# DECISION SUPPORT FOR CROWD CONTROL: USING GENETIC ALGORITHMS WITH SIMULATION TO LEARN CONTROL STRATEGIES

Johan Schubert and Robert Suzić
Department of Decision Support Systems
Division of Command and Control Systems
Swedish Defence Research Agency
SE-164 90 Stockholm, Sweden
schubert@foi.se
http://www.foi.se/fusion

## ABSTRACT

*In this paper we describe the development of a decision support system for crowd control. Decision support is provided by suggesting a control strategy needed to control a specific current riot situation. Such control strategies consists of deployment of several police barriers with specific barrier positions and barrier strengths needed to control the riot. The optimal control strategy for the current situation is found by comparing the current situation with pre-stored example situations of different sizes. The control strategies are derived for these pre-stored example situations by using genetic algorithms where successive trial strategies are evaluated using stochastic agent-based simulation.*

Keywords: Crowd control, riot control, decision support, learning, simulation, fuzzy measure, genetic algorithms.

## INTRODUCTION

In international operations today, our forces must be able to handle riots and be successful in crowd control. This makes it important to derive control strategies for different riot situations [1]. Such crowd control strategies may be evaluated using a riot simulator and used to provide decision support when found effective.

In this paper we develop a decision support system for crowd control. Decision support is provided by suggesting a control strategy needed to control a specific current riot situation. Such control strategies consists of deployment of several police barriers with specific barrier positions and barrier strengths needed to control the riot. The tactical commanders, responsible for keeping security, may use those control strategies derived for simulated situations that are most similar to the on-going situation, and study the predictive situation picture given alternative courses of actions.

The control strategies are derived for these pre-stored example situations by using genetic algorithms [2] where successive trial strategies are evaluated using stochastic agent-based simulation. Each strategy corresponds to the collected information about the strength at every possible predetermined barrier position. The population of all individuals in the genetic algorithm makes up the collected set of all alternative strategies for positioning and manning the barriers.

The stochastic agent-based simulation is based on stochastic agent models, embedded simulations and its predicted effects. We have developed a proof-of-concept model that is aimed to exemplify and give new research findings of embedded simulations [3] [4] [5]. Inputs are strategies and positions of real world agents with uncertain estimates of hostility. Effects that may occur are destroyed buildings of importance and destroyed barriers. The strategies are scored based on these effects. The simulations are used to generate a data base over simulated riot cases in a certain city environment. This is done beforehand for a specific city or suburb using its street and block network, before an assumed critical situation might arise.

The problems we have to solve to realize the sought-after decision support are:

- a decision theoretic matching algorithm comparing a current situation with pre-simulated situations,
- an appropriate computer representation of control strategies for the genetic algorithm,
- an effective agent simulation model,

in *Proc. Third IEEE Workshop Situation Management* (SIMA 2007), Paper SIMA-3.4, pp. 1-7, Orlando, USA, October 2007.

- the level of simulation complexity.

A novelty in our approach is that we combine agent based simulation and genetic learning to generate optimal ways to control a crowd.

In the last several years there has been a substantial amount of research on how to model crowds and give them complex behaviors using intelligent agents. In spite of all this research there is very little work on the subject of crowd control. Several scientists involved in Project Albert [6] (US Marine Corps) have also used genetic algorithms for optimization of agent-based behavior. In a project report Graves *et al*. [7] demonstrated how to use genetic algorithms to improve the rule bases that define when individual agents take various actions in agent-based simulation, e.g., when advancing towards enemy lines or shooting at enemy soldiers. Dixon and Reynolds [8] used genetic algorithms for peacekeeping scenarios on their Behavior Action Simulation Platform (BASP) for cases where it is anticipated that the model will evolve from one or more preceding models.

Previously, we have used genetic algorithms to derive prediction rules within anti-submarine intelligence analysis [9]. While that simulation was done in a much simpler way without any interaction with own forces it also modeled situation awareness towards the opposing forces. This is not considered here. Presumably, we could do both to improve on realism in decision support.

In the next section we describe how decision support is designed and realized using crowd control strategies. This is followed by a section on learning the strategies for crowd control. The fourth section presents how stochastic agent-based simulation is used in the learning process. Finally, conclusions are drawn.

## DECISION SUPPORT

In this section we describe how decision support is designed using the results from a learning and simulation system.

Decision support is provided to the tactical commander as the optimal control strategy for the current situation. It is found by comparing the current situation with pre-stored example situations of different sizes, Figure 1. With each of these pre-stored situations an optimal control strategy is associated. The current riot situation may have a best match to a superposition of a subset of these pre-stored situations. The decision support will then be given as the corresponding superposition of control strategies.

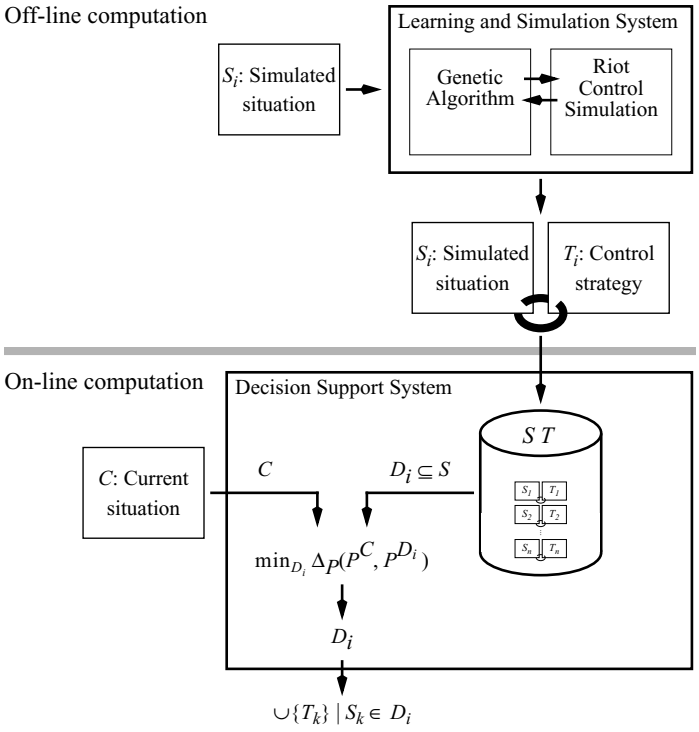


Figure 1. On-line decision support is given as $\cup\{T_k\}$ where $\{T_k\}$ is a subset of all control strategies such that the corresponding set of simulations $D_i = \{S_k\}$ minimizes the distance $\Delta_P$ to the current situation $C$.

Let us investigate how the optimal control strategy may be constructed. Let $S = \{S_i\}$ be the set of all simulations $S_i$. We have $S_i = \{\mu_j\}$, where $\mu_j$ is the starting position[1] of the $j$th agent of $S_i$.

For a particular simulation $S_i$ we calculate a 2-dimensional fuzzified position map $P^{S_i}$ around each agent starting position $\mu_j$ using a normal distribution

$$P^{S_i}_{\mu_j}(x_k) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x_k-\mu_j)^2}{2\sigma^2}}, \qquad (1)$$

where $P^{S_i}_{\mu_j}(x_k)$ is the degree to which the $j^{\text{th}}$ agent's starting position is $x_k$, $x_k - \mu_j$ is the euclidean distance between the two positions and $\sigma^2$ is the variance of the distribution. The variance used is domain dependent and must be adjusted in relation to the used resolution of the position map, Figure 2.

[1]It should be noted that we treat each position of a simulation run as alternative starting positions for different simulations labeled $S_i$.

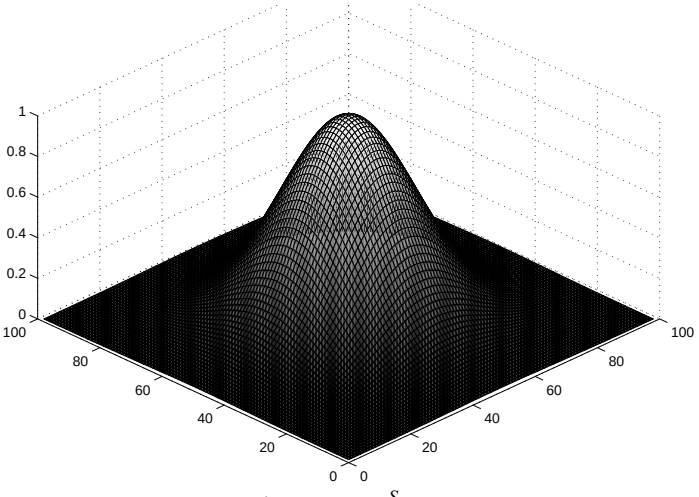

Figure 2. $P^{S_i}_{\mu_j}(x_k)$.

For each position $x_k$ in the map we sum up the contribution from each agent's starting position $\mu_j$

$$P^{S_i}(x_k) = \sum_{\mu_j \in S_i} P^{S_i}_{\mu_j}(x_k), \tag{2}$$

Figure 3.

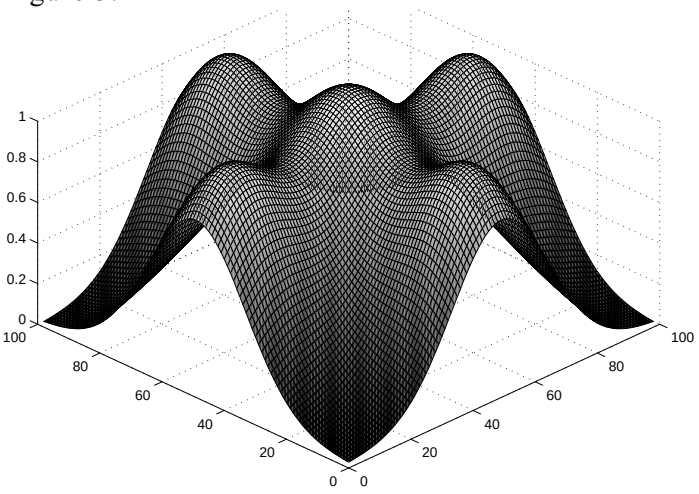

Figure 3. $P^{S_i}(x_k)$.

Using the position maps

$$\left\{ P^{S_i}(x_k) \right\}_k, \tag{3}$$

we may calculate the total support $P^{D_j}(x_k)$ for each position $x_k$ in the map from any subset of all simulations $D_j \subseteq S$, where $\{D_j\}_j = 2^S$.

We have

$$P^{D_j}(x_k) = \sum_{S_i \in D_j} P^{S_i}(x_k). \tag{4}$$

Here,

$$\left\{ P^{D_j}(x_k) \right\}_k \tag{5}$$

is the support map for $D_j$ and

$$\left\{ \left\{ P^{D_j}(x_k) \right\}_k \right\}_j \tag{6}$$

is the set of all alternative superpositioned support maps.

To each simulation $S_i$ we have a strategy $T_i$ attached. In order to find the best decision support we compare the current situation $C$ with all subsets of all simulations $D_j \subseteq S$.

First, we calculate a 2-dimensional fuzzified position map $P^C$ around each agent starting position $\mu_j$ in the current situation $C$, using a normal distribution in the same way as was done in Eq. (1),

$$P^C_{\mu_j}(x_k) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x_k - \mu_j)^2}{2\sigma^2}}. \tag{7}$$

Here, $P^C_{\mu_j}(x_k)$ is the degree to which the $j^{\text{th}}$ agent's starting position in the current situation $C$ is $x_k$.

As in Eq. (2) we sum up the contribution from each agent's starting position $\mu_j$ for each position $x_k$ in the map

$$P^C(x_k) = \sum_{\mu_j \in C} P^C_{\mu_j}(x_k). \tag{8}$$

Using $P^C$ we can now find the best decision support. We calculate the sum of absolute differences between $P^C$ and $P^{D_j}$ for all positions $x_k$ in the map

$$\Delta_P(P^C, P^{D_j}) = \sum_{x_k} \left| P^C(x_k) - P^{D_j}(x_k) \right| \tag{9}$$

for all $D_j \subseteq S$ including $D_j = \varnothing$, where

$$P^{\varnothing}(x_k) \equiv 0 \qquad \forall x_k, \tag{10}$$

Figure 4.

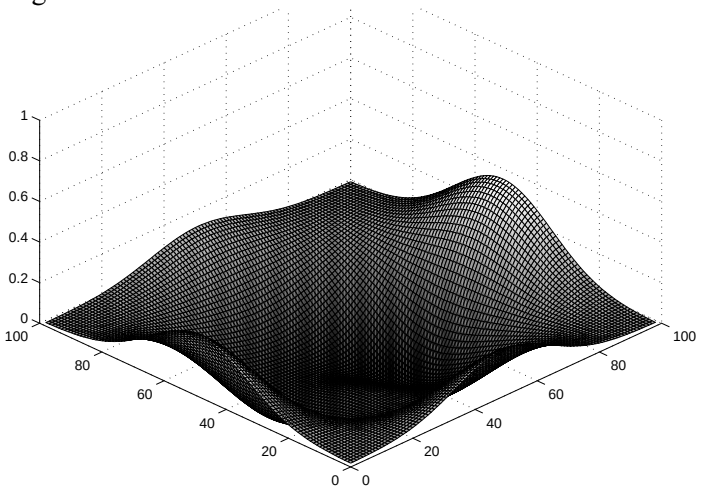

Figure 4. $P^C(x_k) - P^{D_j}(x_k)$.

Decision support is given as the union of all strategies $\cup\{T_k\}$ whose corresponding subset of simulations $S_k \in D_i$ minimizes the difference between the fuzzified position-map of $C$ and $D_i$, i.e.,

$$\cup\{T_k\} \Big| \begin{array}{l} S_k \in D_i, S_l \notin D_i, l \neq k \\ \forall D_j \subseteq S, j \neq i.\ \Delta_P(P^C, P^{D_i}) < \Delta_P(P^C, P^{D_j}) \end{array} \quad (11)$$

Whenever several control strategies in $\{T_k\}$ have a strength greater than zero for some barrier position their strengths are simply summed up as the decision support regarding that particular barrier.

## LEARNING STRATEGIES

A genetic algorithm is a method for solving optimization problems that is based on natural selection, the process that drives evolution. Starting from a population of random individual problem solutions, Figure 5, the genetic algorithm evolves the population by repeated incremental modifications. At each step the genetic algorithm selects at random two individuals from the population as parents and uses them to create an offspring for the next generation. Successively over time the population evolves towards an optimal solution to the problem at hand. This biologically inspired process is a very robust optimization tool suitable for many difficult optimization problems.

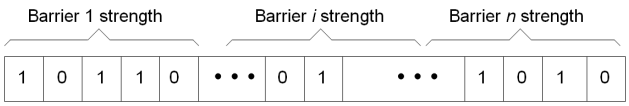


Figure 5. A genetic binary representation (chromosome) of a complete strategy for all barrier strengths.

In the problem of optimizing road barrier strengths and positions each individual corresponds to the collected information about the strength at every possible predetermined barrier position. A strength of zero at a certain barrier position corresponds to no barrier at this position. The higher the strength, the more agents a barrier can manage before it is broken and more resources are needed to man the barrier. The sum of all barrier strengths is constrained in the optimization. This corresponds in a real riot case to the situation that the resources for manning barriers are limited.

The population of all individuals makes up the collected set of all alternative strategies for positioning and manning the barriers. The optimization of strategies for barrier positions and strengths is carried out in each generation by evaluating each strategy by itself. Each strategy is scored based on its success in the riot control simulation from two aspects, on the one hand in protecting certain designated buildings and on the other hand that the barriers themselves are not disrupted by the rioters.

An obvious balance is that the barriers must be placed in positions where they are at risk themselves in order to protect the designated buildings. However, they must not be placed in such a way that they are at risk needlessly nor be given such a low strength that they are easily broken without giving protection to the designated buildings. A suboptimal solution will be achieved if proper consideration is not taken to the fact that both barriers and designated buildings may become destroyed by the rioters in the simulation.

The initial set of strategies are generated randomly. As this set of alternative strategies develops, good strategies with high scores are chosen and generate offspring. This is done in such a way that all strategies in one generation are evaluated by the riot control simulator and awarded score points from the perspective of how well the designated buildings were spared and that the barriers themselves where not disrupted by the rioters. From this scoring all strategies are ranked. The strategies are now given new points where the best strategy is awarded one point, the second best ½ point, the third best ⅓ point, etc. After normalization by the sum of these new scores they yield the probability with which two parents are selected. This is called Rank Selection. Thus, two individuals are selected randomly from the population with these probabilities. This selection of pairs of individuals is repeated as many times as there are individuals in the population. If the population consists of 100 individuals, 100 pairs of individuals are selected. Each individual can be selected several times. Some individuals that represent strategies that have been successful will certainly be selected several times, while other individuals that represent strategies that fail may have such a low probability that they are not selected at all. Each pair generates one offspring to the next generation which will consist of the same number of individuals as the last. In this manner the optimization will continue generation by generation

towards better and better results until a good enough solution is obtained.

Finally, generation of a new strategy (offspring) is created from a selected pair of alternative strategies (parents) by selecting some barrier positions and strengths from one parent and some from the other. This is done randomly. It is also possible to have a new barrier strength by selecting bits and pieces from each parent's barrier representation. This will give us a new strategy that is based on the two parent strategies. To select parents randomly using a probability based on their performance in simulation, but not directly selecting only the best, leads to an optimization that is very robust.

## STOCHASTIC AGENT-BASED SIMULATION

In this section we describe the simulation that is used to score strategies in the genetic algorithm.

Stochastic agent-based simulation (SABS) is based on stochastic agent models, embedded simulations and the predicted effects that are the output of the simulation. In Figure 6 we describe some of the features of SABS. An agent is anything that can perceive using its sensors and act using its effectors. An agent can be an individual or a group of individuals [10]. In our simulation, a software agent is a representation of a real world agent.

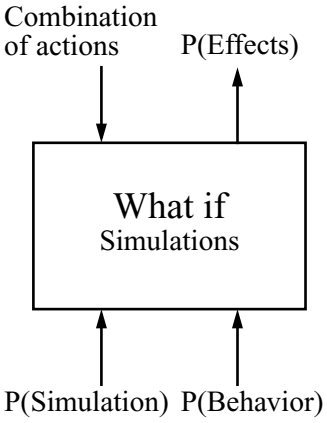


Figure 6. Stochastic agent-based simulation testing different actions.

Inputs to the simulation are strategies, uncertainty based representation of situation and behavior. Strategies are represented here by all barrier positions and strengths. Situations are represented by positions of real world agents, both rioters and peaceful protesters, with an uncertainty-based estimate of their hostility. By specifying the number in a range from zero to one the user may enter its own belief of the median value of the statistical distribution of hostility. Simulation runs until a pre-defined stop time, determining how long the situation is predicted.

Scenarios have to be developed for each area of interest. They are not assumed to be generic in nature. Given a specific scenario, barrier strategies and protesters actions, embedded simulation can be used, Figure 7. Since the simulations use stochastic agents, different effects may occur from one simulation to the next. Simulated effects are destroyed buildings of importance and destruction of the barriers. The prioritization between buildings and barriers is presumed to be made by a tactical commander. In learning strategies by genetic algorithms and simulation we put a weight on how a destroyed building is prioritized compared to a disrupted barrier.

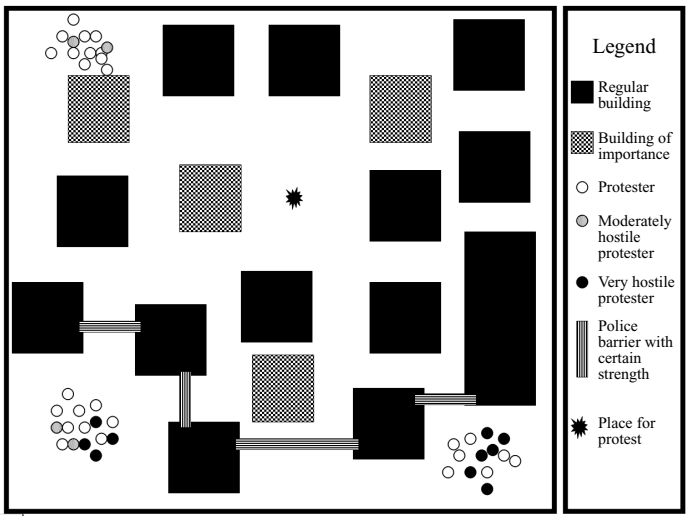


Figure 7. Visualization of a simulation.

The agent behavior is modeled for both rioters and peaceful demonstrators. The model consists of three parts.

The first part defines crowd dynamics. The research of Helbing [11] has been a guideline for describing how a group of agents moves through an environment and interact with each other. According to Helbing, the motion of pedestrians can be described as if they were subject to social forces. The social force is a sum of inertial ($\bar{F}_i$), repelling ($\bar{F}_r$) and attracting forces ($\bar{F}_a$) [11], [12], i.e.,

$$\bar{F}_{total} = \bar{F}_i + \bar{F}_r + \bar{F}_a . \quad (12)$$

Inertial force means that agents continue to move in the present direction. This is not modeled in our SABS implementation. Since the agents we model are lacking mass we use velocity vector addition, resulting in the following motion model

$$\bar{v}_{resulting} = \bar{v}_{attraction} + \bar{v}_{repelling} . \quad (13)$$

The length of the repelling velocity vector of an agent is modeled as an exponential function in analogy to the repelling force described in [11], i.e.,

$$|\bar{v}_{repelling}| = c \cdot e^{(a_p - a_{closest})}, \quad (14)$$

where $a_p$ is the agent's position, $a_{closest}$ is the position of the closest other agent and $c$ is a constant.

The value of the attraction velocity vector $|\bar{v}_{attraction}|$ is dependent on distances to barriers and other places of interest. Those places include meeting points, important buildings and places where police or military forces are located. The direction of the attraction velocity vector depends on where important buildings are and what is the A* -optimal [10] path to reach them. Helbing's computer simulations of interacting pedestrians show that the social force model is capable of describing the self-organization of several observed collective effects of pedestrian behavior very realistically. In other words, Helbing's results have been submitted to verification and were shown to have high accuracy in crowd modeling [11].

The second part of the behavior model concerns the interaction between different agents and objects. It is modeled mainly in a heuristic manner and additional study is needed. The interaction between rioters and our own forces is a matter of force balance, i.e., the strength of our own forces vs. that of rioters. If our own forces are stronger than the rioters then barriers will hold. A barrier event is modeled as follows:

IF $w_{normal} \cdot a_n + w_{halfangry} \cdot a_{ha} + w_h \cdot a_h > w_{strength} \cdot a_{own}$

THEN event: BarrierBroken

(15)

The third part is statistical sampling. It is limited to sampling one property of each agent. In every simulation step, agents' hostilities are sampled from an initial distribution of our estimate about the emotional state of demonstrators. If agents during the simulation face situations where congestion occurs, then we will see an increase in the level of hostility. This leads to different behaviors depending on the level of hostility, and as a result to different effects. These predicted effects are used by the genetic algorithm to evaluate barrier placements and their corresponding strengths.

## RESULTS

We ran a scenario with 21 different possible placements of barriers. In these scenarios barriers have a strength from zero (meaning there is no barrier) to $2^5$ (i.e., the strongest possible barrier given total amount of resources). In the scenario we involve several key buildings that the tactical commander has the task to protect. In the tests we use several groups of agents, each representing different groups of protesters with variable level of hostility. A brute fore approach would be to run $2.1 \cdot 10^{(2^5)}$ simulations to find the optimal strategy according to the simulation model. However, we use a genetic algorithm that leads the simulation to an optimum after only 1000 simulations (10 generations), Figure 8.

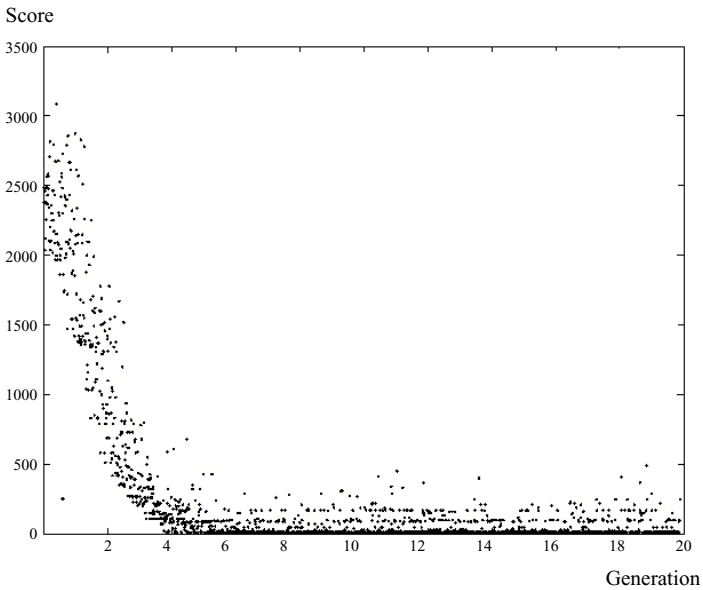


Figure 8. Convergence of GA by using simulations.

In the scenario with 21 different possible placements of barriers we were able to derive optimal and near optimal control strategies.

## CONCLUSIONS

We have demonstrated that it is possible to derive riot control strategies using genetic algorithms and stochastic agent-based simulation. Furthermore, we have developed a decision making algorithm where a current situation is compared to all simulated situations. The union of control strategies whose corresponding superposition of simulated situations most closely resembles the current situation, is given as decision support.

## FUTUTRE WORK

At this stage the genetic algorithm for learning strategies and the stochastic agent-based simulation are implemented. The next step in our work will be an implementation of the full decision support system using the strategies, the current situation and Eq. (11) for selecting riot control strategies as decision support.

## REFERENCES


[1] D. Grieger. An Overview of Crowd Control Theory and Considerations for the Employment of Non-Lethal Weapons. DSTO-GD-0373, Defence Science and Technology Organisation, Edinburgh, SA, Australia, 2003.

[2] J. H. Holland (1973). Genetic Algorithms and the Optimal Allocation of Trials. *SIAM Journal on Computing* 2(2):88–105.

[3] R. Suzić and K. Wallenius. Effects Based Decision Support for Riot Control: Employing Influence Diagrams and Embedded Simulation. In *Proceedings of the Workshop on Situation Management* (SIMA 2005), Atlantic City, NJ, USA, 17 October 2005. IEEE, Piscataway, NJ, 2005.

[4] R. Suzić. A generic model of plan recognition using embedded simulations, microeconomics and behavior models. In *Proceedings of the 15th Conference on Behavior Representation in Modeling and Simulation* (BRIMS 2006), Baltimore, MD, USA, 15–18 May 2006. SISO, Orlando, FL, 2006.

[5] R. Suzić. Stochastic Multi-Agent Plan Recognition, Knowledge Representation and Simulations for Efficient Decision Making. PhD Thesis, ISBN 91-7178-498-5, TRITA-CSC-A 2006 : 21, ISSN-1653-5723, ISRN-KTH/CSC/A--06/21--SE. Royal Institute of Technology, Stockholm, 2006.

[6] Project Albert (2007, August). [Online]. Available: http://www.projectalbert.org

[7] T. Graves, R. Picard, and S. Upton. Improving Rule Bases for Agent Based Simulations. Unclassified Report 00-2566. Los Alamos National Laboratory, Los Alamos, NM, 2000.

[8] D. S. Dixon and W. N. Reynolds. The BASP Agent-Based Modeling Framework: Applications, Scenarios and Lessons Learned. In *Proceedings of the 36th Hawaii International Conference on System Sciences*, Track 3, Vol. 3, p. 93.3, Waikoloa, Hawaii, USA, 6–9 January 2003. IEEE Computer Society, Washington, DC, 2003.

[9] U. Bergsten, J. Schubert and P. Svensson. Applying Data Mining and Machine Learning Techniques to Submarine Intelligence Analysis. In *Proceedings of the Third International Conference on Knowledge Discovery and Data Mining* (KDD'97), pp. 127–130, Newport Beach, CA, USA, 14–17 August 1997. The AAAI Press, Menlo Park, CA, 1997.

[10] S. J. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach*. Prentice-Hall, Englewood Cliffs, NJ, 1994.

[11] D. Helbing and P. Molnár (1995). Social force model for pedestrian dynamics. *Physical Review E* 51(5):4282–4286.

[12] D. Helbing, I. Farkas, and T. Vicsek (2000). Simulating dynamical features of escape panic. *Nature* 407(6803):487–490.