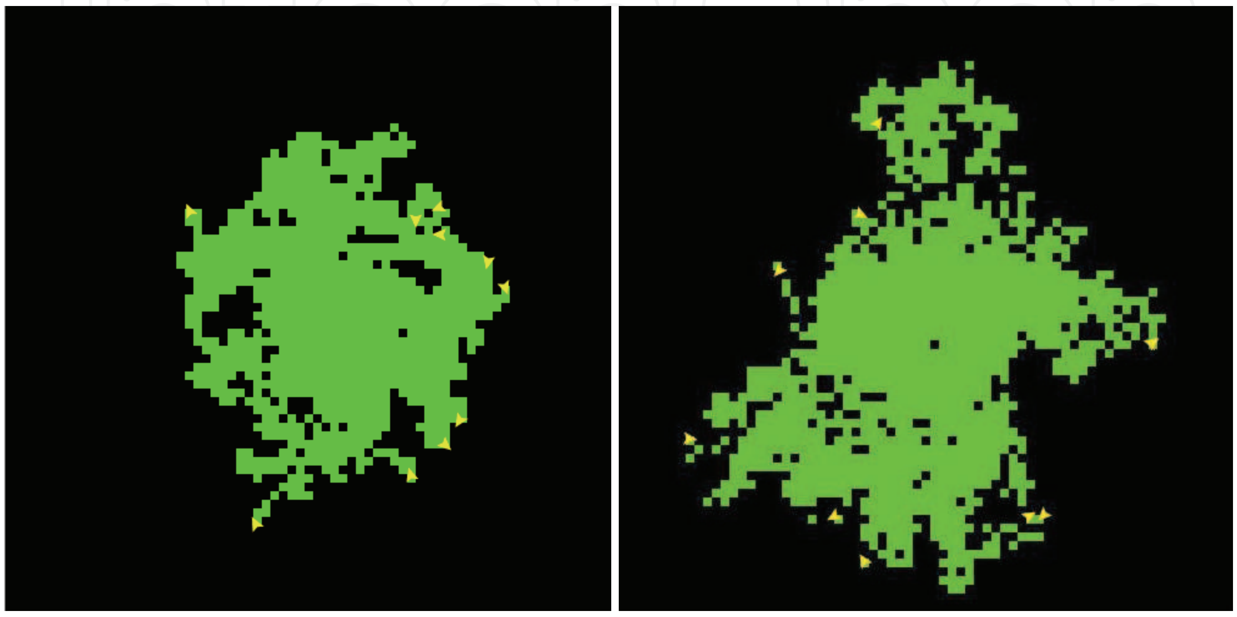


Fig. 5. Non-Destructive Search: (Left) The simulation made with Version 1. (Right) The simulation made with Version 2: Levy flight with $\mu = 2$ and $l = 2$.

7. Conclusions

This paper presents a new agents-based algorithms which can be used for identifying efficient search strategies for locating objects (e.g. mines) distributed over large areas. The originality of our work consists in applying a non-cooperative non-zero sum game, namely the distributed Goore Game model. As opposed to the classical collaborative and competitive strategies, the agents are not aware of the actions of the other agents.
We investigated two methods, namely the destructive search and the non-destructive search. The proposed algorithm has two versions: one when the agent can move with a step equal to unity and the other when the step of the agent follows a Levy flight distribution. The latter version is inspired by the work of Reynolds, motivated by biological examples. We present the area covered in each simulation function of the length of the movement step. The reader can evaluate the benefit of each version.
In the case of the non-destructive search, the reader can observe a trade off between the maximum of area covered and the freedom to search far from the initial center. The Version 1 of the algorithm covers the greatest area. However, the Version 2 of the algorithm is able to search after targets at a greater distance from the initial center.
In the case of the destructive search, the reader can observe that the Version 2 of the algorithm covers the greatest area. Increasing l will stabilize the variation of the searching area (the standard deviation is decreasing).

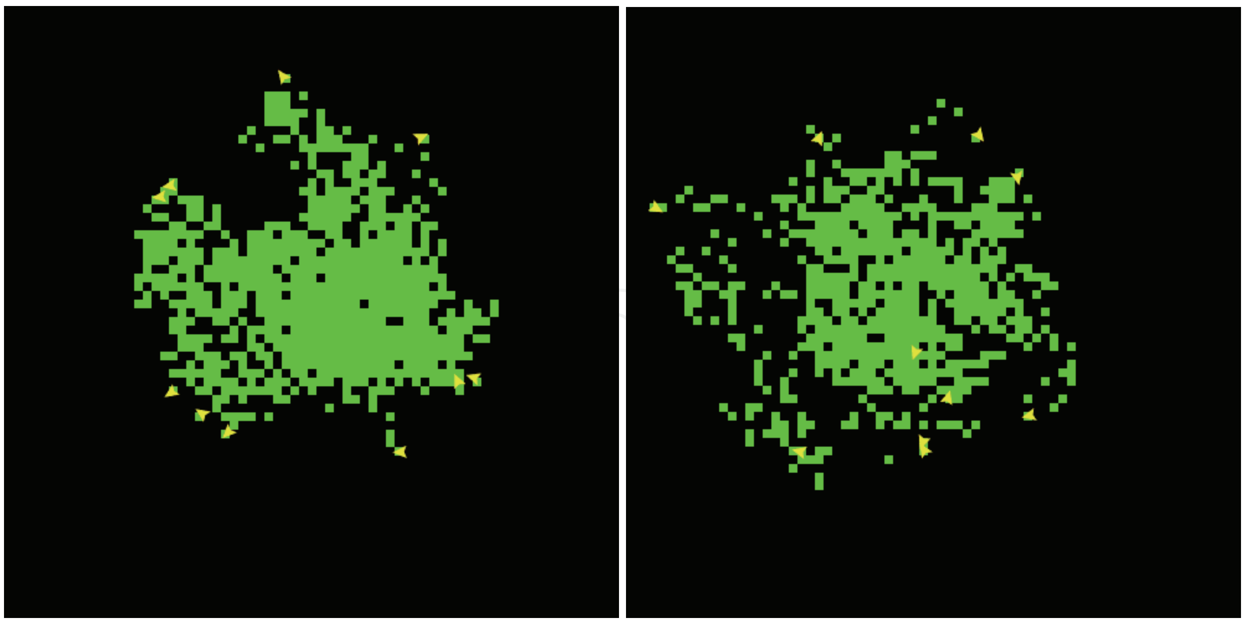


Fig. 6. Non-Destructive Search: (Left)The simulation made with Version 2: Levy flight with $\mu = 2$ and $l = 3$. (Right) The simulation made with Version 2: Levy flight with $\mu = 2$ and $l = 4$.

The future direction of this research are: i) producing a sensitivity analysis of the search problem, namely how the area covered by the agents depends on N, R_1, R_2 and G and ii) comparing the benefits of destructive vs. non-destructive search algorithms.

8. References

Benhamou, S. (2008). How many animals really do the Levy walk? reply, *Ecology* Vol. 89(No. 8): 2351–2352.

Cai, X., Zeng, J., Cui, Z. & Tan, Y. (2007). Particle Swarm Optimization using Levy probability distribution, *L.Kand, Y.Liu and S. Zeng (Eds): ISICA 2007, LNCS 4683* pp. 353–361.

Cao, Y., Fukunaga, A. & Kahng, A. (1997). Cooperative mobile robotics: Antecedents and directions, *Autonomous Robots* Vol. 4(No. 1): 1–23.

Chen, D. & Varshney, P. (2004). QoS support in wireless sensor networks: A survey, *The 2004 International Conference on Wireless Networks (ICWN 2004)* pp. 227–233.

Chignell, R. (1998). MINREC - a development platform for anti-personel mine detection and recognition, *Detection of abandoned land mines, Conference publication no. 458*, IEE, Edinburg, U.K., pp. 64–76.

Ho, S., Nassef, N., Pornsin-Sirirak, N., Tai, Y.-C. & Ho, C.-M. (2002). *Flight dynamics of small vehicles*, ICAS CONGRESS, Toronto, Canada.

Iyer, R. & Kleinrock, L. (2003). Qos control for sensor networks, *IEEE International Conference on Communications* Vol. 1: 517–521.

Kareiva, P. & Shigesda, N. (1983). Analysis insect movement as a correlated random walk, *Oecologia* Vol. 56: 234–238.

Lakshmivarahan, S. (1981). *Learning Algorithms Theory and Applications*, Springer-Verlag, Berlin.

Narendra, K. & Thathachar, M. (1989). *Learning Automata: An Introduction*, Prentice-Hall Inc., Upper Saddle River, NJ.

Norman, M. F. (1968). On linear models with two absorbing barriers, *Journal of Mathematical Psychology* Vol. 5: 225–241.

Okubo, A. & Levin, S. (2002). *Diffusion and ecological problems: modern perspectives*, Springer-Verlag, New-York.

Oommen, B., Granmo, O. & Pederson, A. (1999). Using stochastic AI techniques to achieve unbounded resolution in finite player Goore game and its applications, *Proceedings of IEEE-CIG'07, the 2007 IEEE Symposium on Computational Intelligence and Games*, IEEE, Hawaii, pp. 161–167.

Plice, L., Pisanich, G. & Young, L. (2003). Biologically inspired behavioural strategies for autonomous aerial explorers on Mars, *Proceedings of the 2003 IEEE Aerospace Conference, Big Sky* 1: 1–304.

Poznyak, A. & Najim, K. (1997). *Learning Automata and Stochastic Optimization*, Springer-Verlag, Berlin.

Reynolds, A. (2006a). On the intermittent behavior of foraging animals, *Europhysics Letters* Vol. 75(No. 4): 517–520.

Reynolds, A. (2006b). Optimal scale-free searching strategies for the location of moving targets: New insights on visual cued mate location behaviour in insects, *Physics Letters A* Vol. 360: 224–227.

Reynolds, A. (2007). Avoidance of specific odour trails results in scale-free movement patterns and the execution of an optimal searching strategy, *Europhysics Letters* Vol. 79: 30006–30011.

Reynolds, A. (2008a). How many animals really do the Levy walk? comment, *Ecology* Vol. 89(No. 8): 2347–2351.

Reynolds, A. (2008b). Optimal random Levy-loop searching: new insights into the searching behaviours of central-place foragers, *Europhysics Letters* Vol. 82(No. 2): 20001–20006.

Reynolds, A. (2009). Adaptive Levy walks can outperform composte Brownian walks in non-destrcutive random searching scenarios, *Physica A* Vol. 388: 561–564.

Reynolds, A. & Rhodes, C. (2009). The Levy flight paradigm: random search patterns and mechanisms, *Ecology* Vol. 90: 877–887.

Rhodes, C. & Reynolds, A. (2007). The influence of search strategies and homogeneous isotropic turbulence on planktonic contact rates, *Europhysics Letters* Vol. 80: 60003–60012.

Shlesinger, M. & Klafter, J. (1986). *Growth and Form, H.E.Stanley and N.Ostrowski (Eds)*, Martinus Nijhof Publishers, Amsterdam.

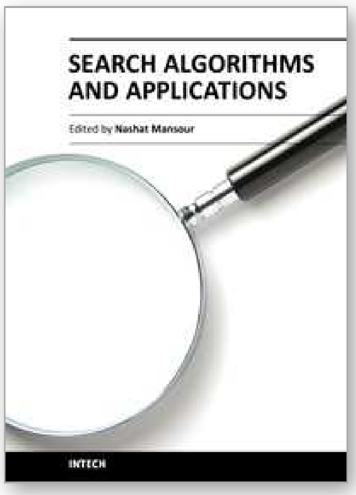
Thathachar, M. & Arvind, M. (1997). Solution of Goore game using models of stochastic learning automata, *Journal of Indian Institute of Science* (No. 76): 47–61.

Tsetlin, M. (1963). Finite automata and the modeling of the simplest forms of behavior, *Uspekhi Matem Nauk* Vol. 8: 1–26.

Tung, B. & Kleinrock, L. (1996). Using finite state automata to produce self-optimization and self-control, *IEEE Transactions on parallel and distributed systems* Vol. 7(No. 4).

Viswanathan, G., Afanasyev, V., Buldyrev, S., Murphy, E., Prince, P. & Stanley, H. E. (1996). Levy flight search patterns of wandering albatrosses, *Nature* Vol. 381: 413–415.

Viswanathan, G., Buldyrev, S., Havlin, S., da Luz, M. G., Raposo, E. P. & Stanley, H. E. (1999). Optimizing the success of random searches, *Nature* Vol. 401: 911–914.



Search Algorithms and Applications

Edited by Prof. Nashat Mansour

ISBN 978-953-307-156-5

Hard cover, 494 pages

Publisher InTech

Published online 26, April, 2011

Published in print edition April, 2011

Search algorithms aim to find solutions or objects with specified properties and constraints in a large solution search space or among a collection of objects. A solution can be a set of value assignments to variables that will satisfy the constraints or a sub-structure of a given discrete structure. In addition, there are search algorithms, mostly probabilistic, that are designed for the prospective quantum computer. This book demonstrates the wide applicability of search algorithms for the purpose of developing useful and practical solutions to problems that arise in a variety of problem domains. Although it is targeted to a wide group of readers: researchers, graduate students, and practitioners, it does not offer an exhaustive coverage of search algorithms and applications. The chapters are organized into three parts: Population-based and quantum search algorithms, Search algorithms for image and video processing, and Search algorithms for engineering applications.

How to reference

In order to correctly reference this scholarly work, feel free to copy and paste the following:

Dragos Calitoiu and Dan Milici (2011). Modeling with Non-cooperative Agents: Destructive and Non-Destructive Search Algorithms for Randomly Located Objects, Search Algorithms and Applications, Prof. Nashat Mansour (Ed.), ISBN: 978-953-307-156-5, InTech, Available from: http://www.intechopen.com/books/search-algorithms-and-applications/modeling-with-non-cooperative-agents-destructive-and-non-destructive-search-algorithms-for-randomly-

INTECH
open science | open minds

InTech Europe
University Campus STeP Ri
Slavka Krautzeka 83/A
51000 Rijeka, Croatia
Phone: +385 (51) 770 447
Fax: +385 (51) 686 166
www.intechopen.com

InTech China
Unit 405, Office Block, Hotel Equatorial Shanghai
No.65, Yan An Road (West), Shanghai, 200040, China
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元
Phone: +86-21-62489820
Fax: +86-21-62489821

© 2011 The Author(s). Licensee IntechOpen. This chapter is distributed under the terms of the Creative Commons Attribution-NonCommercial-ShareAlike-3.0 License, which permits use, distribution and reproduction for non-commercial purposes, provided the original is properly cited and derivative works building on this content are distributed under the same license.

IntechOpen

IntechOpen