Proceedings of the 37th Chinese Control Conference
July 25-27, 2018, Wuhan, China

A Multi-model Estimation of Distribution Algorithm for Agent Routing Problem in Multi-point Dynamic Task

Sai Lu[1,2,3], Bin Xin[*,1,2,3], Lihua Dou[1,2,3], Ling Wang[4]

1. School of Automation, Beijing Institute of Technology, Beijing 100081, China
2. Key Laboratory of Intelligent Control and Decision of Complex Systems
3. Beijing Advanced Innovation Center for Intelligent Robots and Systems
4. Department of Automation, Tsinghua University, Beijing 100084, China

E-mail: 2220170505@bit.edu.cn; brucebin@bit.edu.cn; doulihua@bit.edu.cn; wangling@tsinghua.edu.cn

Abstract: The agent routing problem in multi-point dynamic task (ARP-MPDT) is a multi-task routing problem of a mobile agent. In this problem, there are multiple tasks to be carried out in different locations. As time goes on, the state of each task will change nonlinearly. The agent must go to the task points in turn to perform the tasks, and the execution time of each task is related to the state of the task point when the agent arrives at the point. ARP-MPDT is a typical NP-hard optimization problem. In this paper, we establish the nonlinear ARP-MPDT model. A multi-model estimation of distribution algorithm (EDA) employing node histogram models (NHM) and edge histogram models (EHM) in probability modeling is used to solve the ARP-MPDT. The selection ratio of NHM and EHM probability models is adjusted adaptively. Finally, performance of the algorithm for solving the ARP-MPDT problem is verified by the computational experiments.

Key Words: Estimation of distribution algorithm, Multi-model, Routing, Multi-point dynamic task

1 Introduction

The common routing problems are often limited to the shortest routing optimization, such as the traveling salesman problem (TSP) [1] and the vehicle routing problem (VRP) [2]. Currently, there are many studies on the route problems with time windows. In recent research, a multi-agent routing problem called multi-point dynamic aggregation (MPDA) [3] is proposed. In the MPDA task, a number of task points are located in different places and their states change over time [4]. Multiple unnamed vehicles aggregate to these task points and execute the tasks cooperatively to make the states of all the task points change to an expected level. This problem features the dynamic evolution of the task state, the time-sensitive nature, and the distribution of the task points [5]. MPDA focuses on the distributed coordination of multiple agents. However, in this paper, only one agent executes al tasks without coordination, and the optimal solution of this routing problem is expected to be discovered. We call it as agent routing problem for multi-point dynamic task (ARP-MPDT). There is little research in this aspect.

For the ARP-MPDT, it is easy to find that as time goes by, the difficulty or urgency of task execution will continue to increase. Because of time delay, failure of the task may even happen. Task state changes over time, which increases the complexity of optimizing ARP-MPDT. In ARP-MPDT, the total time consists of the execution time which the agent spends in executing each task and the traveling time which the agent spends during its moving from one task point to another.

Estimation of distribution algorithm (EDA) [6] is a common and effective method to solve the combination optimization problem. In [9], the node histogram matrix (NHM) was introduced and it is suitable to solve the quadratic assignment problem (QAP) and the linear ordering problem (LOP) very well. The edge histogram matrix (EHM) was also introduced, and it matches the traveling salesman problem (TSP) and the flow shop scheduling problem (FSSP) better than NHM. In this paper, in view of the coexistence of execution time and traveling time in ARP-MPDT, the EDA employing both NHM and EHM is proposed to solve the new agent routing problem. The selection ratio of the two models is regulated dynamically.

2 Problem Formulation

In order to simplify the model of the problem and highlight the research focus, the actual task model will be simplified and the key parameters are extracted. So, before the various models are established, the following assumptions are made:

A1. All tasks are located on a two-dimensional plane, and the agent moves in the plane.
A2. The capacity of the agent is constant and it moves at a constant speed at any time.
A3. The agent is not allowed to go to other task points before its current task is completed.
A4. The agent has enough energy to execute all the tasks， and moves between any two points in a straight line.
A5. The agent makes the decision at $t=0$ at its initial position and executes the tasks immediately.
A6. The agent is able to finish each task independently, and there is no case of failure of the task.
A7. Assume that each task is completed when its state value

This work was supported in part by the National Natural Science Foundation of China under Grant 61673058, in part by the NSFC-Zhejiang Joint Fund for the Integration of Industrialization and Informatization under Grant U1609214, in part by the Foundation for Innovative Research Groups of the National Natural Science Foundation of China under Grant 61621063.

*Corresponding author: Bin Xin (brucebin@bit.edu.cn).

is lower than a specified threshold rather than zero to avoid infinite execution time.

2.1 Model of ARP-MPDT

Through the analysis of the problem, we can easily extract model parameters about the agent, as follows:

1. The speed of the agent is V;
2. The capability parameter of the agent is β;
3. The initial position of the agent is $P_0=(x_0, y_0)$;

The parameters of the task points are as follows:

1. The number of tasks is n;
2. The serial number of a task is i;
3. The initial state of task point i is S_i;
4. The state growth index of task i is α_i;
5. The position of task point i is $P_i=(x_i, y_i)$;
6. The time to leave the task point i is t_i^r;
7. The time to leave the task point i is t_i^l;
8. The execution time of the agent at task point i is Δt_i;
9. The state threshold indicating task completion is ξ;
10. The distance between task point k and the initial position of the agent is $D_{0,k}$, and the distance between any two task points is $D_{j,k}$:

$$D_{j,k} = \sqrt{(x_j - x_k)^2 + (y_j - y_k)^2} \tag{1}$$

where $j, k = 1, 2, ..., n$.

In order to ensure that each task can be completed by the agent, the following condition must be met:

$$\alpha_i \le \beta \qquad \forall i = 1, 2..., n \tag{2}$$

Before the agent arrives at the task point i, the state of the task point at t changes as follows:

$$S_i(t) = S_i * e^{\alpha_i * t} \tag{3}$$

So, the state of task point i when the agent arrives at it is as follows:

$$S_i(t_i^r) = S_i * e^{\alpha_i * t_i^r} \tag{4}$$

When the agent reaches the task point i, it starts to execute the task. Then, the task point state model changes as follows:

$$S_i(t) = S_i(t_i^r) * e^{(\alpha_i - \beta)*(t - t_i^r)} \tag{5}$$

We can get its execution time at the task point i.

$$\Delta t_i = \frac{\ln(\xi) - \ln(S_i(t_i^r))}{\alpha_i - \beta} \tag{6}$$

The task access sequence is denoted as follow:

$$\mathbf{z} = [z(1), z(2), ..., z(n)] \tag{7}$$

where $z(i) \in \{1, 2, ..., n\}$.

The solution of the problem can be expressed as the following form:

$$T = \{(t_{z(1)}^r, t_{z(1)}^l), (t_{z(2)}^r, t_{z(2)}^l), ..., (t_{z(n)}^r, t_{z(n)}^l)\} \tag{8}$$

Before the agent leaves the task point i, the state of the task point i must be lower than the threshold ξ, so the constraint is as follow:

$$S_i(t_i^l) \le \xi \tag{9}$$

Obviously, the following constraint should be satisfied:

$$t_{z(i)}^r \le t_{z(i)}^l \le t_{z(i+1)}^r \le t_{z(i+1)}^l \tag{10}$$

The objective is to minimize the time to complete all tasks. The optimization model of ARP-MPDT is given as follows:

$$\min \quad f(\mathbf{Z}) = t_{z(n)}^l \qquad s.t. \text{ (9) } and \text{ (10)} \tag{11}$$

This formula means that it is expected to minimize the time when the agent completes the final task.

In the ARP-MPDT problem, both the traveling time and the task execution time affect the quality of routing schemes. If the execution time is neglected, ARP-MPDT will degenerate into a TSP [7]. In this sense, TSP can be regarded as a special case of ARP-MPDT.

2.2 The coding/decoding scheme

According to the aforementioned assumptions, because the agent is not allowed to go to other task points before completing its current task, the feasible solution of ARP-MPDT can be expressed by the access sequence of task points. In the following, the coding scheme will be explained by an example including 6 task points:

The access sequence of all task points is denoted as follows:

$$\mathbf{z} = \{1, 2, 4, 6, 3, 5\} \tag{12}$$

The value of each variable in T can be calculated according to the following formulas:

$$\begin{aligned} t_{z(1)}^r &= 0 \\ t_{z(i)}^l &= t_{z(i)}^r + \Delta t_{z(i)} \\ t_{z(i+1)}^r &= t_{z(i)}^l + \frac{D_{z(i),z(i+1)}}{V} \end{aligned} \tag{13}$$

In this way, the value of the objective function can be calculated for each solution encoded by an access sequence of all task points.

3 Design of the EDA with multiple probability models

3.1 Overview of EDA

EDA describes the distribution information of the superior solutions of the problem by establishing probability models [9], and the probability model is sampled to get a new generation of population. EDA relies on the loop of probabilistic modeling and sampling to discover the best solutions. The basic algorithm procedure is shown as the following [10]:

Step 1: Initialize a population of solutions(individuals);
Step 2: Calculate fitness value of every individual, and select some individuals which has better fitness value into the dominant group;
Step 3: Use the dominant group as samples to update the probability model;
Step 4: Sample the probability model to generate new solutions;
Step 5: Judge the termination condition. If the termination condition is satisfied, the algorithm stops and the result is output; otherwise, go to step 2.

3.2 Details of the proposed algorithm

A graph for ARP-MPDT can be built by taking the task points as nodes and using the distance between any two points as the weight of the arc connecting them. It is obvious that the quality of solutions to ARP-MPDT relies on both the states of nodes and the weights of arcs in the graph. In this sense, neither NHM nor EHM can solely adapt to ARP-MPDT in general cases.

When the execution time takes up most of the total time, the agent's traveling time is not a primary element. It is better to solve the ARP-MPDT with EDA employing NHM, because the execution time mainly depends on the property of task points(nodes). On the other hand, if the traveling time takes up most of the total time, it is better to solve the ARP-MPDT with EDA employing EHM. Both NHM and EHM are adopted in the proposed algorithm. However, it is hard to determine which probability model to use without any prior knowledge. A coefficient is designed to adjust dynamically the selection proportion of NHM to EHM in generating new individuals.

Because of the difference between the two models based on NHM or EHM, the sampling strategies and the updating strategies of probability models are also different. A certain proportion of the individuals in new population will be selected as samples by truncation sorting to update both EHM and NHM.

The flowchart of the proposed algorithm for ARP-MPDT is illustrated in Fig.1and the notations are explained as Table1.

Table 1: Notations of the proposed algorithm

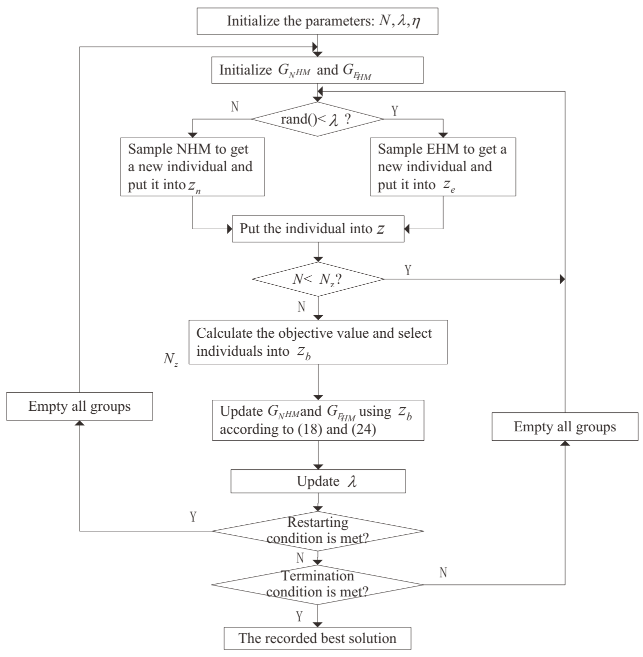
Notation	Implication
G_{NHM}	Node histogram matrix(NHM);
G_{EHM}	Edge histogram matrix(EHM);
λ	The selection ratio coefficient of NHM to EHM;
Z	Population containing individuals generated using NHM and EHM;
N	The population size;
N_Z	The number of individuals generated by the probability models;
η	Proportion of samples to population;
Z_b	The group of samples for updating NHM and EHM;
N_b	The number of individuals in z_b
Z_e	The group in which individuals are generated using EHM;
Z_n	The group in which individuals are generated using NHM;
$g_{\max}$	The maximum number of iterations
g	The identifier of iteration

A. Probability model dominated by execution time

1) Initialization

In this paper, the initial G_{NHM} is related to the state growth index of each task. It can be initialized as follows:

$$G_{NHM}=[G_{NHM}(i,j)]_{n\times n}=\begin{bmatrix}\alpha_1 & \alpha_2 & \cdots & \alpha_n\\ \alpha_1 & \alpha_2 & \cdots & \alpha_n\\ \vdots & \vdots & \ddots & \vdots\\ \alpha_1 & \alpha_2 & \cdots & \alpha_n\end{bmatrix} \tag{14}$$



Initialize the parameters: N,λ,η
Initialize G_{NHM} and G_{EHM}
N
rand()< λ ?
Y
Sample NHM to get a new individual and put it into z_n
Sample EHM to get a new individual and put it into z_e
Put the individual into z
N< N_z?
Y
N
Calculate the objective value and select individuals into z_b
N_z
Empty all groups
Update $G_{NHM}$and G_{EHM} using z_b according to (18) and (24)
Empty all groups
Update λ
Y
Restarting condition is met?
N
Termination condition is met?
N
Y
The recorded best solution

Fig.1: Flowchart of the proposed algorithm

2) Updating NHM

It is necessary to introduce F_n as the frequency matrix to save frequency data of Z_b. F_n can be initialized as follows:

$$F_n=[F_n(i,j)]_{n\times n}=\begin{bmatrix}0 & 0 & \cdots & 0\\ 0 & 0 & \cdots & 0\\ \vdots & \vdots & \ddots & \vdots\\ 0 & 0 & \cdots & 0\end{bmatrix}. \tag{15}$$

Then, $I_n^k(i,j)$ is the indicator function corresponding to the k-th individual in Z_b:

$$I_n^k(i,j)=\begin{cases}1, \text{ if task } i \text{ is the } j\text{-}th \text{ task finished in } k\text{-}th \text{ individual in } Z_b;\\ 0, \text{ otherwise.}\end{cases} \tag{16}$$

The procedure of statistics is showed in (17):

$$F_n(i,j)=\sum_{k=1}^{N_b} I_n^k(i,j). \tag{17}$$

The formula of updating G_{NHM} is as follows:

$$G_{NHM}(i,j)=(1-\mu_n)\cdot G_{NHM}(i,j)+\mu_n\cdot\frac{F_n(i,j)}{N_b} \tag{18}$$

where $\mu_n\in(0,1)$ denotes the learning rate and $i,j=1,2,...,n.$

3) Sampling method

The access sequence of each individual is obtained by the roulette wheel. The procedure of generating a new individual based on G_{NHM} is as follows:

Step 1: Set i=1;

Step 2: Select task point k according to the elements in the line i of G_{NHM} by the roulette wheel. Let $\mathbf{z}(i)=k$;

Step 3: Set all the elements of the column k in G_{NHM} to zero;

Step 4: Let $i=i+1$ and repeat Steps 2 and 3 until all of the task points are selected.

B. Probability model dominated by traveling time

1) Initialization

In this case, the traveling time dominates the total time and EHM suits this case. G_{EHM} can be initialized as follows:

$$G_{EHM}=[G_{EHM}(i,j)]_{n\times n}=\begin{bmatrix}0 & 1 & \cdots & 1\\ 1 & 0 & \ddots & \vdots\\ \vdots & \ddots & 0 & 1\\ 1 & \cdots & 1 & 0\end{bmatrix} \tag{19}$$

2) Updating EHM

Introduce F_e as the frequency matrix to save frequency information of Z_b. F_e can be initialized as follows:

$$F_e=[F_e(i,j)]_{n\times n}=\begin{bmatrix}0 & 0 & \cdots & 0\\ 0 & 0 & \cdots & 0\\ \vdots & \vdots & \ddots & \vdots\\ 0 & 0 & \cdots & 0\end{bmatrix}. \tag{20}$$

Then, $I_e^k(i,j)$ is the indicator function corresponding to the *k-th* individual in Z_b:

$$I_e^k(i,j)=\begin{cases}1, \text{ if task } i \text{ is before task } j \text{ in } k\text{-}th \text{ individual in } Z_b;\\ 0, \text{ otherwise.}\end{cases} \tag{21}$$

Because of the effect of the execution time, ARP-MPDT is no longer a symmetric routing problem. It is necessary to make an adaptable change in the updating strategy. Then, an offset is introduced as follows:

$$\Delta F_e(i,j)=\frac{K_1\cdot\alpha_j}{\beta\cdot V} \tag{22}$$

where $\Delta F_e(i,j)\in[0,1)$ is an offset related to partial elements of the problem, and K_1 is a constant coefficient to ensure that $\Delta F_e(i,j)\in[0,1)$.

The procedure of statistics is showed in (23):

$$F_e(i,j)=\sum_{k=1}^{N_b}\{I_e^k(i,j)(1+\Delta F_e(i,j))+I_e^k(j,i)(1-\Delta F_e(j,i))\}. \tag{23}$$

The formula of updating G_{EHM} is as follows:

$$G_{EHM}(i,j)=(1-\mu_e)\cdot G_{EHM}(i,j)+\mu_e\cdot K_2\cdot\frac{F_e(i,j)}{D(i,j)\cdot N_b} \tag{24}$$

where $\mu_e\in(0,1)$ denotes the learning rate; K_2 is a constant coefficient to offset the influence of the order of magnitude of $D(i,j)$ and $i,j=1,2,...,n.$

3) Sampling method

The roulette wheel is used to sample the probability model to get the access sequence of each individual. The procedure of generating a new individual based on G_{EHM} is as follows:

Step 1: Set i=1;

Step 2: Select task point k according to the elements in the line i of G_{EHM} by the roulette. Let $\mathbf{z}(i)=k$;

Step 3: Set all the elements of the column k in G_{EHM} to zero;

Step 4: Let $i=k$ and repeat Step 2 and 3 until all of the task points are selected.

C. Updating selection ratio coefficient

The coefficient $\lambda(g)$ denotes the selection ratio of NHM to EHM in the *g-th* iteration. This updating method assumes that $\lambda(0)=0.5$. It means that without the prior knowledge, the chance of selecting NHM and EHM is equal. The updating rule is showed as follows:

$$\lambda(g+1)=(1-\mu_\lambda)\cdot\lambda(g)+\mu_\lambda\cdot\frac{N_n}{N_b} \tag{25}$$

where μ_λ denotes the learning ratio and N_n is the number of individuals in Z_n.

4 Computational experiments

In this section, seven instances are designed. The mainly parameters of each instance are showed in Table 2.

The other parameters which need to be set before optimizing are showed in (26).

Table 2: Mainly parameters of each instance

No.	n	(x_i, y_i)(m)	S_i	α_i	V (m/min)	β
1	4	[0,500]	[0,10]	[0,0.2]	50	1
2	8	[0,500]	[0,10]	0	50	1
3	8			[0,0.2]	500	
4	8				50	
5	30	[0,500]	[0,10]	[0,0.2]	50	1
6	100	[0,500]	[0,10]	[0,0.015]	100	1
7	200	[0,500]	[0,10]	[0,0.015]	100	2

*[a,b]: the data is generated randomly in the range [a,b].

$$\begin{cases}\eta=0.5\\ \mu_e,\mu_n,\mu_\lambda=0.1\\ \lambda(0)=0.5\\ N=10\cdot n\\ g_{\max}=300\\ K_1=K_2=100\end{cases} \tag{26}$$

A restarting mechanism is designed which will initialize EHM and NHM according to (14) and (19) when all the individuals in the population are exactly the same as each other.

In this paper, all the algorithms run 20 times on a computer configured as Inter(R) Xeon(R) CPU E5-2620 v4, 32GB RAM, windows 7 operation system.

4.1 An illustrative example

This example with four task points is used to explaining the change of mainly elements in optimizing and the change of the state of each task point in the form of the illustrations.

Four subgraphs can be drawn in Fig. 2.

In Fig.2(a), it shows the access sequence of the task points and routing information. The agent starts from the initial coordinates and visits every task point successively.

Fig.2(b) shows that the tendency of selection ratio coefficient with iteration. Finally, the coefficient tends to 0. That means all individuals of the population are generated from G_{NHM}.

In Fig.2(c), it shows the convergence of the best objective value. Because of the restarting mechanism, the average of the objective value features periodical oscillations above the average of the objective value.

In Fig.2(d), the state of each task point is displayed. Before the agent arrives at it, the state increases non-linearly. After the task is executed, the state of the task decreases fast until it falls to ξ.

4.2 Comparative experiments

Three instances with 8 task points are used to show the difference of the best solutions under different cases.

The three cases are as follows:

Case 1: The speed of the agent is small or the rate of state growth is very small, and the execution time is short. ARP-MPDT is degenerated into TSP in this case.

Case 2: The speed of the agent is relatively large. The execution time is much longer than traveling time.

Case 3: This is a general ARP-MPDT. The traveling time has no obvious dominant relationship with the execution time.

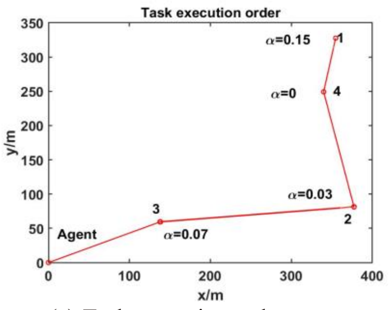
Comparative diagrams of different cases are shown in Fig.3. The task orders of the solutions in all cases above are in the first column. The changes of the selection ratio are showed in the second column. The convergences of the objective values are showed in the third column in Fig.3.

The solution obtained by the algorithm is optimal, which has been proved by enumeration.

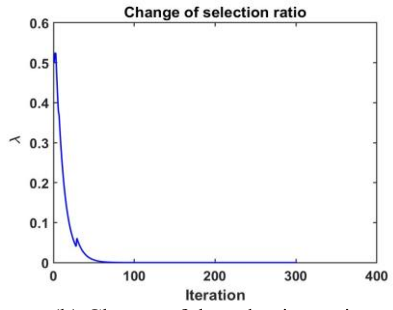
In Case 1, because the execution time of the agent is short, the traveling time takes up most of the total time. So, it can be found that the best solution is more inclined to the solution which has shorter traveling time. In Fig.3(b), the selection ratio of the probability models tends to 1 which means that finally the majority of the solutions are generated based on G_{EHM}. In this case, ARP-MPDT is degenerated into TSP.

In Case 2, because the speed of the agent is large enough, its traveling time between task points is not dominant, so the algorithm will tend to give priority to the task points with more urgent task.

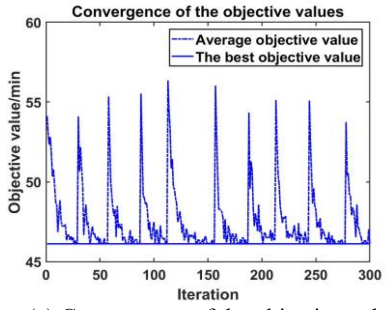
In Case 3, according to Fig.3(g), the algorithm takes into account two kinds of probability models. Finally, it does not choose the solution with the shortest path length, and also does not choose the optimal solution in Case 2. However, the solution of Case 3 obtained by the algorithm is optimal.



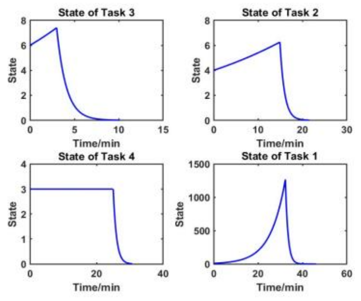
(a) Task execution order



(b) Change of the selection ratio

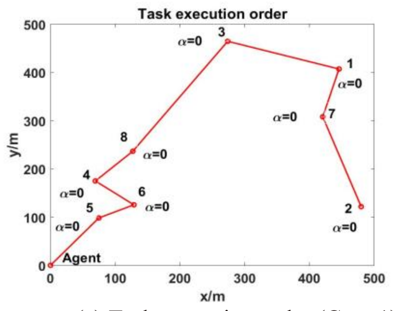


(c) Convergence of the objective values

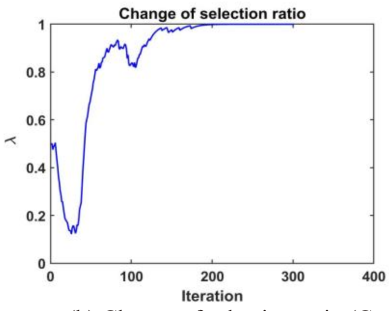


(d) State of each task point

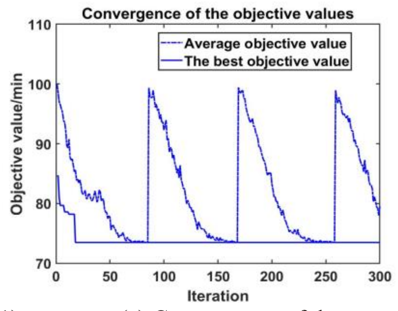
Fig.2: Result of algorithm optimization for instance with 4 tasks



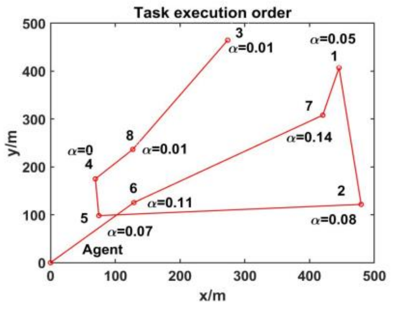
(a) Task execution order (Case 1)



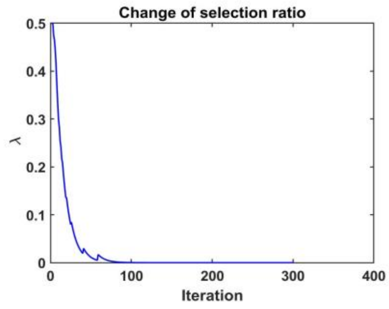
(b) Change of selection ratio (Case 1)



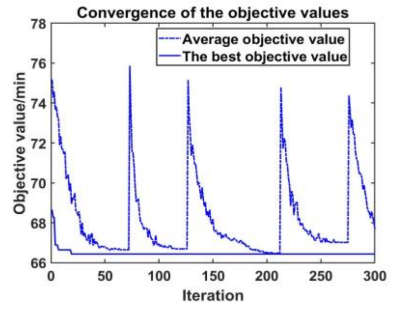
(c) Convergence of the objective values (Case 1)



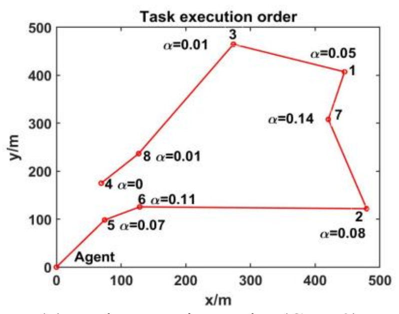
(d) Task execution order (Case 2)



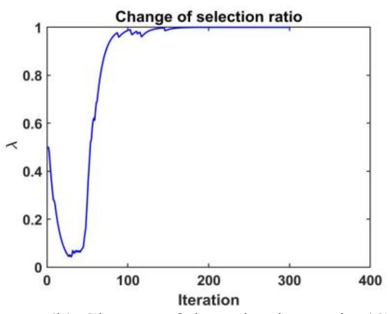
(e) Change of selection ratio (Case 2)



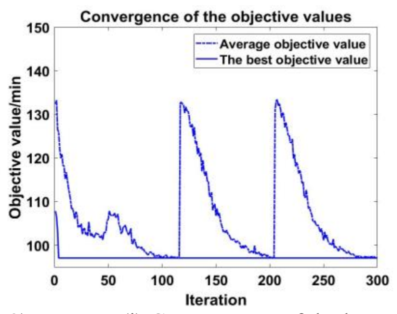
(f) Convergence of the objective values (Case 2)



(g) Task execution order (Case 3)



(h) Change of the selection ratio (Case 3)



(i) Convergence of the best objective values (Case 3)

Fig.3: Comparative diagram of different cases

4.3 General tests

In this experiment, record and analysis of data are carried out. The objective values of the solutions of seven instances are showed in Table 3. This table includes six parameters, such as the best value, the worst value and so on.

In this table, it can be discovered that the solutions are stable for different instances.

The cost of the calculating time is also shown in Table 3. With the increase of the number of tasks, the calculating time becomes much longer and even unacceptable.

Table 3: Statistical results

No.	n	optimal value(min.)	objective value(min.)			runtime (sec.)
			best value	worst value	average ± std	average ± std
1	4	46.12	46.12	46.12	46.12 ± 0	0.58 ± 0.039
2	8	97.09	97.09	97.09	97.09 ± 0	1.50 ± 0.055
3	8	66.43	66.43	66.64	66.47 ± 0.075	1.46 ± 0.007
4	8	73.48	73.48	73.48	73.48 ± 0	1.49 ± 0.037
5	30	n.a.	544.77	546.90	545.57 ± 0.61	13.82 ± 0.153
6	100	n.a.	934.97	946.17	942.97 ± 3.14	153.40 ± 3.033
7	200	n.a.	986.69	994.45	990.84 ± 2.08	719.20 ± 2.749

*n.a.: not available due to the limit of runtime.

5 Conclusion

In this paper, a new routing problem, ARP-MPDT which depends on task execution time and traveling time was introduced. In ARP-MPDT, the state of each task point changes non-linearly. Then, a novel EDA employing both NHM and EHM in probability modeling was proposed to solve ARP-MPDT. A coefficient was designed to adjust dynamically the selection ratio of NHM to EHM in sampling. Through the comparative experiments and general tests, the effectiveness and stability of the proposed algorithm for ARP-MPDT were proved.

Reference

[1] S. M. Chen and C. Y. Chen, Solving the traveling salesman problem based on the genetic simulated annealing ant colony system with particle swarm optimization techniques, *Expert Systems with Applications*, 38(12): 14439-14450, 2011.

[2] J. Tang, Y. Ma and J. Guan, A max-min ant system for the split delivery weighted vehicle routing problem, *Expert Systems with Applications*, 40(18): 7468-7477, 2013.

[3] X. C. Lin and C. G. Cassandras, An optimal control approach to the multi-agent persistent monitoring problem in two-dimensional spaces, *IEEE Transactions on Automatic Control*, 60(6): 1659-1664, 2015.

[4] B. Xin, G. Q. Gao, Y. L. Ding, Y. G. Zhu and H. Fang, Distributed multi-robot motion planning for cooperative multi-area coverage, in *13th IEEE International Conference on Control Automation(ICCA)*, 2017: 361-366.

[5] B. Xin, Y. G. Zhu, Y. L. Ding and G. Q. Gao, Coordinated motion planning of multiple robots in multi-point dynamic aggregation task, in *12th IEEE International Conference on Control and Automation(ICCA)*, 2016: 933-938.

[6] S. D. Zhou and Z. Q. Sun, Survey on estimation of distribution algorithm, *Acta Automatica Sinica*, 33(2): 113-124, 2007.

[7] L. L. Wang and Q. B. Zhu, An efficient approach for solving TSP: The rapidly convergent ant colony algorithm, in *Proceedings of the 4th International Conference on Natural Computation(ICNC)*, 2008: 448-452.

[8] H. C. B. de. Oliveira and G. C. Vasconcelos, A hybrid search method for the vehicle routing problem with time windows, *Annals of Operations Research*, 180(1): 125-144, 2010.

[9] J. Ceberio, E. Irurozki, A. Mendiburu and J. A. Lozano, A review on estimation of distribution algorithm in permutation-based combinatorial optimization problems, *Progress in Artificial Intelligence*, 1(1): 103-117, 2012.

[10] J. Ocenasek, A. Ochoa and M.Soto, Towards a new evolutionary computation: Advances on estimation of distribution algorithms, J. A. Lozano, P. Larranaga, I. Inza, E. Bengoetxea, Eds. Berlin: Springer, 2006.