The 2010 IEEE/RSJ International Conference on
Intelligent Robots and Systems
October 18-22, 2010, Taipei, Taiwan

A Decentralized Multi-Robot System for Intruder Detection in Security Defense

Yuyang Zhang and Yan Meng

***Abstract*—In security defense tasks, multiple robots need work cooperatively to detect offensive intrusion to protect some sensitive areas. In this paper, we propose a distributed algorithm for a multi-robot system with some static sensors. The system concept is that static sensors sense intrusions and act as a cueing sensor to an ensemble of robots. These robots in turn engage the potential intruder, performing surveillance and/or neutralization of the intrusion. To minimize the intruder missing rate and average response time, a STAGS (Shame-level Task Allocation and Gap-based Self-deployment) method is proposed, which is a decentralized method without a central control unit. To further improve the system adaptability under dynamic environments, a multi-objective optimization (MOO) method is proposed to adjust the system parameters of STAGS. Extensive simulation results demonstrate the effectiveness and robustness of the proposed algorithm in a dynamic intruder detection task.**

***Index Terms*—multi sensor/multi robot system, artificial immune systems, intruder detection, and perimeter defense.**

I. Introduction

SECURITY defense task is a complex problem, which aims to protect sensitive areas against offensive intrusion. Video surveillance system is one of the solutions for these tasks, which still require manned observation and can be quite costly for large areas. Another alternative solution is to use autonomous multi-robot systems (MRSs) for intruder detection to reduce the overall system cost without compromising security.

In this paper, we will describe an autonomous system consisting of cooperative mobile robots with some static sensors for security defense tasks. The system would utilize many relatively cheap sensors that can be used as a cueing sensor for an ensemble of robots to detect and track the movements of intrusion of any kind through a predetermined area or boundary. Through the use of mobile robots, the intruders can be tracked, intercepted, or neutralized. While some robots are investigating the intruders, the remaining robots would self-deploy themselves to maximize coverage. Fig.1. illustrates our simulator for this problem.

Extensive work has been proposed for multi-robot coordination for various applications, one paradigm is based on organization theory derived from human social behavior and psychology [2][6][12][20]. Another paradigm is bio-inspired algorithms [7][11][18]. Some research proposed artificial immune systems for demining problems [16] and exploration problems [19].

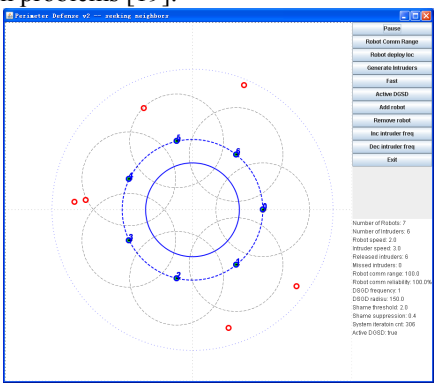

Fig. 1. A snapshot of the security defense problem simulator. The area to be protected is the blue solid circle. Seven robots are deployed on outer blue dotted circle (deployment circle). The communication range of each robot is represented by grey dotted circle. The red dots are intruders and blue dots are robots.

Some work has been directly addressed for security defense problems [1][9][13][14][17]. Agmon et al. [1] studied a multi-robot system for perimeter patrolling, where they proposed a non-deterministic scheme for robots patrolling, which is an optimal polynomial-time algorithm of finding the probability p to maximize the probability of penetration detection throughout the perimeter. Vidal et al. [17] proposed probabilistic pursuit-evasion games with unmanned ground and aerial vehicles. Machado [14] proposed a distributed MRS approach for patrolling in a complex environment based on a market economy approach. In this paper, we propose a STAGS (Shame-level Task Allocation and Gap-based Self-deployment) approach, which consists of a distributed shame-level based dynamic task allocation algorithm for intruder tracking and investigation, and a distributed gap-based self-deployment (DGSD) algorithm for self-deployment. Robots have to choose their own behaviors dynamically based on their current states and the environment. The parameters in STAGS approach need to be defined. To further improve the system robustness and adaptability to various environmental changes, a multi-objective optimization (MOO) method is applied to dynamically tune the parameters of the STAGS approach with two objectives: minimization of both the missing rate and the average response time.

The rest of the paper is organized as follows. The STAGS approach is proposed in Section II. In Section III, a multi-objective optimization (MOO) method is applied to dynamically tune the parameters of the STAGS approach.

Y. Zhang and Y. Meng are with Department of Electrical and Computer Engineering, Stevens Institute of Technology, Hoboken, NJ 07030, USA (phone: 201-216-5496; fax: 201-216-8246; e-mail: yzhang14@stevens.edu, yan.meng@stevens.edu)

Extensive experiments have been conducted to evaluate the proposed method in Section IV. Conclusions and future works are described in Section V.

II. The Decentralized STAGS Approach

A. Problem Statement

The objective of this paper is to protect a sensitive area from the intruders in security defense application using a multi-robot/multi-sensor system, as shown in Fig. 1. The environment is an 800x800 square area. The protected area is defined as a circle with the diameter of 200. It is assumed that new intruders appear in a Poisson distribution pattern as:

$$P(X=k)=\frac{\lambda^k e^{-\lambda}}{k!}(k=0,1,2,....) \quad (1)$$

where $P(X=k)$ is the probability that k new intruders arrive in each simulator iteration (the simulator's basic time unit). The expectation of $P(X)$ is λ, so on average a new intruder appears on every $1/\lambda$ system iteration. The intruders move directly toward the protected area in a straight line with some predefined velocity. In order to detect intruders, static fiber optic sensors are deployed uniformly in a bigger "detecting area" that encloses the protected area (the outer circle in Fig. 1.) with the sensing range of 100. It is assumed that sensors can detect the positions of the intruders, and can inform the intruders' positions to the robots which are within the sensor's communication range. Initially, multiple robots are deployed uniformly around the protected area. When a robot receives the position information of the intruders from the sensors, the robot will communicate with its neighboring robots to decide who will track and investigate the intruders. The selected robot will move toward the intruder directly in a straight line with some predefined speed. Of course, the robot can avoid obstacles on its way to the target intruders. The remaining robots will redeploy themselves to try to cover the protected area accordingly to maximize the coverage area.
In other words, robots switch between two roles dynamically: tracking robot and deploying robot.

The STAGS approach consists of two parts: the first one is a shame-level based algorithm for dynamic task allocation for tracking robots, and the second one is the gap-based algorithm for self-deployment. The goal of the STAGES approach is to coordinate robots to minimize the missing rate and average response time to the intruders. Missing rate is the percentage of intruders which successfully invade the protected area without being investigated by robots over all the intruders. Response time is the time period from the time of an intruder is detected by sensors to the time it is investigated by robots.

B. A Shame-Level based Dynamic Task Allocation Algorithm

Inspired by [10], a shame-level based algorithm is proposed to dynamically allocate robots to detected intruders. Each robot develops a shame level for each detected intruder, which is inversely proportional to its distance to the intruder. The shame level is incremented until it reaches a threshold that causes the robot to respond. Here, the shame level represents how "shame" a robot can feel of not responding to the detected intruder. The greater a shame level of a robot to a detected intruder, the higher possibility that the robot would respond to this detected intruder, which is kind of similar to the motivation level in ALLIANCE [15]. Once a robot starts to respond to an intruder, the robot would broadcast its decision to its neighboring robots so that the neighboring robots would suppress their shame levels to this intruder. In essence, other robots no longer "feel" the shame of not responding to the intruder, so that they can investigate other intruders or self-deploy themselves. Based on this idea, the shame level of a robot r_i on intruder I_j can be defined as:

$$S(r_i,I_j)=\frac{v_i}{d(r_i,I_j)}\prod p(I_j,r_k) \quad (2)$$

where v_i is the robot traveling speed. $d(r_i,I_j)$ is the traveling distance between r_i and I_j. r_k is all robots within the communication range of r_i. $p(I_j,r_k)$ is the shame-level suppression from r_k on intruder I_j, which can be defined as:

$$p(I_j,r_k)=\begin{cases}1 \text{ when } r_k \text{ is not tracking } I_j\\ \chi\ (0<\chi<1) \text{ when } r_k \text{ is tracking } I_j\end{cases} \quad (3)$$

where χ is a constant representing the suppression level.

C. A Decentralized Gap-based Self-Deployment (DGSD) Algorithm

The robots whose shame level is below the threshold should re-deploy themselves properly in the deployment circle to maximize coverage. A gap-based algorithm is proposed here for this self-deployment purpose. A gap is defined as the sectors between the lines which are connecting all the tracking robots and the center of the protected area. For individual gap, the two corresponding tracking robots at the ends are called "gap builders", and the deploying robots within the gap are called "gap members". The gap members should be deployed uniformly within each gap. Based on different situations of intruders and robots, we define a gap weight W_G for each gap as:

$$w_G=\frac{s_G+\beta*(n_{IG})}{n_{rG}} \quad (4)$$

where s_G is the angle of gap G. n_{IG} and n_{rG} are the number of intruders and robots within gap G, respectively. β is a constant that adjusts the importance of n_{IG}. A gap with a higher gap weight has a higher priority to be covered by deploying robots. In other words, more deploying robots should join in the gaps with higher weights. Therefore, the objectives of gap-based method are: (a) deploy gap members uniformly within the gap; (b) switch gap members to a neighboring gap with a higher gap weight; (c) work in a distributed manner.

DGSD process runs repeatedly in each gap to dynamically redeploy gap members based on the changing environment. The DGSD process contains a round-trip to pass information to all the gap members within the gap. The round-trip starts

from a gap builder R_B. R_B generates an information pack containing its local information. The pack is delivered to the other gap builder R_B' by passing through each gap member one by one locally. During the delivering process, the information pack is updated with gap members' local information. So R_B' has a full vision of the current gap, including number of gap members, where gap starts and ends, and gap size, etc. Base on this information, R_B' is able to generate deployment positions for gap members which are distributed uniformly in the gap. The plan is delivered back to R_B through local passing agents one by one in a distributed manner, so that gap members are acknowledged with its deploying position. It is worth to note that only local communication is needed for the robots for DGSD since the information is passed one by one instead of globally broadcasting. The local communication in the DGSD process usually starts in the clockwise direction.

At the boundary of gaps, Gap builders also hold status information of the two neighboring gaps so that it can notify a gap member to switch to a neighboring gap if the neighboring gap has a much higher weight. In this manner, critical gaps will attract more robots.

Fig.2 shows an example for this DGSD process. In gap1, DGSD process is started by gap builder R_1. R_1 sends out the information pack to R_2, then R_2 sends the information to R_3. When R_4 (another gap builder) receives the information pack from R_3, it is notified that there are two gap members (robots) and three intruders in Gap1. Then R_4 updates the memory of w_{G1} and calculates proper deployment. This deployment information is delivered back to R_1 through R_3 and R_2. As a result, R_1 updates the memory of w_{G1}, and R_2 and R_3 deploy on stars. For other gaps, R_5 will switch to Gap1 from Gap2 because Gap1 has a higher weight. R_7 will stay in Gap3 to investigate intruder I_4.

It is possible that during DGSD process, a gap member who is holding the package cannot find the next robot within its communication range to deliver the package. For example, a gap member receives the package from its anti-clockwise neighbor but there is no robot on its clockwise side within its communication range. In this case, the gap member will travel along the deployment boarder in clockwise direction until a robot is detected on the clockwise side in communication range.

Fig.3 shows the block diagram of the STAGS algorithm. These two algorithms have mutual influence with each other. The shame-level based algorithm triggers robots to conduct intruder investigation. Meanwhile, the investigating robots would dynamically formulate gaps. With the new gaps, the robots use the DGSD algorithm to deploy themselves within the gaps to corporately working with the investigating robots, which would further affect the performance of future investigation.

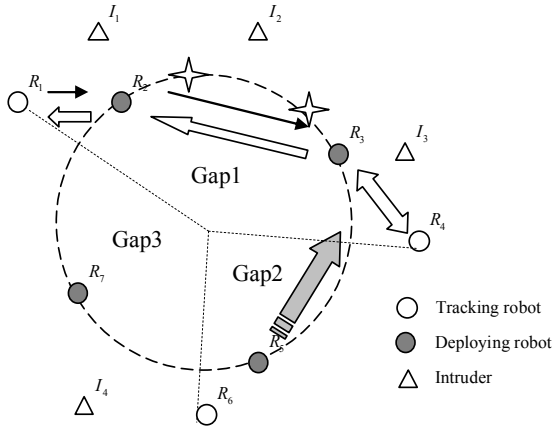

Fig. 2. One example using the DGSD algorithm.

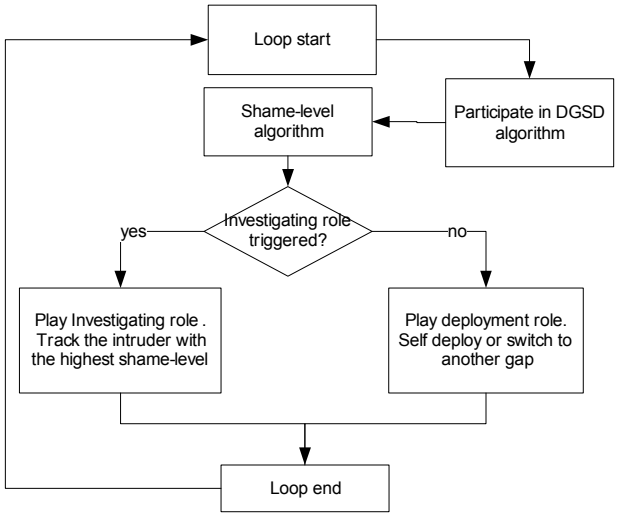

Fig. 3. The block diagram of the STAGS algorithm.

III. Online Learning and Multi-Objective Optimization on Distributed STAGS Method

The parameters of STAGS method can significantly affect the overall system performance. In addition, the environment may dynamically change over time, for example, intruders may change their intrusion behaviors (changing the velocity or strategies), or some robots may suddenly break down. Therefore it is important to find optimal STAGS parameters under dynamic situations. Ideally, the solution should be self-adaptive, which requires the robots to recognize and tune up STAGS parameters for different situations.

We are interested in finding parameters of STAGS that can optimize both missing rate and response time. Therefore, a multi-objective optimization (MOO) technique is applied. Since the robots are highly cooperated in this task, it is difficult to evaluate the system performance based on the behaviors of individual robots only. Therefore, a global sensor node is produced to implement the multi-objective optimization (MOO) to dynamically adjust the parameters of STAGS. This node is responsible of recording all detected situations, using the MOO learning method to obtain optimal STAGS parameters for each situation in real-time, and synchronizing all the robots with the latest STAGS

parameters through sensor network. The pseudo code of this process is summarized as followings:

Step1. Detect environment changes. Record current situation $S_{current}$ and previous situation $S_{previous}$.

Step2. If $S_{current}$ matches with $S_{previous}$, go to step 4; otherwise, go to step 3.

Step3. If learning on $S_{previous}$ is not finished, protect the learning process of $S_{previous}$. Go to step 4.

Step4. If a learnt adjustment $A_{current}$ for $S_{current}$ is found in memory, use $A_{current}$ to adjust the STAGS parameters. Otherwise, go to step 5.

Step5. Start, continue or resume the learning on $S_{current}$ using the NSGA-II method. When the learning process on $S_{current}$ is finished, store { $S_{current}$, $A_{current}$ } into memory.

Problem situation is defined as $S = \{s_1, s_2, ..., s_k\}$, where $s_j \mid j = 1...k$ are the parameters that describe situation S. The matching between situation S and S' can be estimated by the following equation:

$$\text{Matching} = \left(\frac{(s_1 - s'_1)}{s_1} < d_1\right) \wedge \left(\frac{(s_2 - s'_2)}{s_2} < d_2\right) \wedge ... \wedge \left(\frac{(s_k - s'_k)}{s_k} < d_k\right) \quad (5)$$

where d_k is the upper bound of the difference for s_k. Here the parameters of problem situation *S* are chosen as *{arriving rate of intruders, speed of intruders, number of robots, speed of robots}*, where the arriving rate of the intruders is the frequency of the arrivals of new intruders. d_k is set as 10% of s_k. The adjustment solution A is defined as: A = {*shame-level threshold, shame-level suppression, deployment radius*}. Deployment radius is the radius of the deployment circle as shown in Fig.1.

The learning process is evaluated based on two criteria: the intruder missing rate and the average response time to intruders. A good strategy should strike a balance between these two criteria.

NSGA-II [4] has been adopted as the MOO method for the parameter optimization and learning in STAGS. NSGA-II is a popular and efficient evolutionary algorithm for solving multi-objective optimization problems. In our work, simulated binary crossover (SBX) [3] and polynomial mutation [5] have been employed to generate offspring. After the offspring population is generated, the elitist crowded non-dominated sorting is used for selecting parents for the next generation. The complexity of NSGA-II is $O(MN^2)$, where *M* is number of objectives and *N* is population size of each evolution.

Different from single objective optimization algorithms, where often only one optimal solution is achieved, NSGA-II produces a set of Pareto-optimal solutions, i.e. in our case, the produced parameter sets to balance the missing rate and the average response time. How to pick one among Pareto-optimal solutions depends on user's preference on emphasizing one goal over another. In the experiment part, the parameter set with the lowest missing rate is selected as final adjustment option. We will analyze the solutions in discussing the simulation results using NSGA-II.

IV. SIMULATION

A. Simulation Results of STAGS with Fixed Predefined Parameters

To evaluate the performance of STAGS algorithm (S for shame-level based algorithm and D for gap-based self-deployment algorithm), two simple algorithms are defined here: numb tracking (NT) algorithm where robots always track the closest intruder, and numb deployment (ND) algorithm, where robots are initially distributed uniformly on the deployment circle and return to their initial locations when the investigation jobs are finished.

In this simulation, shame-level threshold=2.4, shame-level suppression=0.2, deployment-range=235, and $\beta = 1$ for the gap weight. Simulations each with 50,000 intruders are carried on for different algorithm combinations NT+ND, S+ND, NT+D, S+D under different situations. The missing rate and the average response time are listed in Table I. The results illustrate that S+D algorithm outperforms others.

TABLE I: SIMULATION RESULTS

		NT+ND	NT+D	S+ND	S+D
RN=8, IR=8	MS	45.28%	46.96%	8.40%	6.72%
RS=4, IS=3.0	RT	66.46	67.37	43.68	40.38
RN=8, IR=8	MS	57.03%	55.76%	17.54%	15.63%
RS=4, IS=3.5	RT	62.11	62.14	44.29	42.45
RN=6, IR=8	MS	46.20%	45.14%	14.84%	14.34%
RS=4, IS=3.0	RT	67.27	66.81	49.31	48.60
RN=6, IR=8	MS	58.18%	57.18%	23.50%	22.64%
RS=4, IS=3.5	RT	62.91	62.72	48.88	48.75
RN=4, IR=8	MS	51.84%	50.40%	25.98%	25.81%
RS=4, IS=3.0	RT	69.27	69.13	58.45	58.25
RN=4, IR=8	MS	58.94%	58.85%	36.00%	35.77%
RS=4, IS=3.5	RT	65.29	63.49	55.37	55.34

RN: number of robots, IR: intruder arriving rate ($1/\lambda$ in (7)), RS: robot speed, IS: intruder speed, MS: missing rate, RT: average response time

B. Simulation Results of STAGS with MOO-based Online Learned Parameters

In this part, the STAGS method with MOO learning approach (STAGS-MOO) is compared with the STAGS method with fixed parameters (STAGS-fixed). jMetel software package is implemented in the simulator to realize NSGA-II algorithm. jMetel is a Java-based framework aimed at facilitating the development and experiment for solving multi-objective optimization (MOO) problems [8]. In our experiment, we configure NSGA-II's evolution population size as 8.0 and the maximum evolutions as 8.0. Other parameters use default values in jMetel package: crossover probability is 0.9 and mutation probability is 1/(size of A) which is 0.33.

1) Performance Comparison under Static Situations

One merit of using the MOO-based learning is that it can learn optimal parameter sets of STAGS method to adapt to current system situation. In this part STAGS-fixed are compared with STAGS-MOO under a static situation where the environmental parameters are fixed as:

{number of robots = 8, intruders arriving rate = 8,
robots' velocity = 4/system iteration,
intruders' velocity of = 3/system iteration }

In order to provide a thorough comparison, first for each parameter we define a searching space for MOO learning based on the experiences we obtained from the simulations. Then we set up 27 static parameter sets for the STAGS-fixed method within the searching space. The searching space and the candidate static parameters are given as follows.

Shame-level threshold: 1.2, 2.4, 3.6 ∈ [0, 4]
Shame-level suppression: 0.2, 0.6, 0.8 ∈ [0, 1]
Deployment-range: 145, 235, 285 ∈ [100, 350]

For each individual fixed parameter set, a simulation is carried on for 50,000 intruders, empirically good enough to display system performance. From the experiments, the parameter set {3.6, 0.6, 145} has the best performance on missing rate, and {3.6, 0.6, 235} outperforms on response time. Then, STAGS-fixed method with these two parameter sets is compared with STAGS-MOO method. The comparison results are shown in Fig. 4.

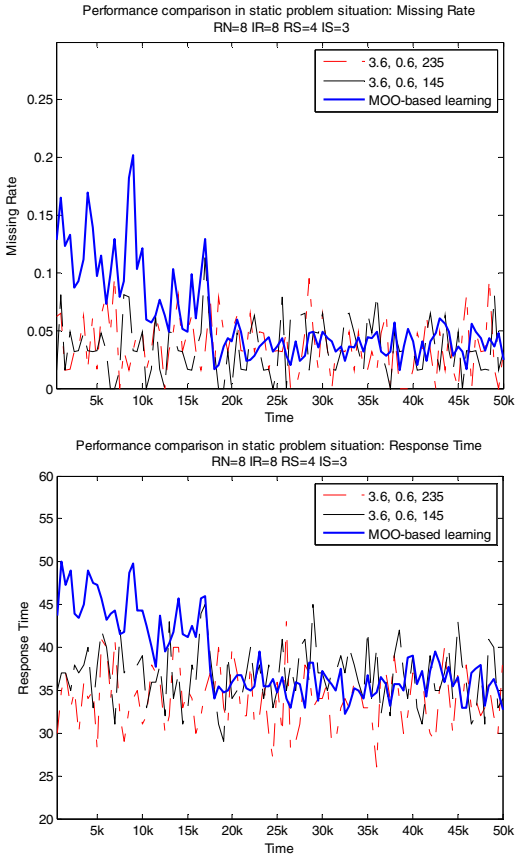

Fig. 4. The comparison results of the missing rate and the response time of the system using the STAGS-fixed method and STAGS-MOO method under a static situation.

Firstly, it can be seen that both the average missing rate and response time fluctuate at the early phase and converge over time. The main reason of this fluctuation is because that at the early phase the parameter sets generated from the NSGA-II algorithm are randomly distributed in the searching space, which may lead to significant difference in system performance. Secondly, the STAGS-MOO method can converge to a competitive system performance with STAGS-fixed method using the best parameter sets. In other words, it demonstrates that the MOO-learning method is able to find out an optimal or sub-optimal parameter set automatically for the STAGS method based. This feature can enhance the capability of the STAGS method to automatically adapt to various environmental situations instead of manually adjusting the parameters.

2) Performance Comparison under Dynamic Situations

In this sub-section, we will further compare the system performance of the STAGS-fixed method and the STAGS-MOO method under dynamic changing situations. A sequence of dynamic changing situations is listed in TABLE II, where there are 8 different situations and in each situation 50,000 intruders are released.

Firstly, simulations using the STAGS-fixed method with 27 fixed parameter sets are conducted under the dynamic situations. The result is shown in Table III, where the parameter set {1.2, 0.2, 145} and {3.6, 0.6, 235} outperforms others on missing rate and response time, respectively. 4 simulations using the STAGS-MOO method are conducted under the same dynamic situations. The average converged missing rate and response time is attached to the end of Table III. From Table III, it can be seen that the advantage of the MOO-based learning still holds under dynamic situations. The converged average missing rate and response time ranks the third place and the fourth place among all 27 STAGES-fixed experiments. There is only one fixed parameter set that can beat STAGS-MOO on both goals. However, STAGS-MOO method is not able to achieve performance that perfectly beats all STAGS-fixed cases on both goals. This may be caused by the local minimum and randomness issue of the genetic algorithm.

TABLE II: A SEQUENCE OF DYNAMIC SITUATIONS

Time	Environmental situations
T 0	arriving rate of intruders = 8, number of robots = 8, velocity of robots = 4, velocity of intruders = 3
T 1	Decrease number of robots by 1
T 2	Decrease velocity of robots by 0.5
T 3	Increase arriving rate of intruders by 3
T 4	Increase velocity of intruders by 0.5
T 5	Increase number of robots by 1
T 6	Increase velocity of robots by 0.5
T 7	Decrease arriving rate of intruders by 3
T 8	Decrease velocity of intruders by 0.5. Now the environmental situation is the same with T0 again.

3) The Performance Improvement with the Learnt Experience

At last we will demonstrate how the learnt experience of the MOO-based learning can contribute to the performance of the STAGS system under dynamic situations. We extend the experiments from the previous sub-section by repeating all of the situations one more time from T8 to T16. Applying STAGS-MOO, we got the average missing rate and response time for T0~T8 are 0.1608 and 45.20, respectively. The average missing rate and response time for T8-T16 have been reduced to 0.1330 and 43.47, respectively which are improved by 20.90% and 3.9%, respectively. This is because that the STAGS-MOO system can use the learnt experience

(the perimeter sets) for the initial generation for the situations instead of starting from a random distribution.

TABLE III: PERFORMANCE OF STAGS-FIXED AND STAGS-MOO UNDER DYNAMIC SITUATIONS

ST	SS	DR	MS	RT
3.6	0.8	285	0.2302	48.92
3.6	0.8	235	0.2266	48.84
3.6	0.8	145	0.2322	50.83
3.6	0.6	285	0.1675	42.95
3.6	0.6	235	0.1504	41.69
3.6	0.6	145	0.1307	42.67
3.6	0.2	285	0.2108	44.34
3.6	0.2	235	0.1821	43.51
3.6	0.2	145	0.1352	43.58
2.4	0.8	285	0.3070	55.48
2.4	0.8	235	0.3091	55.79
2.4	0.8	145	0.3128	55.89
2.4	0.6	285	0.2003	47.78
2.4	0.6	235	0.1970	47.75
2.4	0.6	145	0.1944	48.61
2.4	0.2	285	0.2068	45.79
2.4	0.2	235	0.1841	45.78
2.4	0.2	145	0.1517	48.38
1.2	0.8	285	0.3212	56.47
1.2	0.8	235	0.3152	56.17
1.2	0.8	145	0.3184	56.29
1.2	0.6	285	0.2247	51.03
1.2	0.6	235	0.2243	51.01
1.2	0.6	145	0.2263	50.97
1.2	0.2	285	0.1550	45.33
1.2	0.2	235	0.1414	44.61
1.2	0.2	145	0.1242	45.98
MOO	**MOO**	**MOO**	**0.1330**	**43.47**

Notes: ST: shame level threshold, SS: shame level suppression, DR: deployment range, MS: missing rate, RT: average response time. The last row is performance of STAGS-MOO

V. CONCLUSIONS

In this paper, we propose a STAGS algorithm for intruder detection in complex security defense tasks. A shame-based approach is developed for dynamic task allocation among robots to track the detected intruders, and a gap-based method is developed for the self-deployment of remaining robots. This STAGS algorithm is distributed, where only local communication among robots are needed and robots make their movement decisions only based on their local contextual information. To further improve the system robustness and adaptation, a MOO-based online learning method is developed to dynamically adjust the parameters of the STAGS method.

In the future, we will investigate the following issues. First more rational and smart intruders, who are not limited to appear under Poisson distribution and travel linearly, will be considered in the simulations. Second, research on more complex terrain situation will be conducted, such as some static and more obstacles in the environment.

REFERENCES

[1] N. Agmon, S. Kraus, and G. A. Kaminka, "Multi-Robot Perimeter Patrol in Adversarial Settings", *In Proceedings of the 2008 IEEE International Conference on Robotics and Automation*, pp. 2339–2345.

[2] S.C.Botelho and R. Alami, "M+: A scheme for multi-robot cooperation through negotiated task allocation and achievement," *In Proceedings of the 1999 IEEE International Conference on Robotics and Automation*, pp. 1234–1239.

[3] K. Deb and R. B. Agrawal, "Simulated binary crossover for continuous search space," in *Complex Syst.*, Apr. 1995, vol. 9, pp. 115–148.

[4] K. Deb, S. Agrawal, A. Pratap, and T. Meyarivan, "A fast elitist nondominated sorting genetic algorithm for multi-objective optimization: NSGA-II," in *Parallel Problem Solving from Nature (PPSN VI)*, M. Schoenauer et al., Eds. Berlin, Germany: Springer, 2000, pp. 849–858.

[5] K. Deb and M. Goyal, "A combined genetic adaptive search (GeneAs) for engineering design," *Comput. Sci. and Informatics*, vol. 26, no. 4, pp. 30–45, 1996.

[6] B. Dias, R. Zlot, N. Kalra, and A. Stentz, "Market-based multirobot coordination: A survey and analysis," Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, *Tech. Rep. CMU-RI-TR-05-13*, April 2005.

[7] M. Dorigo, E. Bonabeaub, and G. Theraulaz, "Ant algorithms and stigmergy", *Future Generation Computer Systems*, 16, pp. 851–871. 2000.

[8] J.J. Durillo, A.J. Nebro, F. Luna, B. Dorronsoro and E. Alba, "JMetel - A Framework for Multi-Objective Optimization," http://jmetal.sourceforge.net/

[9] F. Fave, S. Canu, L. Iocchi, D. Nardi, and V. A. Ziparo, Multi-Objective Multi-Robot Surveillance, *4th Int. Conf. on Autonomous Robots and Agents*, 2009.

[10] A. Gage, R. Murphy, K. Valavanis, and M. Long, "Affective Task Allocation for Distributed Multi Robot Teams." *CRASAR-TR2004-26*.

[11] Y. Gao and W. Wei, "Multi-Robot Autonomous Cooperation Integrated with Immune Based Dynamic Task Allocation", *In Proceedings of the Sixth International Conference on Intelligent Systems Design and Applications (ISDA'06)*.

[12] B. P. Gerkey and M. J. Matari´c, "Sold! Auction methods for multirobotCoordination", *IEEE Transactions on Robotics and Autonomous Systems*, 18(5):758–768, October 2002.

[13] D. B. Kingston, R. Holt, R.W. Beard, T. McLain, and D. Casbeer, "Decentralized perimeter surveillance using a team of UAVs," *AIAA Guidance, Navigation, and Control Conference and Exhibit*, San Francisco, California, August 2005, AIAA 2005-5831.

[14] A. Machado, G. Ramalho, J. Zucker and A. Drogoul, "Multi-Agent Patrolling: an Empirical Analysis of Alternative Architectures," *Multi-Agent Based Simulation (MABS'2002)*, Bologna, 2002.

[15] L. E. Parker, ALLIANCE: An Architecture for Fault Tolerant Multi-Robot Cooperation, *IEEE Trans. on Robotics and Automation*, 14(2), 1998, pp.220-240.

[16] S. Singh and S. Thayer, "Immunology Directed Methods for Distributed Robotics: A Novel Immunity-Based Architecture for Robust Control & Coordination" *SPIE: Mobile Robots XVI*, v. 4573.2001

[17] R. Vidal, O. Shakernia, H. J. Kim, H. Shim, and S. Sastry, "Multi-Agent Probabilistic Pursuit-Evasion Games with Unmanned Ground and Aerial Vehicles", *IEEE Trans. on Robotics and Automation*, vol. 18, no. 5, pp.662-669, 2002.

[18] G. Wang, W. Gong, and R. Kastner, "System Level Partitioning for Programmable Platforms Using the Ant Colony Optimization", *13th International Workshop on Logic and Synthesis(IWLS'04)*, June 2004.

[19] H. Wu, G. Tian and B. Huang, "Multi-robot Collaboration Exploration Based on Immune Network Model," *In Proceedings 2008 IEEE/ASME International Conference on Advanced Intelligent Mechatronics*, pp. 1207-1212.

[20] R. Zlot and A. Stentz, "Market-based multi-robot coordination for complex tasks", *International Journal of Robotics Research*, 25(1), January 2006, pp73-101.