# Performance Analysis of Evolutionary Computation Based on Tianchi Service Scheduling Problem

Jun YU[1], Yuhao LI[2], Tianwei ZHOU[3] (✉), Churong ZHANG[3], Guanghui YUE[4], and Yunjiao GE[5]

[1] Institute of Science and Technology, Niigata University, Nigata, Japan
https://www.eng.niigata-u.ac.jp/~yujun
[2] ChengDuGuoYiDianZi Co., Ltd., Sichuan, China
[3] College of Management, Shenzhen University, Shenzhen, China
✉ tianwei@szu.edu.cn
[4] School of Biomedical Engineering, Health Science Centre, Shenzhen University, Shenzhen, China
[5] Hubei Aerospace Vehicle Research Institute, Hubei, China

**Abstract.** We choose the well-known evolution strategy (ES) in the evolutionary computation (EC) community to solve the large-scale scheduling problem provided by Alibaba cloud services. Since the problem is accompanied by multiple strong constraints, we design two additional strategies for improving the search efficiency with a given limited computational cost. Compared with widely used numerical benchmark test suits, this problem arises from the requirements of real-world applications and has strict constraints that cannot be violated, such as processing time, response timeout, load balance, and so on. The main contribution of this paper is to establish a bridge between EC algorithms and the characteristics of real-world problems so that EC algorithms can solve real-world problems more effectively and smoothly. Based on the difficulties encountered in the experiment, we summarize some of our experiences and insights, and hope that they may bring new enlightenment to the latecomers.

**Keywords:** Evolutionary computation · Evolution strategy · Scheduling optimization · Large-scale optimization.

## 1 Introduction

Optimization has always been a hot topic, and practitioners are committed to reducing consumption costs while obtaining higher returns. With the continuous growth of customers' personalized demand and the advent of the era of big data, real-world applications arising from industry have become quite complicated, so that many traditional optimization methods, such as linear programming [1] and nonlinear programming [2], are difficult to deal with these emerging large-scale optimization problems. As a new branch of finding the global optimal solution,

evolutionary computation (EC) algorithms have attracted extensive attention [3] and solved many industrial problems successfully [4] thanks to their various advantages, such as robustness, parallelism, and usability.

EC algorithms have similar optimization framework and usually maintain a population composed of multiple individuals (candidate solutions). They borrow the idea of survival of the fittest to improve the quality of individuals and gradually converge to the global optimum. Since the pioneering genetic algorithm [5] started the upsurge of heuristic optimization, various novel EC algorithms have been proposed after decades of development [6]. Among them, many powerful EC algorithms have received lots of attention, e.g. evolution strategy (ES) [7], particle swarm optimization (PSO) [8], differential evolution (DE) [9] and others [10] [11]. Besides, many researchers also focus on how to introduce new strategies into existing EC algorithms to further improve their performance [12] [13]. For example, some use the model of fitness landscape to reduce the number of fitness evaluations [14] and accelerate EC search [15] [16]. Owing to their continuous contribution, EC algorithms show excellent performance on various optimization problems, e.g., multimodal optimization [17], multi-objective optimization [18], and constrained optimization [19].

The main objective of this paper is to establish a bridge between EC algorithms and the characteristics of real-world problems so that EC algorithms can solve real-world problems more effectively and smoothly. Specifically, we try to analyze the match between the EC algorithms' performance and the problem characteristics by using the competition problem derived from real application scenarios, so as to take them away from the laboratory and better serve the industry.

Following this introductory Section, the Tianchi service scheduling problem is described in Section 2, and we present our proposal comprehensively in Section 3. The parameter configuration of our proposal used in the competition is given in Section 4. Although the results submitted to the organizer are not public, we still give a detailed analysis of our proposal and offer several potential topics in Section 5, and Section 6 summarizes our work.

## 2 Tianchi Service Scheduling Problem

As one of the most important global cloud service providers, Alibaba cloud provides full-cycle technical services for numerous enterprises, government agencies and developers. Every day, a large number of new technical problems (tasks) from customers are submitted to the service system and need to be assigned to technical experts for processing. Due to the rapid increase in demand, traditional rule-based scheduling schemes cannot meet the growing demand and may cause some new contradictions, such as uneven dispatch and continuous dispatch. Thus, it is necessary to rely on new scheduling algorithms that can meet the large-scale needs of customers and balance various factors well [20]. This is also why the competition is held.

Before explaining the competition problem, we first give some definitions to avoid confusion in the subsequent description.

**Task:** a task refers to a problem to be solved raised by a customer. The system will produce a task when the customer submits a problem to the cloud service system.

**Expert:** the person who handles the tasks produced by the system.

**Problem classification:** the category of a problem is specified by a customer when submitting the task.

**Skill set:** the corresponding relationship between the time required to solve different problem classification and the experts.

The competition problem can be described as follows. There are $I$ experts and $J$ tasks, each task belongs to only one problem classification and each expert has his own dedicated skill set to show his field of expertise. All unfinished tasks are needed to be assigned to experts and must meet the constraints mentioned below. The organizer (Alibaba cloud service) uses three factors, i.e., (1) average response timeout $\bar{M}$, (2) average processing efficiency $\bar{R}$, and (3) standard deviation of experts' working time $\sigma_L$, to evaluate the submitted solutions. Here, the Eq. (1) is used to calculate the score of a submitted solution in the preliminary stage. The higher the score, the better the performance.

$$score = \frac{c \times \bar{R}}{a \times \bar{M} + b \times \sigma_L} \tag{1}$$

where $a$, $b$, and $c$ are constants, and they are set to 3, 2, and 3000, respectively.

When a task is submitted to the system, the timestamp is marked as the generation time. The period from the generation time of a task to the first processed timestamp is called the response time, and the period from the start timestamp of processing to the end timestamp of the processing is called the processing time. To ensure the customers' service experience, each task has a maximum response time limit $T$, that is, a task needs to be processed within the time $T$ after it is generated, otherwise it will be regarded as a service response timeout. Besides, the processing time for an expert to complete a task depends on his skill set, i.e., the processing time of an expert for different tasks is different.

A task that is being processed but not completed can be reassigned to another expert, but the total number of task transfer is no more than five. Once a task is transferred, the task must stay at least one time unit (one minute) in the assigned expert, i.e., a task can only be assigned once within a minute, and the previous processing progress is cleared and needs to start all over again. Naturally, a completed task will not be reassigned again regardless of whether the maximum number of allocations is reached. Besides, each expert is not allowed to handle more than 3 tasks at the same time, and assumes that each task will be processed immediately after it is assigned to an expert.

## 3 Proposed Solution

Although a variety of powerful EC algorithms have been proposed, we choose ES as the baseline algorithm on account of the better local search ability, and propose two mutation methods to further customize our proposal. Besides, we also design corresponding mechanisms to deal with the above constraints so that our proposal can match the nature of the problem well.

The conventional ES algorithm mainly consists of mutation and selection operations. The mutation operation uses normally distributed random vectors to perturb $\mu$ parent individuals and then generate new $\lambda$ offspring individuals. Based on the fitness rankings, top $\mu$ individuals are selected to the next generation from $\lambda$ offspring individuals or the mixed group of $\mu$ parent individuals and $\lambda$ offspring individuals. The above two operations are repeated to gradually optimize candidate solutions until a termination condition is satisfied. Although a large number of literatures show that the conventional ES plays an important role in real number optimization, the competition problem can be classified as a large-scale discrete problem. We thus propose the following two new mutation methods to replace the original mutation operation, but retain the original greedy selection operation.

**Random variation:** randomly select $k$ genes and change their values.
**Random exchange:** randomly select $k$ genes and exchange their values.

where $k$ is a predetermined constant to determine the number of modified genes. Fig. 1 is an example to show the process of the proposed two mutations.

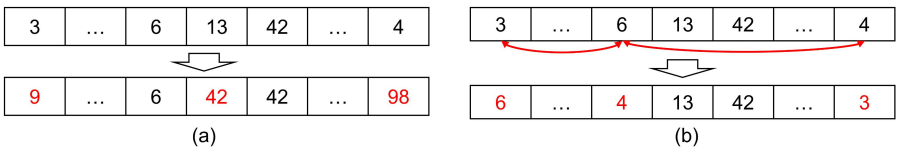


**Fig. 1.** Subgraphs (a) and (b) show the process of random variation and random exchange, respectively. The red parts indicate where the genes have changed.

We virtualize a real expert into three fictitious experts to solve the constraint that an expert cannot handle more than three tasks at the same time. Fig. 2 shows the effect of virtualization experts. Three virtual experts have exactly the same skill set as the original real expert before virtualization, but can only handle one task instead of multiple tasks at the same time. In other words, we transform the original constraint into whether there are free virtual experts to accept tasks.

Since a task can be transferred up to five times between different experts, we use the back-to-front order to assign virtual experts for handling tasks. Specifically, we first decide the last expert to complete the task, and then determine

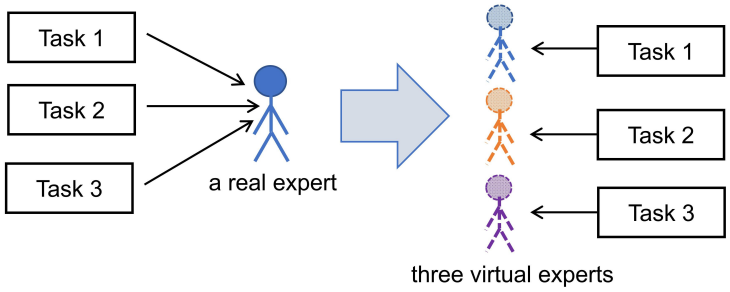


**Fig. 2.** A real expert is virtualized as three virtual experts, and all tasks are handled by virtual experts instead of real experts.

whether it is necessary to assign other experts to take over the task in turn from back to front. Suppose that the assigned sequence of experts dealing with a given task is $A$, $B$, and $C$, which means that the task is handled by three experts in total and experts $A$ and $C$ are the last and the first to deal with the task respectively. First, expert $C$ handles the task but does not complete it, the task is then taken over by expert $B$. Similarly, expert $B$ also does not complete the task and transfers it to expert $A$. Finally, expert $A$ successfully completes the task, and the task is no longer assigned. Although we use a back-to-front allocation method, the actual processing order of tasks is the opposite of the allocation order. Besides, the task transfer is not mandatory, and no more than 5 transfers are allowed.

## 4 Experimental Evaluation

The organizer provides different data set in the preliminary and semifinal, however, only the data set for the preliminary round can be accessed freely, and another undisclosed data set is used in the semifinal. Unfortunately, we cannot know the implementation details of the other competitors, and only know the ranking information of the submitted results. Here, we give a brief introduction to the data set used in the preliminary round. The total number of tasks is 8840, but there are only 107 types of tasks, i.e., a task belongs to only one task type, but a task type may contain multiple different tasks. Compared to the number of tasks, the number of experts is very small and set to 133. We thus can say that the competition problem can be regarded as a large-scale discrete NP-hard problem.

Based on the characteristics of the competition problem, an individual (candidate solution) is encoded into the structure shown in Fig. 3, which contains $5J$ genes. The first $J$ genes represent the sequence of experts who finally completed tasks, and the following four $J$ genes indicate the order of experts who take over transferred tasks according to the back-to-front configuration method. Since the organizer only allows the proposed algorithm to run for four hours and output the optimal solution found, we thus use a two-stage search strategy to

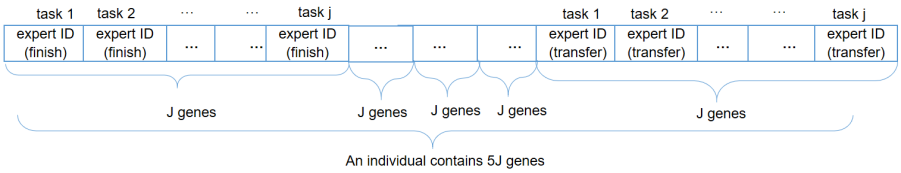


**Fig. 3.** The data structure of an individual. The first $J$ genes record experts who finally completed tasks, and following $4J$ genes record the sequence of experts who take over tasks accord to the back-to-front order assignment. Only when tasks are transferred, the following $4J$ genes will be configured.

determine the sequence of experts to handle tasks. Specifically, we first optimize the experts who are most suitable to handle tasks in the first two hours based on the three factors mentioned above, and then optimize whether tasks need to be transferred in the remaining time to get a better score. Based on the perspective of calculating cost and keeping feasible solutions, we only use one individual and set the number of mutated genes, $k$, to 100.

## 5 Discussions

We start the discussion by analyzing the new benefits of our proposal. Since the competition problem is a large-scale discrete problem with strong constraints, we choose the conventional ES as the baseline algorithm since it has strong local search ability, and customize two new mutation methods to better solve to the competition problem. Generally, changing too many genes at once may easily destroy the feasibility of individuals. We thus set the number of mutated genes, $k$, to a smaller number to maintain the feasibility as much as possible and gradually improve the quality of individuals.

Since constraint processing can improve search efficiency significantly and avoid invalid search in infeasible areas, we design two strategies, i.e., virtualization experts and back-to-front configuration, to deal with constraints. The first strategy, virtualization experts, can simplify the original constraint but keep the exactly same restriction effect. In other words, the constraint is transformed into whether there are available virtual experts to handle tasks, which can reduce the cost of detecting constraint violations and improve operational efficiency.

The other strategies, back-to-front configuration, is to emphasize different optimization indicators at different stages. Here, we are first committed to improving the efficiency of task processing, that is, let suitable experts finish the task as much as possible in the first two hours, then determine whether to add experts between task generation and final processing time to trigger task transfer in the remaining optimization time, which aims to reduce the response time and balance the working time of experts. Fortunately, our proposal won the 16th place among all 1382 participants.

Next, we would like to discuss the potential of our proposal. Not limited to the ES selected as baseline algorithm, we can also choose other EC algorithms

to combine our proposed strategies without drastically changing their original optimization framework. As a first attempt, we fixed all the parameters in the entire search process instead of adjusting them dynamically. Actually, we firmly believe that the same problem may have different characteristics in different optimization periods, and the search strategy should be changed accordingly. Thus, how to use the information collected in evolution to tune parameters in real-time may be a topic worthy of further study.

Different from manually constructed benchmark functions, real-world problems often have various inviolable constraints, and usually cannot use a large number of fitness evaluations to optimize the problem. How to use a small amount of fitness evaluations to obtain satisfactory feasible solutions is an urgent problem that needs to be solved. Since every fitness evaluation may consume high computational costs, we should make full use of the existing individuals even if their fitness is poor. Thus, another potential topic is how to efficiently guide the subsequent search using existing information.

Finally, we want to give some insights about the practicality of EC algorithms. As stated by the no free lunch theorem, no one algorithm can be applied to all problems well. However, once we know the characteristics of the problem to be optimized, we can customize EC algorithms so that they can show stronger performance on the problem. We thus believe that establishing the connection between EC algorithms and real-world problems will be an important means to promote the practicality of EC algorithms.

## 6 Conclusion

We designed two constraint processing strategies and customized the ES algorithm to solve the competition problem provided by Alibaba Cloud. Although our proposal has achieved good results according to the submitted ranking, there is still much room to further improve our proposal.

In future work, we will try to use historical information to extract the characteristics of real-world problems, and use them to improve the performance of EC algorithms.

**Acknowledgments** This work was supported in part by Natural Science Foundation of China under Grant 62001302, in part by Guangdong Basic and Applied Basic Research Foundation under Grant 2019A1515110401, 2019A1515111205, 2021A1515011348, and in part by Natural Science Foundation of Shenzhen under Grant JCYJ20190808145011259, and in part by Shenzhen Science and Technology Program under Grant RCBS20200714114920379.

## References

1. D. F. Votaw: Methods of solving some personnel-classification problems. Psychometrika **17**(3), 255–266 (1952).

2. P. Wolfe: Recent developments in nonlinear programming. Advances in Computers **3**, 155–187 (1962).
3. T. Back, U., Hammel, H. P. Schwefel: Evolutionary computation: Comments on the history and current state. IEEE Transactions on Evolutionary Computation **1**(1), 3–17 (1997).
4. Q.G. Xiao, C.B. Li, Y.Tang, J. Pan, J. Yu, X.Z. Chen: Multi-component energy modeling and optimization for sustainable dry gear hobbing. Energy, **187**, 1–16 (2019).
5. J. H. Holland: Outline for a logical theory of adaptive systems. Journal of the ACM **9**(3), 297–314 (1962).
6. R. Kicinger, T. Arciszewski, K. D. Jong: Evolutionary computation and structural design: A survey of the state-of-the-art. Computers & Structures **83**(23–24), 1943–1978 (2005).
7. H. G. Beyer, H. P. Schwefel: Evolution Strategies: A Comprehensive Introduction. Natural Computing **1**(1), 3–52 (2002).
8. J. Kennedy and R. Eberhart: Particle swarm optimization. IEEE International Conference on Neural Networks, pp. 1942–1948 (1995).
9. R. Storn, K. Price: Differential Evolution: A Simple and Efficient Heuristic for global Optimization over Continuous Spaces. Journal of Global Optimization **11**, 341–359 (1997).
10. K. Dervis, B. Bahriye: A powerful and efficient algorithm for numerical function optimization: Artificial bee colony (ABC) algorithm. Journal of Global Optimization **39**(3), 459–471 (2007).
11. A. H. Gandomi, A. H. Alavi: Krill herd: A new bio-inspired optimization algorithm. Communications in Nonlinear Science and Numerical Simulation **17**(12), 4831–4845 (2012).
12. J. Yu, H. Takagi: Acceleration for Fireworks Algorithm Based on Amplitude Reduction Strategy and Local Optima-based Selection Strategy. 8th International Conference on Swarm Intelligence, pp. 477–484 (2017).
13. J. Yu, Y. Tan, H. Takagi: Accelerating Fireworks Algorithm with an Estimated Convergence Point. 9th International Conference on Swarm Intelligence, pp. 263–272 (2018).
14. Y. Jin, O. Markus, S. Bernhard: A framework for evolutionary optimization with approximate fitness functions. IEEE Transactions on Evolutionary Computation **6**(5), 484–494 (2002).
15. J. Yu, Y. Pei, H. Takagi: Accelerating Evolutionary Computation Using Estimated Convergence Points. IEEE Congress on Evolutionary Computation, pp. 1438–1444 (2016).
16. Y. Pei, J. Yu, H. Takagi: Search Acceleration of Evolutionary Multi-objective Optimization Using an Estimated Convergence Point. Mathematics **7**(2), 129–147 (2019).
17. J. Yu, H. Takagi, Y. Tan: Fireworks Algorithm for Multimodal Optimization Using a Distance-based Exclusive Strategy. IEEE Congress on Evolutionary Computation, pp. 2215–2220 (2019).
18. B. Niu, H. Wang, J. Wang, and L. Tan: Multi-objective bacterial foraging optimization. Neurocomputing **116**, 336–345 (2013).
19. B. Niu, J. Wang, H. Wang: Bacterial-inspired algorithms for solving constrained optimization problems. Neurocomputing **148**, 54–62 (2015).
20. The homepage of the competition problem (in chinese), `https://tianchi.aliyun.com/competition/entrance/531831/information`, (2020).