creative commons

COMMONS DEED

저작자표시-변경금지 2.0 대한민국

이용자는 아래의 조건을 따르는 경우에 한하여 자유롭게

- 이 저작물을 복제, 배포, 전송, 전시, 공연 및 방송할 수 있습니다.
- 이 저작물을 영리 목적으로 이용할 수 있습니다.

다음과 같은 조건을 따라야 합니다:

저작자표시. 귀하는 원저작자를 표시하여야 합니다.

변경금지. 귀하는 이 저작물을 개작, 변형 또는 가공할 수 없습니다.

- 귀하는, 이 저작물의 재이용이나 배포의 경우, 이 저작물에 적용된 이용허락조건을 명확하게 나타내어야 합니다.
- 저작권자로부터 별도의 허가를 받으면 이러한 조건들은 적용되지 않습니다.

저작권법에 따른 이용자의 권리는 위의 내용에 의하여 영향을 받지 않습니다.

이것은 이용허락규약(Legal Code)을 이해하기 쉽게 요약한 것입니다.

Disclaimer

dCollection

Policy Learning for Task Allocation from Manual Demonstrations by a Human User

A Thesis

by

SEIL AN

Presented to the Faculty of the Graduate School of
Seoul National University
in Partial Fulfillment
of the Requirements
for the Degree of

MASTER OF SCIENCE

Department of Mechanical & Aerospace Engineering

Seoul National University

Supervisor : Professor H. Jin Kim

AUGUST 2014

Abstract

We present a policy learning algorithm for task allocation which has multiple objectives. Usually in many task allocation algorithms, the total distance which robots move is considered as an objective but if there exist threats on robots, avoiding threats also can be an objective of task allocation. The proposed algorithm learns a policy defined by weights of total distance and threat level. A Bayesian approach and k-Nearest Neighbor classifier are employed as learning algorithms and the comparison of these approaches is presented. The policy learning algorithm precisely estimates the policy as the results of task allocations are accumulated. By using the proposed algorithm, a manual task allocation of the user can be successfully replaced with the automated allocation algorithm using the same policy. As an application, reliability of the human user in task allocation will be figured out, and an user assistance algorithm is also presented to support task allocation of the human user. The performance and the reliability of the user can be improved using the assistance algorithm.

Table of Contents

Page

Abstract ii

Table of Contents iii

List of Figures iv

Chapter

1 Introduction 1

2 Background 4

2.1 Problem formulation 4

2.2 Task allocation algorithm 7

3 Policy learning 9

3.1 Setup : data generation and user interface 9

3.2 Feature selection for machine learning 10

3.3 Policy estimation algorithm 12

3.3.1 Naive Bayes classifier 13

3.3.2 Maximum likelihood estimation 14

3.3.3 k-Nearest Neighbor Classifier 16

4 Results 20

4.1 Convergence of policy estimation error 21

4.2 Policy estimation test 22

4.3 User policy tracking test 23

4.4 User data reliability test 25

4.5 User assistance algorithm 27

5 Conclusions 37

List of Figures

1.1 An algorithm for learning manual task allocations 3

2.1 Definition of threat level. Thick line is the exposed section. 6

3.1 (a) Manual task allocation in user interface (solid lines), (b) Automated task allocation in user interface (thick dotted lines) 18

3.2 Plot of training data in the feature space 19

4.1 Plot of estimation errors as the results of task allocations are accumulated 21

4.2 (a) Results of task allocation of User A in feature space, (b) Results of task allocation of User N in feature space 29

4.3 (a) Automated task allocation with learned policy from User A, (b) Automated task allocation with learned policy from User B. 30

4.4 (a) Plot of tracking performance in terms of trials, (b) Plot of tracking performance in terms of weight sets 31

4.5 In each plot, automated algorithm(solid contour) shows better distributions than manual allocations(dotted contour).(a) Comparison of task allocation results from User A and automated algorithm(contour is 2σ of distribution), (b) Comparison of task allocation results from User B and automated algorithm(contour is 2σ of distribution) 32

4.6 (a) Distribution of the estimated policy from User A and automated algorithm, (b) Distribution of the estimated policy from User B and automated algorithm 33

4.7 Manual task allocation plot(circle) and suggested automated allocations (square) 34

4.8 Manual task allocation plot(circle) and suggested automated allocations (square) 34
4.9 (a) Manual task allocation demonstration, (b,c,d) Suggested task allocations by the automated algorithm 35
4.10 Cost plot of manual task and supported task allocation 36
4.11 Estimated policy distribution of manual task and supported task allocation 36

1

Introduction

For an operation of multi-robot systems, task allocation is an important problem. The main purpose of task allocation is assigning tasks or resources effectively and quickly to the robots, but in some cases, survivability can also be an important performance measure [1]. For example, if the robots are deployed to a rescuing mission or battlefield, there may exist threats on robots [2]. For those situations, threat avoidance is included as an additional mission of task allocation. In the task allocation algorithm, these two performance metric, total distance and threat avoidance, are needed. In this paper, we simplified this as a weighted sum of two performances in one objective function. Then the weight set can be understood as a policy of task allocation.

In this situation, however, setting the proper objective function autonomously can be a difficult problem because there are various policies as the goals which the user wants to achieve. For this reason, human user's monitoring or intervening the process is required in many cases, though there are various task allocations algorithms which run fully autonomously. In Multi Autonomous Ground-robotic International Challenge (MAGIC 2010), many teams considered interventions of human operators while autonomous algorithms are

running [3–5]. On the other hand, due to insufficiency of possible human operators, it requires an automation of task allocation which includes setting the policy.

To improve automation in task allocation, learning from demonstration can be an answer. Several algorithms for policy learning have been introduced by using demonstrated trials as training data to learn the policy [6]. A policy gradient method was used to learn the motion of robot in [7,8]. Dynamic programming was used to learn an air-combat strategy in [9]. Bayesian approaches were tried for policy estimation in [10–12] which assume probabilistic distribution of training data. These approaches are useful when there are a lot of training data for policy learning. In several researches k-Nearest Neighbors (kNN) classifier was used as a policy learning algorithm [13–16]. Although these researches about policy learning are for motion planning or path planning of robots, there exist learning algorithms for the task allocation problem. In [17], imitation learning was conducted for expert's demonstration of task allocation. A closed-loop method which adjusts the parameters for auction algorithm was used to estimate the best cost for tasks in the task allocation problem in [18]. However, the objectives of learning in these researches are focused on the type of tasks and the threats of task allocation are not considered. There were trials to solve inverse linear programming directly but they considered a simple assignment problem in which each agent performs only one task [19].

In this research, we present an algorithm which learns hidden weights of task allocation from manual allocation of the human users. The weights in objective function control importance between fast arrival and threat avoidance in task allocations. To estimate weights of manual task allocation, we first gather automated allocation results from several candidate sets of the weight values. This is a data accumulation process and the results are used as the training set for policy learning. We use the Bayesian approach and kNN for policy learning and comparison of these algorithms are presented. By applying the policy learning algorithm, a policy can be estimated from user's manual mission planning. From the estimated policy, automation of task allocation is possible as shown in Fig. 1.1. In the monitoring part, several applications can be conducted using the estimation algorithm.

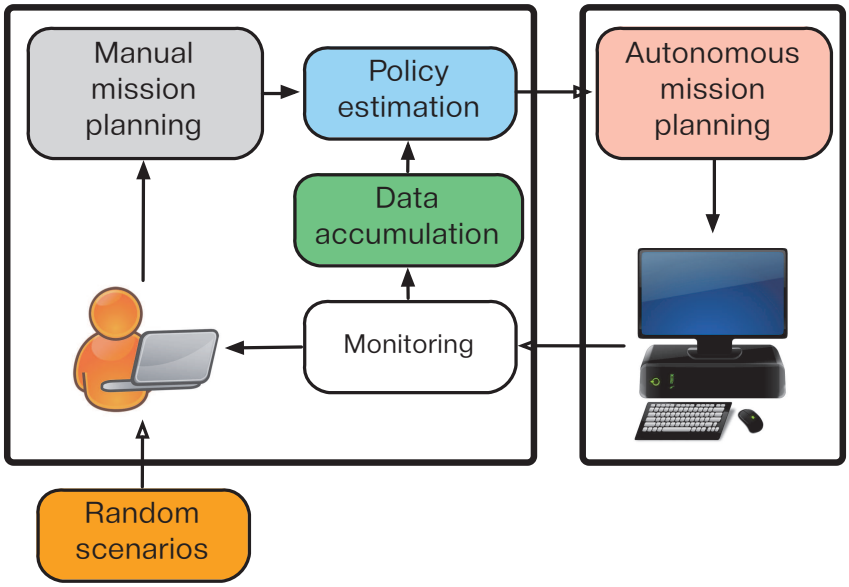


Figure 1.1: An algorithm for learning manual task allocations

The reliability of task allocation demonstrations can be calculated with the algorithm, which shows consistency of task allocations. The user assistance algorithm is introduced to improve the reliability and performance of manual task allocation.

In Section 2, the background of this research is presented. Problem formulation and automated task allocation algorithm are presented. The algorithm used to learn the policy of human user in task assignment is described in Section 3. Several test results are introduced in Section 4. Convergence of policy estimation error, user policy tracking test, user reliability test, and the user assistance algorithm are also presented. Section 5 is conclusions.

2

Background

In this section, problem formulation for task allocation is introduced [20, 21]. The threat level is defined to be similar to the distance in order to make problem simple. Genetic algorithm is used for task allocation algorithm and detailed genetic operations are presented.

2.1 Problem formulation

We consider task allocation scenarios in which the agents perform multiple tasks. To compose a realistic task allocation scenario, it is assumed that every agent starts from a fixed position and has to arrive at a fixed goal position and the tasks are deployed randomly in the field. While moving to goal position, the agents perform a fixed number of tasks. The paths of agents are considered as straight lines between nodes which include agents,

tasks and the goal point. This process can be formulated as follows:

$$x_{i,j} = \begin{cases} 1 & \text{if there exists path } i \text{ to } j \\ 0 & \text{otherwise} \end{cases} \tag{2.1}$$

$$x_{i,j} = 0 \;\; \text{if } i = j \tag{2.2}$$

$$x_{i,j} \neq x_{j,i} \tag{2.3}$$

$$i, j \in \{1, 2, ..., N\}$$

$$N = N_a + N_t + 1 \tag{2.4}$$

The adjacency matrix $x_{i,j}$ for task allocation is defined in (2.1) which can give whole information of paths. In the adjacency matrix, each path is directed as expressed in (2.3). Eq. (2.2) limits making a path to itself. N_a and N_t are the number of total agents and the number of total tasks, respectively. The number of total nodes N is expressed as (2.4) including the agents and tasks. The value 1 is added to total number because of the goal point.

Every path in task allocation can be expressed with the adjacency matrix in (2.1), but it is convenient to use another type of the allocation matrix to express constraints of task allocation. The allocation matrix can be expressed as follows:

$$z_{i,j} = \begin{cases} 1 & \text{if task } j \text{ is assigned to agent } i \\ 0 & \text{otherwise} \end{cases} \tag{2.5}$$

$$\sum_{i}^{N_a} z_{i,j} \leq 1 \tag{2.6}$$

$$\sum_{j}^{N_t} z_{i,j} = 3 \tag{2.7}$$

$$i \in \{1, 2, ..., N_a\}$$

$$j \in \{1, 2, ..., N_t\}$$

The allocation matrix in (2.5) shows which tasks are assigned to each agent but the matrix does not inform the order of task performing. The rules of task allocation can be expressed

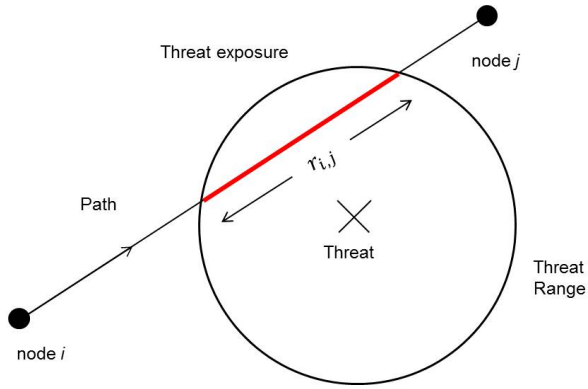


Figure 2.1: Definition of threat level. Thick line is the exposed section.

with this allocation matrix. Eq. (2.6) is a constraint that the task can not be allocated by more than 1 agent. Eq. (2.7) limits the number of tasks which can be assigned by each agent as 3.

The total distances traversed by each agent can be expressed with the adjacency matrix and distance matrix. The distance matrix is a N by N matrix which includes the information of distances between nodes. The distance matrix $q_{i,j}$ and total distance D are defined by the following:

$$q_{i,j} = \text{distance between node } i \text{ and } j \tag{2.8}$$
$$D = \sum_{i}^{N} \sum_{j}^{N} q_{i,j} x_{i,j}$$

In this research, we assume that task allocation considers both distance and threat level as a performance measure. The threat level is defined as total threat exposure distance exposed to the threat, i.e, the length of the straight line segment located in the threat range as shown in Fig. 2.1. The threat level is calculated when the agents pass through threat range. Because the agents follow straight paths between nodes, the threat level can

be formulated similarly with the distance matrix of (2.8).

$$r_{i,j} = \text{threat level between node } i \text{ and } j \tag{2.9}$$
$$T = \sum_{i}^{N} \sum_{j}^{N} r_{i,j} x_{i,j}$$

where T is the total threat level. The matrices $q_{i,j}$ in (2.8) and $r_{i,j}$ in (2.9) contain information of task allocation scenarios.

To consider both total distance and threat level as objectives of task allocation, the objective function is composed of weighted sum of two value, D and T. The equation is

$$P = w_1 D + w_2 T$$
$$\mathcal{W} = (w_1, w_2) \tag{2.10}$$

The weight set of task allocation $\mathcal{W}$ in (2.10) can be understood as a policy of task allocation because this value reflects characteristic of task allocation. For example a policy with the high weight on the total distance is the same with the weight set of which w_1 is high.

2.2 Task allocation algorithm

In this research, the policy learning algorithm uses a model-based learning so the algorithm requires pre-generated training data of task allocations. This means that repeated automated task allocation must be conducted for training data so effective task allocation algorithm is necessary to facilitate the data collection. To achieve this, automated task allocation based on genetic algorithm is used in this research [22, 23]. The main operations of genetic algorithm can be expressed as below.

- *Initialization*

  Before applying genetic operations, feasible solutions have to be made initially as a population of genetic algorithm. In this research, this process means initializing the adjacency matrix in (2.1). The initial solutions are generated randomly satisfying the constraints.

- *Selection*

  Deterministic tournament selection is used in this phase [24]. Tournament selection is a method of selecting a solution which gives best performance among the feasible solutions. For crossover phase, two best solutions are selected as parents solutions.

- *Crossover*

  For the task allocation problem, crossover operation can make infeasible solution because there exist a constraint which prohibits the task being assigned by two agents. Therefore, we used two-point crossover method which changes the order of performing tasks without breaking feasibility of the solution.

- *Mutation*

  For the task allocation problem, there can exist two types of mutation. First one is changing assignment of the tasks to the agents and the other is changing the order of performing tasks. Mixing these two types of mutation properly is important in genetic operation. For example, if the tasks are already assigned to each agent properly, a good solution can be obtained by simply repeating the mutation which changes the order of performing tasks. In the crossover phase, changing the assignment of tasks are not considered so in this phase, mutation which changes the assignment is mainly used for compensation.

3

Policy learning

We use a model-based learning algorithm to estimate the policy of the user task allocation, which necessitate the training data for learning. Several weight set models are defined and autonomous task allocation is conducted with each weight set model for different scenarios. The characteristic of each task allocation depends on the weight set so the weight set model can be understood as the policy of task allocation. Total distance and threat level are set as features for policy learning is introduced. Two Bayesian approaches and kNN are employed as policy learning algorithms. In kNN approach, probabilistic distribution of task allocation demonstrations is used to assign weight of each demonstration.

3.1 Setup : data generation and user interface

For data generation, the automated task allocation algorithm mentioned in the previous section is used. The weight set models are defined as follows:

$$\mathcal{W} = (w_1, w_2) \tag{3.1}$$

$$\mathcal{W}_i \in \{(0.0, 1.0), (0.1, 0.9), ..., (1.0, 0.0)\}$$

where w_1 and w_2 are weights on total distance and total threat level, respectively, as defined in (2.10). For example, the weight set $(0.0, 1.0)$ is a policy which cares only about the avoidance of threat as the performance measure.

In this research, we handle task allocation demonstrations of human users. To achieve this, an interface for human user task allocation is needed so we developed this as shown in Fig. 3.1. With this interface, the user can generate a manual task allocation result quickly by clicking agent and task buttons in sequence. For human-robot interaction, the human user can compare his own task allocation result with automated result right after his demonstration. Two scores, the total distance score and the threat level score, are shown to the user and the task allocation results are accumulated for an analysis. From this, the human user can try to reassign the tasks to improve the resulting scores.

3.2 Feature selection for machine learning

In the previous section, we considered the data generation for the policy estimation. The result of task allocation contains information about every path of the agents and the order of task allocation. However, not all the information is necessary for the policy learning algorithm. The policy learning can be conducted only using the distance score and threat level score from task allocation results. From the generated data, the total distance of task allocation and the total threat level can be derived because task allocation scenarios are assumed to be known in the policy learning phase. These two values are used as features of the policy learning for task allocation. These can be express as follows:

$$
\begin{aligned}
P^\star &= w_1\ D^\star + w_2\ T^\star \qquad (3.2)\\
D^\star &= k_D \sum_i^N \sum_j^N q_{i,j} x^\star_{i,j}\\
T^\star &= k_T \sum_i^N \sum_j^N r_{i,j} x^\star_{i,j}
\end{aligned}
$$

in the equations, $x^\star$ is an optimal solution which gives an optimal performance $P^\star$ for the user. Because matrices $q_{i,j}$ and $r_{i,j}$ are known from the task allocation scenario, the total distance and the threat level can be calculated. The purpose of policy learning is to find out the hidden weight set w_1 and w_2 in (3.2). The reason of using these two values as features is that different policies of task allocation are expected to generate distinctive distributions of task allocation results in the feature space. For example, a policy with high weight on distance in task allocation gives small distance as a result.

In (3.2), k_D and k_T are normalizing factors which are used to make feature values D and T. As mentioned before, every task allocation scenario is randomly generated and the differences in each scenario make errors in learning task allocation results. To reduce gaps from using different scenarios, the normalizing factors k_D and k_T are defined as follows:

$$k_D \sum_i^N \sum_j^N q_{i,j} = 1$$
$$k_T \sum_i^N \sum_j^N r_{i,j} = 1$$

The normalizing factors which are defined as these equations make total distances and total threat levels of all paths as 1. By normalizing distance and threat level, different characteristics of scenarios can be generalized. For example, there only exist long paths between nodes in some scenarios. In this case, all of the elements of matrix r are relatively larger than other scenarios. The normalizing process makes the results from these scenarios also useful for the learning phase. Furthermore, this work adjusts relations between two feature values, the distance and the threat level and reduces outliers in feature space plots.

The distributions of all 11 weight set models are shown in Fig. 3.2. In the figure, the results with high priority on total distances are located in the top left parte showing small total distance and large threat level. Table 3.1 shows the means of feature values of task allocation results by the weight set models.

Table 3.1: Means of training data in feature space

Weightset	Mean(Distance)	Mean(Threat)
(0.0,1.0)	0.513	0.081
(0.1,0.9)	0.488	0.082
(0.2,0.8)	0.456	0.083
(0.3,0.7)	0.433	0.091
(0.4,0.6)	0.416	0.102
(0.5,0.5)	0.398	0.121
(0.6,0.4)	0.388	0.155
(0.7,0.3)	0.372	0.185
(0.8,0.2)	0.366	0.221
(0.9,0.1)	0.361	0.287
(1.0,0.0)	0.363	0.377

3.3 Policy estimation algorithm

The model-based estimation algorithm was used to estimate policy of the user. This method requires pre-generated training data for estimation. The training data was generated from automated task allocations for various scenarios and policies. The best way to estimate the policy of the user is performing task allocations for all possible policies and choosing the most similar policy with the policy of the user but it takes too much time to generate task allocation data for every new scenario. The model-based estimation algorithm uses pre-generated training data to estimate the policy so makes quick estimation for randomly deployed task allocation scenarios.

3.3.1 Naive Bayes classifier

The Naive Bayes classifier can be used to learn the user policy. For a learning process, the training data which are mentioned in previous section are used. The training data set was made by automated allocation algorithm with 11 weights models, (0.0, 1.0), (0.1, 0.9) to (1.0, 0.0). Each weight can represent the policy of task allocation. For example, weight model (1.0, 0.0) is the same with the policy that only considers arrival time in task allocation.

Two values from manual task allocation, arrival time and threat level, are the features for machine learning. From Naive Bayes rule, the posterior probability can be derived as

$$p(\mathcal{W}|D,T) = \frac{p(\mathcal{W})p(D,T|\mathcal{W})}{p(D,T)} \tag{3.3}$$

$$= \frac{p(\mathcal{W})p(D|\mathcal{W})p(T|\mathcal{W})}{p(D,T)} \tag{3.4}$$

where $\mathcal{W}$ is one of the weight set models. From (3.3)–(3.4), Naive Bayes assumption that the feature D and T are conditionally independent is used. We assume Gaussian distribution for each likelihood:

$$p(D|\mathcal{W}) = \frac{1}{\sqrt{2\pi\sigma_D^2}} e^{-\frac{(D-m_D)^2}{2\sigma_D^2}}$$

$$p(T|\mathcal{W}) = \frac{1}{\sqrt{2\pi\sigma_T^2}} e^{-\frac{(T-m_T)^2}{2\sigma_T^2}}$$

where the means m_D, m_T and standard deviations σ_D,σ_T are calculated from training data. The prior probability is assumed to be equal for each weight set and the evidence can be expressed as

$$p(D,T) = \sum_i^n p(\mathcal{W}_i)p(D|\mathcal{W}_i)p(T|\mathcal{W}_i) \tag{3.5}$$

$$\text{where } \mathcal{W}_i \in \{(0.0,1.0),(0.1,0.9),...,(1.0,0.0)\}$$

It is possible to classify the result of task allocations using classifying or weighted average. The classifying rule is:

Approach A:

$$\mathcal{W}(D,T) = \text{argmax}_{\mathcal{W}_i}\ p(\mathcal{W}_i|D,T) \tag{3.6}$$

This equation represents a process of choosing the most probable weight set from the training data.

On the other hand, for the user policy learning, the estimation algorithm is defined as a weighted sum of candidate models from the train data. This can be written as

Approach B:

$$\mathcal{W}(D,T) = \sum_{i}^{n} p(\mathcal{W}_i|D,T)\mathcal{W}_i \tag{3.7}$$

This equation is the expectation value of policy of the manual assignment result.

Up to now, the estimation from one result of task allocation was considered using (D,T) term. The extension for estimation from multiple results using approach B can be written as

$$p(\mathcal{W}_i|\mathcal{D},\mathcal{T}) = \frac{\prod_j^M p(\mathcal{W}_i|D_j,T_j)}{\sum_i^n(\prod_j^M p(\mathcal{W}_i|D_j,T_j))} \tag{3.8}$$

$$\mathcal{D},\mathcal{T} = \{D_0, D_1, ..., D_M\}, \{T_0, T_1, ..., T_M\}$$

where D_j and T_j is arrival time and threat level of the jth trial and M is the total number of trials. Now the estimation becomes:

$$\mathcal{W}(\mathcal{D},\mathcal{T}) = \sum_{i}^{n} p(\mathcal{W}_i|\mathcal{D},\mathcal{T})\mathcal{W}_i \tag{3.9}$$

This equation is used to estimated the user's policy from several manual task allocation results.

3.3.2 Maximum likelihood estimation

Maximum likelihood estimation can be used to learn the policy of the task allocation from the user. This method does not use assumption of conditional independency between

total distance and total threat level. It may cause some errors to assume conditional independency because the relation between two feature values is unknown. For given test data D and T, calculating $p(\mathcal{W}|D,T)$ is goal for policy estimation from the task allocation result. If multivariate Gaussian distribution is assumed for training data of weight sets:

$$p(D,T|\mathcal{W}) \quad = \quad \mathcal{N}(D,T|\mu_{\mathcal{W}},\Sigma_{\mathcal{W}}) \tag{3.10}$$

where $\mu_{\mathcal{W}}$ and $\Sigma_{\mathcal{W}}$ are mean and variance of the weight set $\mathcal{W}$. These parameters are derived from each data set of weight sets. The training data of weight set $\mathcal{W}$ can be expressed as

$$\mathcal{X}_{\mathcal{W}} = (\mathbf{x}_1, ..., \mathbf{x}_N)^T$$
$$\mathbf{x}_i = (D_i, T_i)$$

where N is the number of features in training data of weight set W. The parameters of the distribution $\mu_{\mathcal{W}}$ and $\Sigma_{\mathcal{W}}$ are calculated to maximize the following log likelihood function:

$$\begin{aligned} \ln\, p(\mathcal{X}_{\mathcal{W}}|\mu_{\mathcal{W}},\Sigma_{\mathcal{W}}) = &-N\, \ln(2\pi) - N\, \ln|\Sigma_{\mathcal{W}}| \\ &-\frac{1}{2}\sum_{i=1}^{N}(\mathbf{x}_i - \mu_{\mathcal{W}})^T\Sigma_{\mathcal{W}}^{-1}(\mathbf{x}_i - \mu_{\mathcal{W}}) \end{aligned} \tag{3.11}$$

Differentiating (3.11) by $\mu_{\mathcal{W}}$ and $\Sigma_{\mathcal{W}}$, the parameters of the distributions can be calculated as follows:

$$\mu_{\mathcal{W}} = \frac{1}{N}\sum_{i=1}^{N}\mathbf{x}_i \tag{3.12}$$

$$\Sigma_{\mathcal{W}} = \frac{1}{N}\sum_{i=1}^{N}(\mathbf{x}_i - \mu_{\mathcal{W}})(\mathbf{x}_i - \mu_{\mathcal{W}})^T \tag{3.13}$$

With the parameters, likelihood of each weight set can be calculated with following multivariate Gaussian distribution

$$p(\mathbf{x}|\mathcal{W}) = \frac{1}{\sqrt{(2\pi)^2|\Sigma_{\mathcal{W}}|}}\exp(-\frac{1}{2}(\mathbf{x} - \mu_{\mathcal{W}})^T\Sigma_{\mathcal{W}}^{-1}(\mathbf{x} - \mu_{\mathcal{W}})) \tag{3.14}$$
$$\mathbf{x} = (D,T)$$

where $\mathbf{x}$ is a given task allocation result which can be understood as test data. From (3.3), we assuming prior $p(\mathcal{W})$ is same for all weight set models and the evidence can be written as

$$p(D,T) = \sum_{i}^{n} p(\mathcal{W}_i)p(D,T|\mathcal{W}) \tag{3.15}$$

$$\text{where } \mathcal{W}_i \in \{(0.0,1.0),(0.1,0.9)...(1.0,0.0)\}$$

With this equation, the posterior $p(\mathcal{W}|D,T)$ can be calculated. The remaining classifying algorithm is the same with the previous equations which use Naive Bayes algorithm.

3.3.3 k-Nearest Neighbor Classifier

Non-Bayesian approach can be used as an classifying algorithm. For this kind of problems, k-Nearest Neighbor (kNN) can be a simple approach which can be easily applied. By calculating distances between test data and pre-generated training data, the estimation of the policy can be obtained. A difference with Bayesian approach is that this method does not assume the probabilistic distribution of training data. After a manual task allocation, an estimation for one trial is calculated. Weighted kNN which was presented by Dubani is used to this learning algorithm [25]. The algorithm can be expressed as follows:

$$\text{dist}_1^{j_1} \leq \text{dist}_2^{j_2} \leq ... \leq \text{dist}_k^{j_k} \tag{3.16}$$

$$w_1 \geq w_2 \geq ... \geq w_k \tag{3.17}$$

$$w_i = k - i + 1 \tag{3.18}$$

$$\text{score}_j = \sum_{i}^{k} \begin{cases} w_i \text{ if } j_i = j \\ 0 \text{ otherwise} \end{cases} \tag{3.19}$$

$$\mathcal{W}(D,T) = \text{argmax}_{\mathcal{W}_j} \text{score}_j \tag{3.20}$$

where j is an index of class and the notation j_i means that ith distance is calculated between test point and one of the training point from class j. Eq. (3.18) is a weighting equation which gives high priority to more closer points. Normal kNN is a case when the weight w_i is

constant as 1. In (3.19) and (3.20), the weight set which obtains highest score is selected as a final classified result. k value of kNN is set experimentally to improve overall classifying performance.

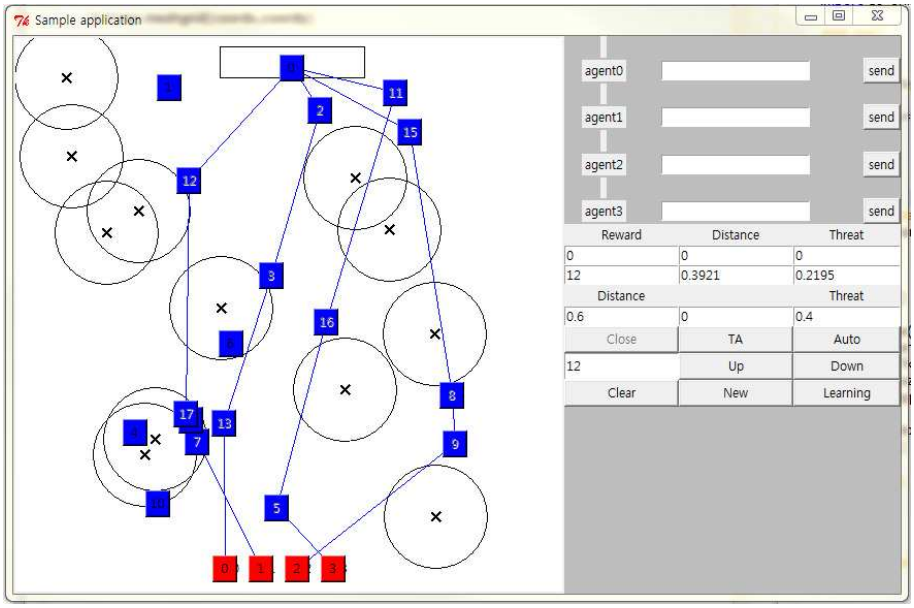
These equations are classifying process for one manual task allocation result but as mentioned before, we are interested in finding out a representing policy of the user using batch process of multiple results of manual allocations. For a batch process, each result of task allocation is differently weighted [26]. Because there is a tendency that test data are sparsely distributed in feature space, to improve classifying performance Gaussian distribution was applied for weighting. The idea is giving higher weight to the test point near the mean of test data distribution, which is assumed to be more important than other points. Parameters of Gaussian distribution for test data can be calculated using expectation maximization algorithm in (3.12), (3.13), and (3.14). The weighting method in batch process can be expressed as:

$$\mathfrak{w}_i = \frac{p(\mathbf{x}_i|\mathcal{X})}{\sum_i^N p(\mathbf{x}_i|\mathcal{X})} \tag{3.21}$$

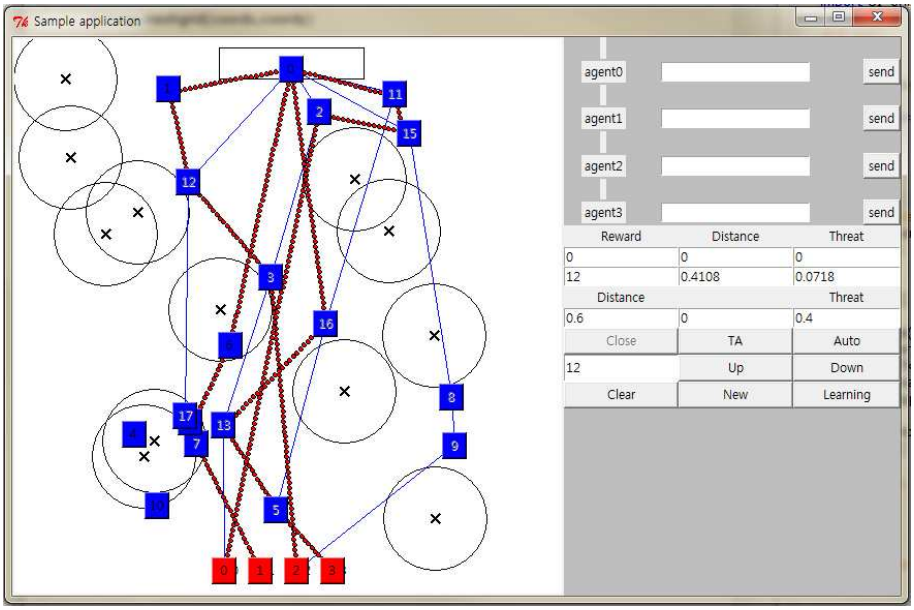
$$\mathcal{W}(\mathcal{D}, \mathcal{T}) = \sum_i^M \mathfrak{w}_i \mathcal{W}(D_i, T_i) \tag{3.22}$$

$$\mathcal{D}, \mathcal{T} = \{D_0, D_1, ..., D_M\}, \{T_0, T_1, ..., T_M\}$$

where $p(\mathbf{x}_i|\mathcal{X})$ is a conditional probability of test datum $\mathbf{x}_i$ in training data set $\mathcal{X}$. These equation puts higher priority on data which are closer to center of distribution and can give improved result rather than assuming every test datum as the same weight.



(a)



(b)

Figure 3.1: (a) Manual task allocation in user interface (solid lines), (b) Automated task allocation in user interface (thick dotted lines)

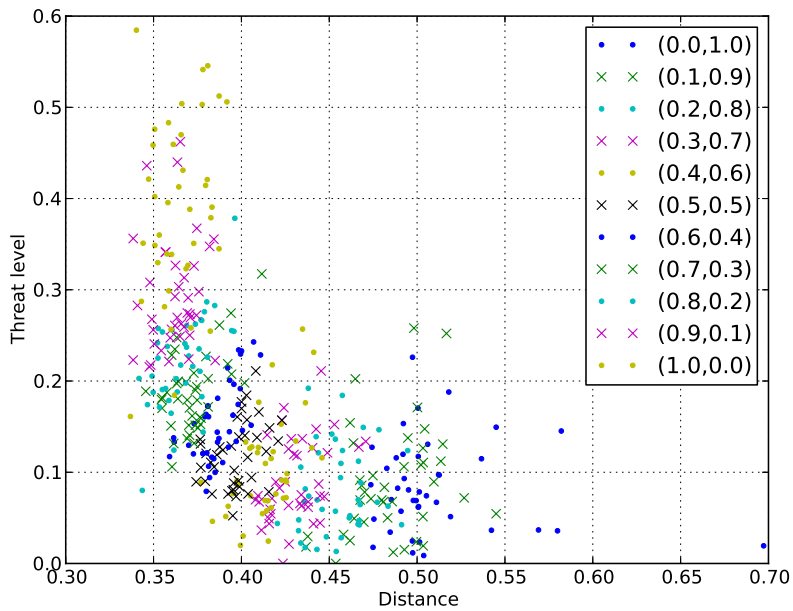


Figure 3.2: Plot of training data in the feature space

4

Results

It is possible to estimate the policy from any allocation results using policy estimation algorithm. Before applying the algorithm to estimate the task allocation of a real human user, the algorithm has to be tested by automated task allocation algorithm. The purpose is to confirm the estimation algorithm is reliable and to check a convergence of the estimation error. The expected policy estimation error can be calculated for a virtual user that uses automated task allocation algorithm. Comparison of policy learning algorithms which are introduced in the previous sections is presented. Using the estimation algorithm, several applications are presented in this section. In the policy estimation test part, an automation of manual task allocation of the user is shown. In the user policy tracking test, the task allocations with time varying policy are carried. User data reliability is an index which shows the consistency of manual task allocations. After the test, user assistance algorithm is presented to improve the reliability of the user.

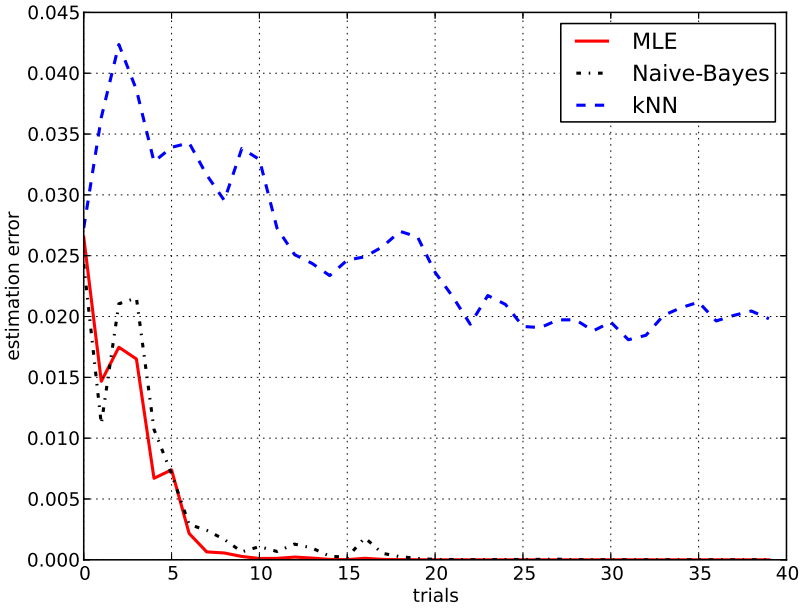


Figure 4.1: Plot of estimation errors as the results of task allocations are accumulated

4.1 Convergence of policy estimation error

The policy estimation for one data of task allocation may make some errors but as the results of task allocation of the users are accumulated, the algorithm can correctly estimate the policy of the user. It means that as the user keeps performing task allocation with consistent policy, the estimation of the policy becomes more precise. We assumed a virtual user which uses the automated task allocation algorithm. The virtual user performed task allocations for various scenarios with specific policy and the user policy is estimated with the results. Because the purpose of this test is to verify increasing precision of policy estimation as the user data is accumulated, the estimation algorithm uses accumulated results of task allocation from each trial. The policy estimation error e at nth trial can be

expressed as follows:

$$e = |\mathcal{W}_{user} - \mathcal{W}_{estimated}(\mathcal{D}, \mathcal{T})| \tag{4.1}$$

$$\mathcal{W}_{estimated}(\mathcal{D}, \mathcal{T}) = \sum_{i}^{n} p(\mathcal{W}_i|\mathcal{D}, \mathcal{T})\mathcal{W}_i \tag{4.2}$$

$$\mathcal{D} = \{D_0, D_1, ..., D_n\}$$

$$\mathcal{T} = \{T_0, T_1, ..., T_n\}$$

where $\mathcal{W}_{user}$ is a weight set of the automated task allocations and $\mathcal{W}_{estimated}$ is estimated weight set from policy estimation algorithm. In (4.2), the estimation algorithm is using accumulated results from 1 to n at nth trial.

Fig. 4.1 is a plot of estimation errors for various policy estimation algorithms. From the figure, it is clear that estimation error decreases as the results of task allocations are accumulated but the result from kNN algorithm is comparatively bad. If there occurs a outlier result in task allocation, the estimation error may seem to increase temporarily but it converges to 0 before 10 trials of task allocations. Maximum likelihood estimation was used for after policy learning process in this research.

4.2 Policy estimation test

The purpose of this section is imitating task allocation of the human user. Two human users performed several manual task allocation trials with their own policies. These data were gathered from task allocation interface for randomly generated scenarios. By applying policy learning algorithm to each result of manual task allocations, representative policies the users can be estimated. Fig. 4.2 shows the plots of results in feature spaces. Differences of distribution in feature space result different estimated policies. For example, in Fig. 4.2(a), the user A makes decisions which are not caring about threat avoidance so the results are located on the top left part of feature spaces showing a high threat level. On the other hand, user B performed task allocations which care about both total distance and threat avoidance and the results are distributed widely in the middle of feature space.

Table 4.1: Performance comparison of manual and automated task allocation after policy learning(#50 task allocations).

User	Estimated policy	Method	Total distance	Threat level
A	(0.9,0.1)	Manual	0.3657	0.2728
		Automated	0.3638	0.2676
B	(0.5,0.5)	Manual	0.4079	0.1749
		Automated	0.4078	0.1232

After the policy estimation process, automated task allocation simulation was done with the estimated policy of the users. Fig. 4.3 shows that the automated task allocation algorithm makes two different results depending on the human user for who the algorithm is learning. Fig. 4.3(a) shows task allocation results weighted on reducing total distance and Fig. 4.3(b) shows result which are weighted on the threat avoidance.

To verify that the automated task allocation algorithm is well following the user's policy, the automated task allocation simulations are conducted for the same scenarios with manual task allocation. In the Table 4.1, the automated task allocation results with the estimated policy show similar or better results in both scores, total distance and total threat level. As shown in this result, the automated task allocation algorithm can effectively replace the manual task allocation of the human users.

4.3 User policy tracking test

In the previous sections, we introduced a policy estimation algorithm from the manual task allocations of the users. The purpose of estimating the policies is to perform similar results of task allocations with the users. With the previously accumulated data, the policy of the user can be estimated quickly right after the user allocation. The automated

task allocation is performed with the estimated policy and the user can compare two task allocation results. The policy of the user is estimated in every task allocation trial so this algorithm can deal with changing of user's policy. To verify this, we assumed a virtual user which autonomously allocates task with its own estimated policy. With this method we can figure out the performance of the policy estimation algorithm. The performance can be written as follows:

$$P_{user}(D,T) = w_{user,1}\ D + w_{user,2}\ T \tag{4.3}$$

This equations means the task allocation result (D,T) is measured by the policy of the user so the result which reflects the policy will give best performance. We propose a policy tracking performance using (4.3) which can be written as

$$P_t(d,t) = \frac{P_{user}(D,T)}{P_{user}(D_{user},T_{user})} \tag{4.4}$$

$$= \frac{w_{user,1}\ D + w_{user,2}\ T}{w_{user,1}\ D_{user} + w_{user,2}\ T_{user}} \tag{4.5}$$

With (4.5), any result of task allocation can be compared to the task allocation of the virtual user. If the result is appropriate to the policy of the virtual user, the value P_t is near 1. The value can exceed 1 because there may exist a better solution of task allocation than the virtual user. The virtual user selects 1 of 6 weight models.

$$\mathcal{W}_{user} \in \{(0.0,1.0),(0.2,0.8),...,(1.0,0.0)\}$$

We assume an autonomous algorithm which allocates tasks according to the estimated policy of the virtual user. For comparison, another autonomous task allocation algorithm uses fixed weight set. The policies of the algorithms are expressed as

$$\mathcal{W}_{estimated}(D_{user},T_{user}) = \sum_{i}^{n} p(\mathcal{W}_i|D_{user},T_{user})W_i \tag{4.6}$$

$$\mathcal{W}_{fixed} = (0.5,0.5)$$

$$\mathcal{D},\mathcal{T} = \{D_0,D_1,...,D_n\},\{T_0,T_1,...,T_n\}$$

The simulated result are plotted in Fig. 4.4. In Fig. 4.4(a), there are performance losses in fixed policy case when the policy of the virtual user is far from fixed policy. However, the estimation algorithm gives reasonable performance for the various polices of the user.

4.4 User data reliability test

We define user data reliability index in this section. User reliability represents how reliable the task allocation of the user is. If the user is generating reliable allocation results, the estimated policy of the user task allocation will be consistent. Because we showed the precision of the user policy estimation algorithm in the previous section, it is reasonable to evaluate the reliability of the human user with the estimation algorithm. For comparison, autonomous task allocations were conducted with the same estimated policy of the human user because basically automated task allocation algorithm is expected to give more reliable result than human users. First, the results of human task allocation were gathered with the task allocation interface and as a result, Fig. 4.5 is a plot of demonstrations of the user. If the results of task allocation from the user are distributed widely in the feature space, the results will be estimated as various policies. The reliability is calculated with following equations:

$$E = \frac{1}{n}\sum_{i}^{n} |\mathcal{W}_{estimated}(D_i, T_i) - \mathcal{W}_{estimated}(\mathcal{D}, \mathcal{T})| \tag{4.7}$$

$$\mathcal{D} = \{D_0, D_1, ..., D_n\}$$

$$\mathcal{T} = \{T_0, T_1, ..., T_n\}$$

where n is the number of user task allocation trials. The equation means a sum of estimated policy difference for each trial of user task allocation. $\mathcal{W}_{estimated}(D, T)$ is the policy estimated from all of task allocations of the user so it means the estimated policy of the user and $\mathcal{W}_{estimated}(D_i, T_i)$ is the policy estimated from one task allocation demonstration

Table 4.2: User data reliability

User	Estimated policy	Method	Policy deviation	Reliability
A	(0.9,0.1)	Manual	0.076	0.66
		Automated	0.050	1
B	(0.5,0.5)	Manual	0.161	0.39
		Automated	0.062	1

at ith trial. With the equation, reliable result of task allocations will give small E value because there will be small outlier policies.

The policy deviation E was calculated for both users and automated allocations for each user. As expected from Fig. 4.5, automated task allocations give better reliability than manual allocations giving small policy deviation in Table 4.2. The automated task allocation is conducted with the same estimated policy because the reliability can be evaluated differently as the policies. For example, task allocation with weight set (1.0, 0.0) may be easier to make reliable data than task allocation with weight set (0.5,0.5). To compensate this, the reliability index R is defined as follows:

$$R = E_{automated}/E_{user} \tag{4.8}$$

This index is relative reliability of the user compared to automated task allocation results. The reliability indexes of the automated task allocation results are set to 1 and the reliability of the user was calculated as values below 1. With this kind of index, the reliability of users can be calculated. Resulting reliability indexes are shown in Table 4.2 and user A shows better reliability than user B.

4.5 User assistance algorithm

We are interested in achieving human-robot interaction in task allocation. In the previous section, the differences between user task allocation and automated task allocation are described. The reliability of the human user task allocation was relatively lower than that of automated task allocation. In this section, we present user assistance algorithm for task allocation. After a task allocation demonstration of the user, the assistance algorithm gives several suggestions which are better than the manual allocation of the user. The suggested allocations from user assistance algorithm always have smaller distance and threat level than those of the manual task allocation. Solutions which are showing worse performance at one of the features, distance and threat level, are considered as allocations of different policies so these allocations are inappropriate as suggestions. The point is that every suggested allocation improves the performance of task allocation and by choosing one of the suggestions, the user can make more detailed decision in terms of the policy. For example, if the estimated policy of the user is a weight set of (0.5,0.5), the suggested policies varies from (0.4,0.6) to (0.6,0.4) then by selecting one of them, the user can adjust the policy of task allocation one more time while improving the performance. Several automated task allocations are conducted to make suggestions with polices near the estimated policy of the user. Fig. 4.7 shows the suggested solutions from the automated algorithm as the manual task allocation is conducted. The suggested solutions are located in the left bottom part of manual allocation plot, which means that the distance and threat level scores are smaller than those of manual allocation. The manual allocation and suggestions are performed in the user task allocation interface which was introduced in the previous sections. Using the interface, the human user is provided with information of each suggested solution and can choose one of suggestions(Fig. 4.9).

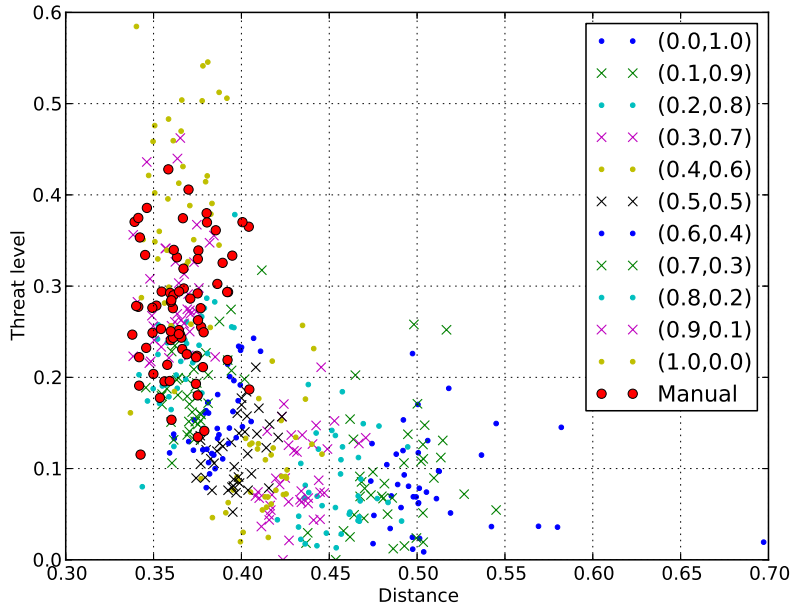
Several trials were conducted by the human user with the assistance algorithm. Improvement by selecting the suggested solution can be expressed as relocation of allocation plot in feature space. In Fig. 4.8, the points which indicate manual allocations relocated

to left bottom region and the distribution became dense. Figs. 4.10 and 4.11 show comparison of two results from the manual task allocations and the modified results from user assistance algorithm. The cost of task allocation is calculated as weighted sum of distance and threat level where the weight set is estimated from manual task allocation result. It is obvious that the cost of task allocation decreases with user assistance algorithm because the suggested solutions provide both smaller distance and threat. In Fig. 4.11, the distribution shows improvement of reliability in task allocation.

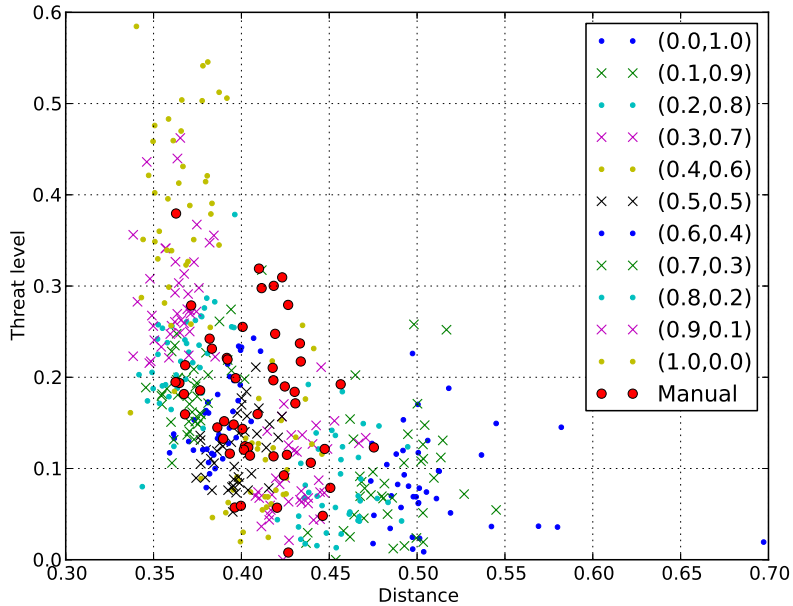
From Table 4.3, the performance and reliability were improved when the user assistance algorithm is used. Because the algorithm uses the manual task allocation demonstration as a reference for searching solutions, it has advantage over a fully automated algorithm which suggests every possible solution to the user. The computational time to make suggestions can be reduced.

Table 4.3: Comparison of manual and supported task allocation

Estimated policy	Method	Policy deviation	Cost
(0.5,0.5)	Manual	0.180	0.30
	Supported	0.040	0.24

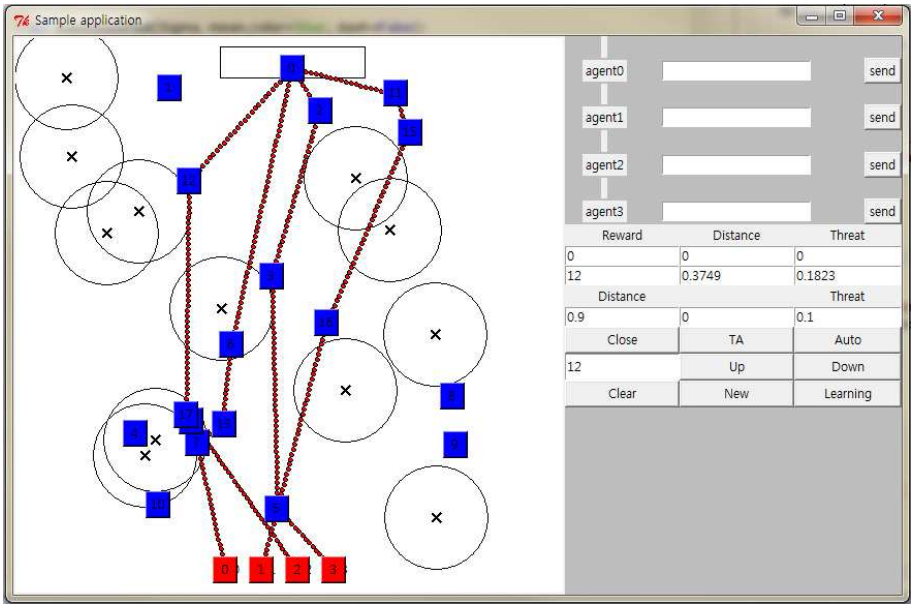


(a)

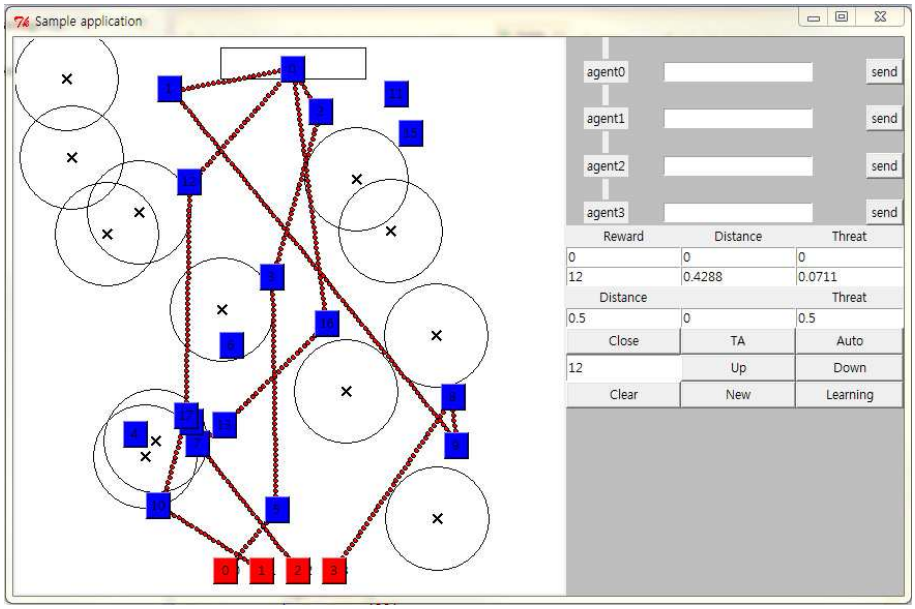


(b)

Figure 4.2: (a) Results of task allocation of User A in feature space, (b) Results of task allocation of User N in feature space

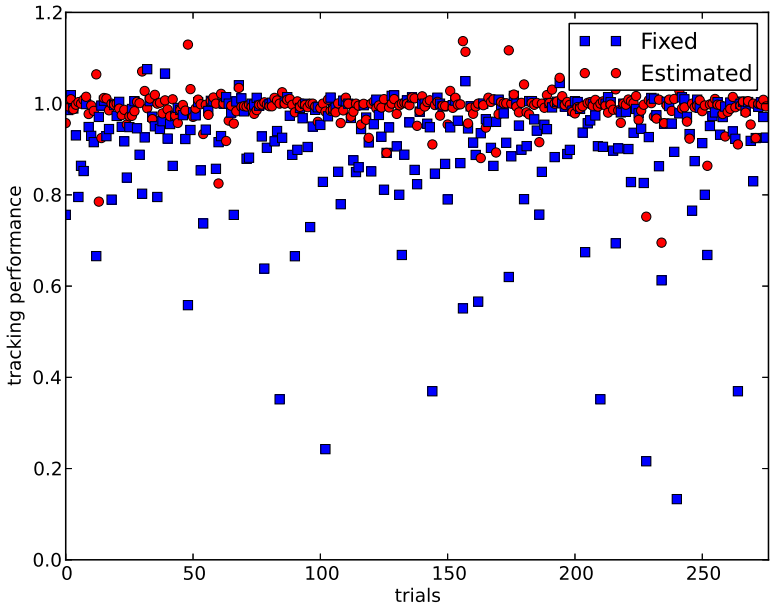


(a)

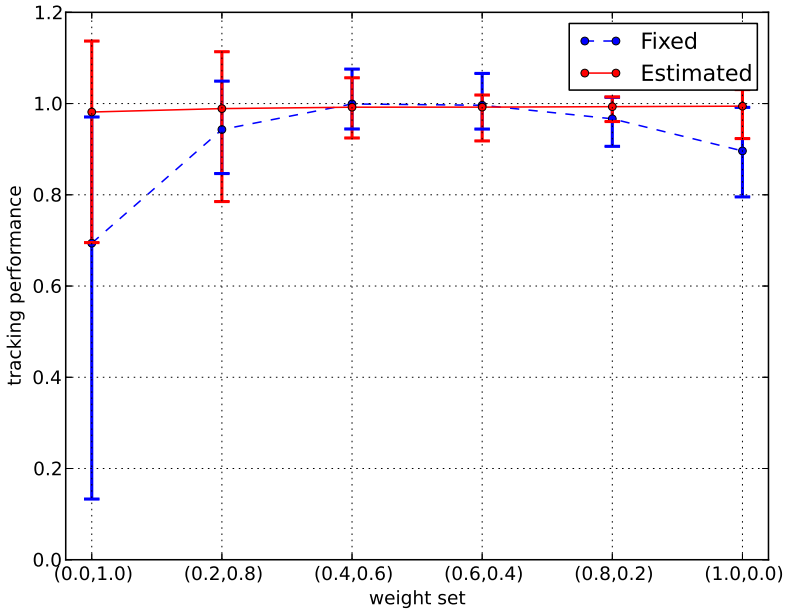


(b)

Figure 4.3: (a) Automated task allocation with learned policy from User A, (b) Automated task allocation with learned policy from User B.

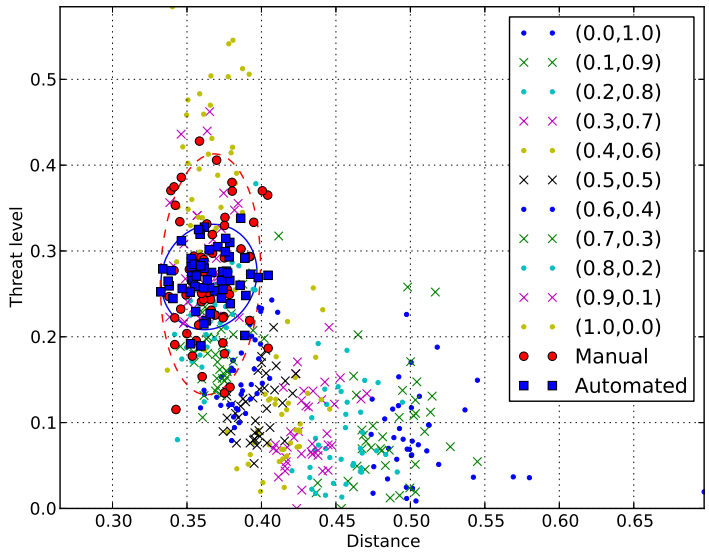


(a)

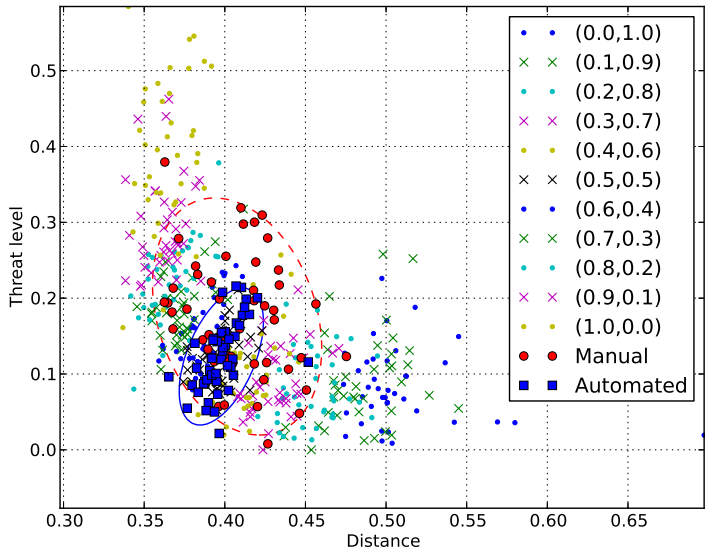


(b)

Figure 4.4: (a) Plot of tracking performance in terms of trials, (b) Plot of tracking performance in terms of weight sets

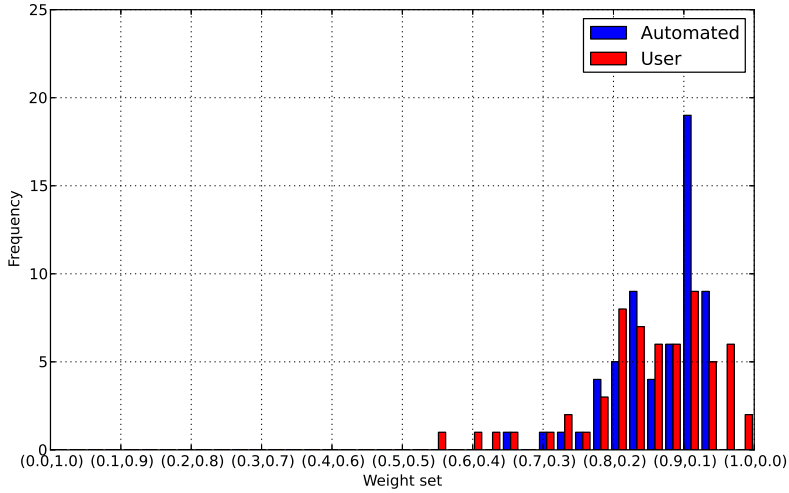


(a)

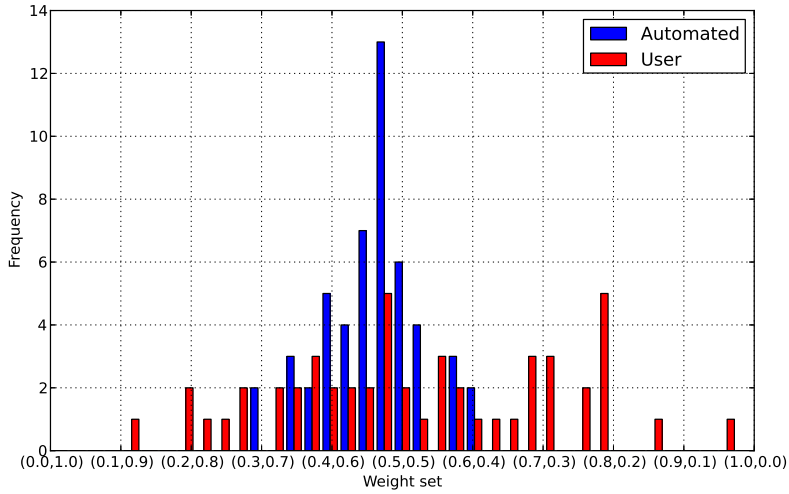


(b)

Figure 4.5: In each plot, automated algorithm(solid contour) shows better distributions than manual allocations(dotted contour).(a) Comparison of task allocation results from User A and automated algorithm(contour is 2σ of distribution), (b) Comparison of task allocation results from User B and automated algorithm(contour is 2σ of distribution)



(a)



(b)

Figure 4.6: (a) Distribution of the estimated policy from User A and automated algorithm, (b) Distribution of the estimated policy from User B and automated algorithm

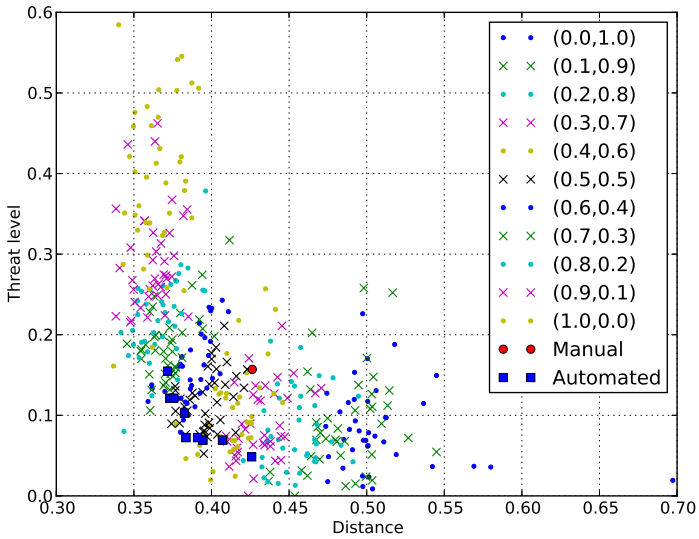


Figure 4.7: Manual task allocation plot(circle) and suggested automated allocations (square)

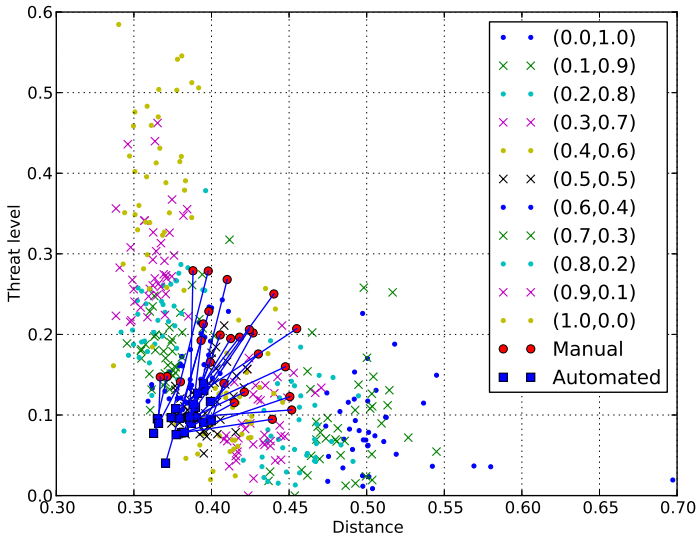
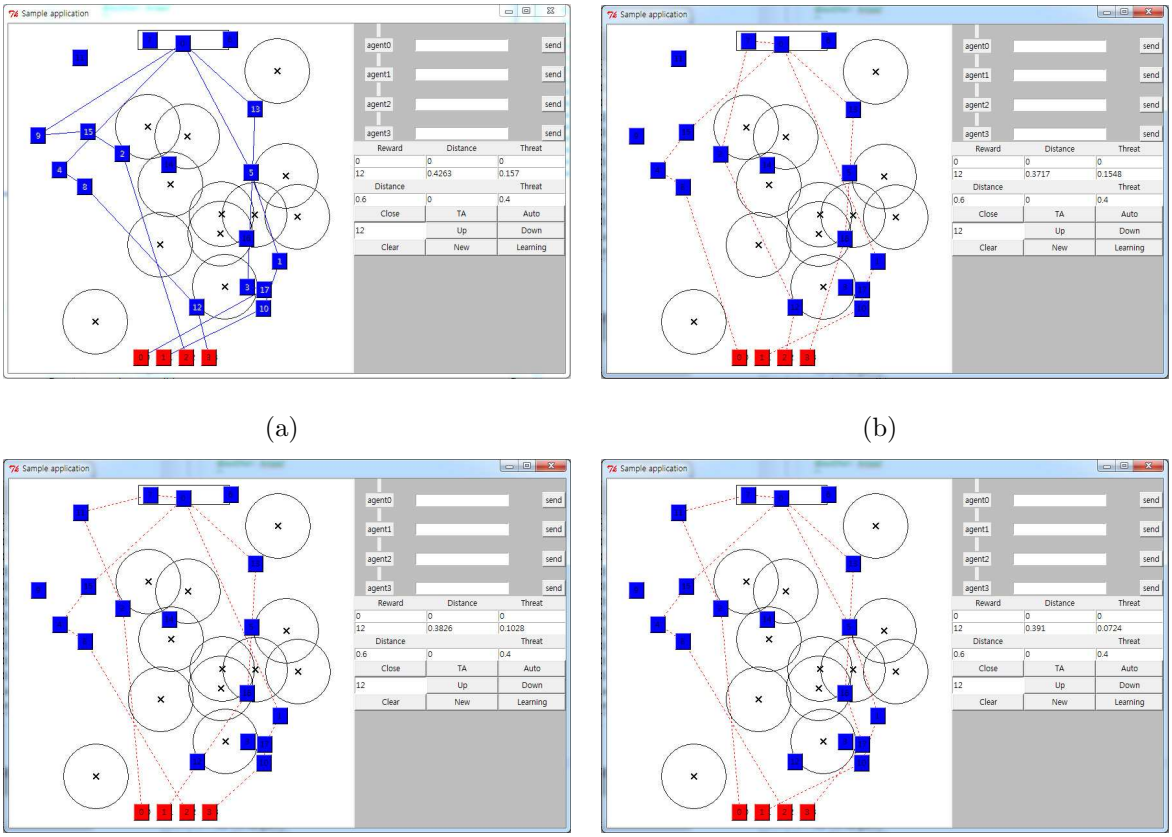


Figure 4.8: Manual task allocation plot(circle) and suggested automated allocations (square)



(a) (b)

(c) (d)

Figure 4.9: (a) Manual task allocation demonstration, (b,c,d) Suggested task allocations by the automated algorithm

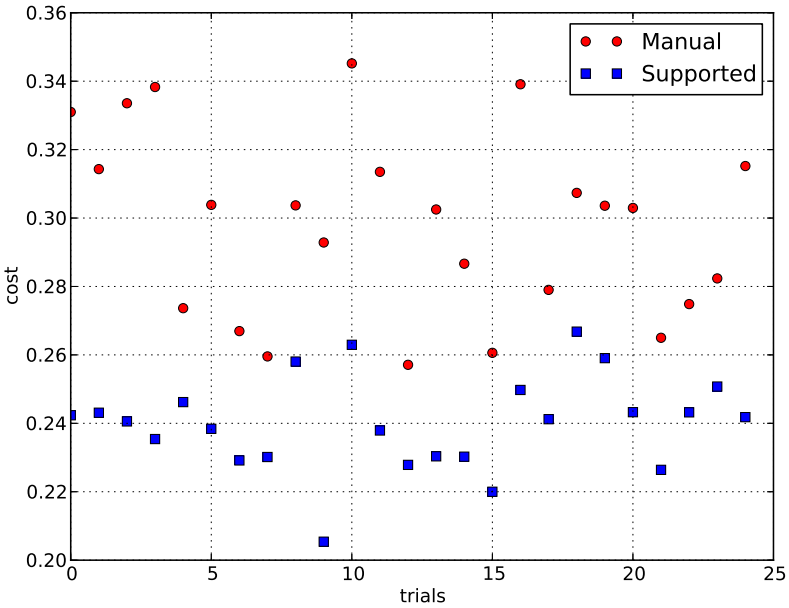


Figure 4.10: Cost plot of manual task and supported task allocation

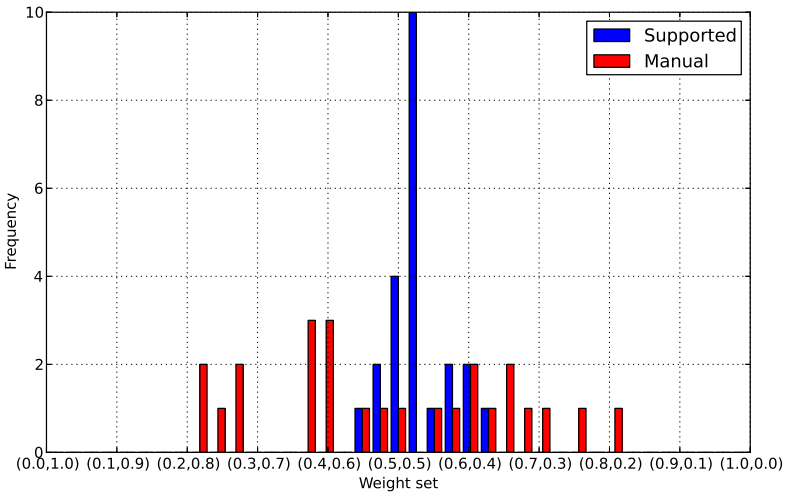


Figure 4.11: Estimated policy distribution of manual task and supported task allocation

5

Conclusions

Although there had been developments in autonomous task allocation algorithms, human users may want to intervene for some reasons. To achieve better human-robot interaction, we introduced quantification and a learning algorithm for task allocation of human users so that the automated allocation result can reflect the intention of the user. The results of user manual allocation are accumulated and using estimation algorithm, the policy of the human user can be figured out. The convergence of estimation error of policy estimation was verified as the data of task allocations are accumulated. With the estimated policy of task allocation from human user, the automated algorithm successfully replaced manual task allocation. The automated task allocation with policy estimation algorithm can conduct task allocation following the time-varying policy of the user while giving reasonable performance. Policy estimation was applied to human users and user reliability test which measures consistency of task allocation policy of the user was conducted. To improve the performance and reliability of the manual task allocation, user assistance algorithm was presented, which suggests several possible solutions for the user. The performance and the reliability of the user were improved using this assistance algorithm.

References

[1] R. W. Beard, T. W. McLain, M. A. Goodrich, E. P. Anderson, Coordinated target assignment and intercept for unmanned air vehicles, Robotics and Automation, IEEE Transactions on 18 (6) (2002) 911–922.

[2] L. Zhong, Q. Luo, D. Wen, S.-d. Qiao, J.-m. Shi, W.-m. Zhang, A task assignment algorithm for multiple aerial vehicles to attack targets with dynamic values, Intelligent Transportation Systems, IEEE Transactions on 14 (1) (2013) 236–248.

[3] A. Boeing, M. Boulton, T. Bräunl, B. Frisch, S. Lopes, A. Morgan, F. Ophelders, S. Pangeni, R. Reid, K. Vinsen, et al., Wambot: Team magician's entry to the multi autonomous ground-robotic international challenge 2010, Journal of Field Robotics 29 (5) (2012) 707–728.

[4] J. Butzke, K. Daniilidis, A. Kushleyev, D. D. Lee, M. Likhachev, C. Phillips, M. Phillips, The university of pennsylvania magic 2010 multi-robot unmanned vehicle system, Journal of Field Robotics 29 (5) (2012) 745–761.

[5] E. Olson, J. Strom, R. Morton, A. Richardson, P. Ranganathan, R. Goeddel, M. Bulic, J. Crossman, B. Marinier, Progress toward multi-robot reconnaissance and the magic 2010 competition, Journal of Field Robotics 29 (5) (2012) 762–792.

[6] B. D. Argall, S. Chernova, M. Veloso, B. Browning, A survey of robot learning from demonstration, Robotics and autonomous systems 57 (5) (2009) 469–483.

[7] T. Matsubara, J. Morimoto, J. Nakanishi, M.-a. Sato, K. Doya, Learning cpg-based biped locomotion with a policy gradient method, Robotics and Autonomous Systems 54 (11) (2006) 911–920.

[8] A. Cherubini, F. Giannone, L. Iocchi, D. Nardi, P. F. Palamara, Policy gradient learning for quadruped soccer robots, Robotics and Autonomous Systems 58 (7) (2010) 872–878.

[9] J. S. McGrew, J. P. How, B. Williams, N. Roy, Air-combat strategy using approximate dynamic programming, Journal of guidance, control, and dynamics 33 (5) (2010) 1641–1654.

[10] S. Chernova, M. Veloso, Confidence-based policy learning from demonstration using gaussian mixture models, in: Proceedings of the 6th international joint conference on Autonomous agents and multiagent systems, ACM, 2007, p. 233.

[11] M. Ollis, W. H. Huang, M. Happold, A bayesian approach to imitation learning for robot navigation, in: Intelligent Robots and Systems, 2007. IROS 2007. IEEE/RSJ International Conference on, IEEE, 2007, pp. 709–714.

[12] V. Huang, H. Huang, S. Thatipamala, C. J. Tomlin, bbc, in: AIAA Guidance, Navigation, and Control (GNC) Conference, 2013.

[13] D. C. Bentivegna, C. G. Atkeson, Learning from observation using primitives, in: Robotics and Automation, 2001. Proceedings 2001 ICRA. IEEE International Conference on, Vol. 2, IEEE, 2001, pp. 1988–1993.

[14] J. Saunders, C. L. Nehaniv, K. Dautenhahn, Teaching robots by moulding behavior and scaffolding the environment, in: Proceedings of the 1st ACM SIGCHI/SIGART conference on Human-robot interaction, ACM, 2006, pp. 118–125.

[15] M. Stolle, C. G. Atkeson, Knowledge transfer using local features, in: Approximate Dynamic Programming and Reinforcement Learning, 2007. ADPRL 2007. IEEE International Symposium on, IEEE, 2007, pp. 26–31.

[16] B. Argall, B. Browning, M. Veloso, Learning by demonstration with critique from a human teacher, in: Proceedings of the ACM/IEEE international conference on Human-robot interaction, ACM, 2007, pp. 57–64.

[17] F. Duvallet, A. Stentz, Imitation learning for task allocation, in: Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on, IEEE, 2010, pp. 3568–3573.

[18] W. Zhu, S. Choi, An auction-based approach with closed-loop bid adjustment to dynamic task allocation in robot teams, in: Proceedings of the world congress on engineering, 2011, pp. 1016–1066.

[19] J. Zhang, Z. Liu, Calculating some inverse linear programming problems, Journal of Computational and Applied Mathematics 72 (2) (1996) 261–273.

[20] T. Bektas, The multiple traveling salesman problem: an overview of formulations and solution procedures, Omega 34 (3) (2006) 209–219.

[21] G. Binetti, D. Naso, B. Turchiano, Decentralized task allocation for surveillance systems with critical tasks, Robotics and Autonomous Systems 61 (12) (2013) 1653–1664.

[22] J. Grefenstette, R. Gopal, B. Rosmaita, D. Van Gucht, Genetic algorithms for the traveling salesman problem, in: Proceedings of the first International Conference on Genetic Algorithms and their Applications, Lawrence Erlbaum, New Jersey (160-168), 1985, pp. 160–168.

[23] Y. Eun, H. Bang, Cooperative task assignment/path planning of multiple unmanned aerial vehicles using genetic algorithm, Journal of aircraft 46 (1) (2009) 338–343.

[24] B. L. Miller, D. E. Goldberg, Genetic algorithms, tournament selection, and the effects of noise, Complex Systems 9 (3) (1995) 193–212.

[25] S. A. Dudani, The distance-weighted k-nearest-neighbor rule, Systems, Man and Cybernetics, IEEE Transactions on (4) (1976) 325–327.

[26] D. C. Bentivegna, C. G. Atkeson, G. Cheng, Learning tasks from observation and practice, Robotics and Autonomous Systems 47 (2) (2004) 163–169.