SCIENCE CHINA
Information Sciences

• LETTER •

September 2019, Vol. 62 199102:1–199102:3
https://doi.org/10.1007/s11432-018-9539-6

Zoning search using a hyper-heuristic algorithm

Qinqin FAN[1,2], Ning LI[1*], Yilian ZHANG[3] & Xuefeng YAN[4]

[1]*Key Laboratory of System Control and Information Processing, Ministry of Education of China, Shanghai Jiao Tong University, Shanghai* 200240*, China;*
[2]*Logistics Research Center, Shanghai Maritime University, Shanghai* 201306*, China;*
[3]*Key Laboratory of Marine Technology and Control Engineering, Ministry of Communications, Shanghai Maritime University, Shanghai* 201306*, China;*
[4]*Key Laboratory of Advanced Control and Optimization for Chemical Processes of Ministry of Education, East China University of Science and Technology, Shanghai* 200237*, China*

Received 14 June 2018/Revised 27 June 2018/Accepted 31 July 2018/Published online 29 July 2019

Citation Fan Q Q, Li N, Zhang Y L, et al. Zoning search using a hyper-heuristic algorithm. Sci China Inf Sci, 2019, 62(9): 199102, https://doi.org/10.1007/s11432-018-9539-6

Dear editor,

With the development of high-performance computing techniques, run time of meta-heuristic algorithms can be reduced effectively. However, improved computation power has not been indirectly converted into search capability in most of previous studies [1]. Moreover, the solution precision of an optimization problem is directly determined by the algorithm performance and problem complexity. On one hand, giving enough computational resources does not mean that a single meta-heuristic algorithm can find a high-quality solution for complex problems due to performance limitation. Therefore, using an algorithm that has good robustness and performance is important [2]. On the other hand, the problem complexity is determined by both the search and the objective spaces. However, in reality, it is hard to reduce the problem complexity according to the objective space. Therefore, partitioning the search space is a promising approach for reducing the problem complexity, i.e., dividing the entire search space into some subspaces, especially when parallel computing, distributed computing, and cloud computing are used to assist the algorithms.

In the present study, a recently proposed hyper-heuristic algorithm [3], which uses differential evolution algorithm (DE) as a search engine, is applied to search a good-quality solution in each subspace. The hyper-heuristic algorithm can automatically select a suitable algorithm from an algorithm pool for solving a particular problem. Moreover, search space partition can reduce the problem complexity. Therefore, current studies not only use a well-performing algorithm, but also reduce the problem complexity by search space partition. To the best of our knowledge, a hyper-heuristic algorithm is first used in zoning search. Further, the effectiveness of the proposed algorithm is evaluated on a widely used test suite proposed in IEEE CEC2014. The results indicate that the proposed algorithm is effective and efficient.

Differential evolution. Without loss of generality, a single objective minimization problem can be defined as follows:

$$f(\boldsymbol{x}) = \min_{\boldsymbol{x}_i \in \Omega} f(\boldsymbol{x}_i), \quad \boldsymbol{x}_i \in \boldsymbol{S} \subseteq \prod_{j=i}^{D} [L_j, U_j], \quad (1)$$

where f denotes the objective function; $\boldsymbol{x}_i = (x_{i,1}, \ldots, x_{i,D})$ is a D-dimensional decision vector; $\Omega \subseteq \mathbb{R}^D$; L_j and U_j $(j = 1, 2, \ldots, D)$ are the minimum and maximum bounds of the j-th decision variable of $\boldsymbol{x}_i$, respectively; and $\boldsymbol{S}$ is the search space.

The procedure of DE can be formulated as:

(1) **Initialization.** Set mutation control parameter F, crossover control parameter CR, population size NP, and maximum number of generations $G_{\max}$. Initial individuals $\boldsymbol{x}_i^0$ $(i = 1, 2, \ldots,$ NP) are generated randomly from $\boldsymbol{S}$. The current generation G is set to be 0.

* Corresponding author (email: ning_li@sjtu.edu.cn)

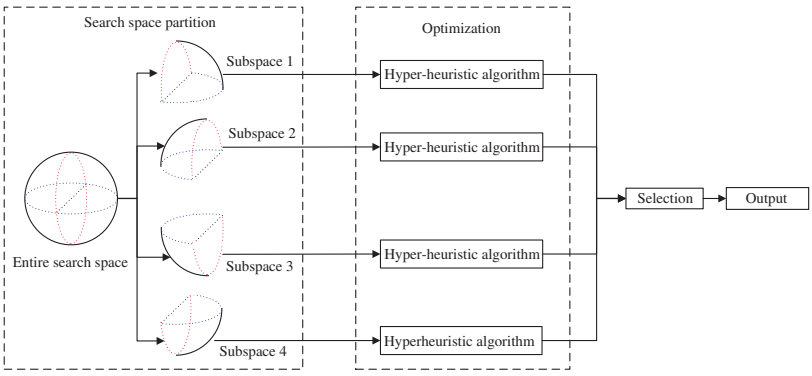

Figure 1 (Color online) An illustrative example of ZS.

(2) **Mutation.** After the initialization operation, for each target vector $\boldsymbol{x}_i^G$ in the current population, a mutation strategy is utilized to generate a mutant vector $\boldsymbol{v}_i^G$. One of the most frequently used mutation strategy is described as follows:

$$\boldsymbol{v}_i^G = \boldsymbol{x}_{r_1}^G + F\big(\boldsymbol{x}_{r_2}^G - \boldsymbol{x}_{r_3}^G\big), \tag{2}$$

where r_1, r_2, and r_3 are mutually exclusive integers randomly chosen within the range $[1, 2, \ldots, \mathrm{NP}]$, and are also different from the index i, i.e., $r_1 \neq r_2 \neq r_3 \neq i$.

(3) **Crossover.** For each target vector $\boldsymbol{x}_i^G$, a trial vector $\boldsymbol{u}_i^G$ is generated as follows:

$$u_{ij}^G = \begin{cases} v_{ij}^G, & \mathrm{rand}_j \leqslant \mathrm{CR} \text{ or } j = j_{\mathrm{rand}}; \\ x_{ij}^G, & \text{otherwise}, \end{cases} \tag{3}$$

where rand_j and j_{rand} are random numbers uniformly generated within the range $[0, 1]$ and $[1, D]$, respectively.

(4) **Selection.** The trial vector $\boldsymbol{u}_i^G$ competes with its target vector $\boldsymbol{x}_i^G$ and then the better individual will survive for the next generation

$$\boldsymbol{x}_i^{G+1} = \begin{cases} \boldsymbol{u}_i^G, & f(\boldsymbol{u}_i^G) \leqslant f(\boldsymbol{x}_i^G); \\ \boldsymbol{x}_i^G, & \text{otherwise}. \end{cases} \tag{4}$$

(5) Implement steps (2) to (4) repeatedly until the number of generations is equal to $G_{\max}$.

Zoning search (ZS). To improve the solution precision of an optimization problem, two approaches are usually adopted in previous studies. One way is to develop powerful algorithms, while the other is to simplify the problem. Overall, most previous studies have focused on improving the algorithm performance. In the present work, a ZS strategy, which can obtain some reduced search spaces, is proposed. Two main operations are implemented in the proposed algorithm. (1) Search space partition. The entire search space is divided into some subspaces. (2) Optimization. A hyper-heuristic algorithm is employed to locate a promising solution in each subspace. Based on the above two steps, it can be observed that, search space partition can effectively reduce the problem complexity and using a hyper-heuristic algorithm can improve the probability of finding a good solution. Additionally, high-performance computing techniques can be implemented in the proposed algorithm easily. The detailed steps are described as follows.

Step 1. Search space partition. For a D-dimensional optimization problem, the first step is to choose h $(1 \leqslant h \leqslant D)$ variables randomly. Subsequently, each selected variable is divided into m $(m > 1)$ segment vectors. Hence, the entire search space is divided into $n = m^h$ subspaces.

Step 2. Using a hyper-heuristic algorithm within each subspace. A hyper-heuristic algorithm is used to find a high-quality solution within each subspace. Clearly, because information in each subspace does not need to exchange with each other, high-performance computing techniques can be used in ZS, such as parallel computing, distributed computing, and graphics process unit (GPU).

To further introduce the proposed algorithm, an illustrative example of ZS is presented in Figure 1. The entire search space is divided into four subspaces in this example.

Experimental study. A frequently used test suite proposed in IEEE CEC2014 is employed to test the effectiveness of ZS. A hyper-heuristic algorithm (named as ASM-JADE-SSCPDE) proposed in [3] is used. The performance of ZS-based ASM-JADE-SSCPDE is compared with that of JADE [4], SSCPDE [5], ZS-JADE, and ZS-SSCPDE. Additionally, four nonparametric statistical analysis approaches using a significance level of 5%, which contain the Wilcoxon's rank sum test, the Bonferroni-Dunn's test, the Holm's procedure, and the Hochberg's procedure, are employed to analyze the obtained experimental results. The more detailed parameter settings of the selected algo-

rithms are presented in Table A1.

In the present work, the maximum number of function evaluations (MFEs) is set to be $10000 \times D$. The problem dimensionality is set to be 30. It should be noted that, we assume that computational resource is enough, thus MFEs is set to be $10000 \times D$ in each subspace. All compared algorithms are run for 30 independent times on each test function. Additionally, eight variables are randomly selected in the proposed algorithm, i.e., $h = 8$. Moreover, each variable is uniformly divided into two segment vectors. Namely, the number of subspaces is $2^8 = 256$.

The values of the mean and standard deviation for all compared algorithms are summarized in Table A2. Moreover, the statistical analysis results achieved by the Wilcoxon's rank sum test are also listed in Table A2. It can be observed that, the results of the proposed algorithm are significantly better than that of JADE and SSCPDE without using ZS on 23 and 25 test functions, respectively. Therefore, ZS and ASM-JADE-SSCPDE can significantly improve the solution precision. Additionally, Table A2 indicates that JADE outperforms the proposed algorithm on four functions. The reason is twofold: (1) JADE performs well on some simple unimodal and multimodal problems, whereas $F1_{\text{CEC2014}} - F3_{\text{CEC2014}}$ are unimodal functions, and $F4_{\text{CEC2014}} - F16_{\text{CEC2014}}$ are simple multimodal functions; (2) SSCPDE has good performance on complex problems, but it may perform poorly on some unimodal and multimodal test functions. Therefore, SSCPDE in ASM-JADE-SSCPDE may waste some computational budgets. It can be observed from Table A2 that the overall performance of the proposed algorithm is similar to that of ZS-JADE. The main reason may be that, although ASM-JADE-SSCPDE can automatically select a suitable algorithm to solve a particular optimization problem, a poorly performing individual algorithm would wastes some computational budgets during the evolutionary process. Therefore, the results achieved by ASM-JADE-SSCPDE are between JADE and SSCPDE on most functions. However, the performance of ASM-JADE-SSCPDE is more robust when compared with JADE. Table A2 also shows that the performance of the proposed algorithm is slightly better than that of ZS-SSCPDE. This is because ASM-JADE-SSCPDE is capable of selecting JADE to solve some unimodal and multimodal functions. Moreover, ASM-JADE-SSCPDE can inherit the capability of SSCPDE on some functions, thus the proposed algorithm is competitive when compared with SSCPDE by using ZS. Table A3 indicates that the average performance of the proposed algorithm is significantly better than that of JADE and SSCPDE, and is similar to that of ZS-JADE and ZS-SSCPDE.

Based on the above comparisons, we can see that ZS can reduce the problem complexity and ASM-JADE-SSCPDE improves the algorithm robustness. Therefore, the proposed algorithm is effective and efficient.

Conclusion and future work. This study proposed a zoning search strategy and uses a hyper-heuristic algorithm for finding a promising solution in each subspace. A famous test suite introduced in IEEE CEC2014 is employed to verify the performance of the proposed algorithm. The experimental results indicate that the proposed algorithm is a competitive optimization method for solving complex optimization problems. In the future, we will try to extend the work for solving other types of optimization problems, such as multi-modal optimization problems, multi-/many-objective optimization problems, and large-scale optimization problems. Moreover, how to use advanced machine learning methods to select variables of an optimization problem is vital.

Acknowledgements This work was supported in part by National Natural Science Foundation of China (Grant Nos. 61590925, 61603244, 61773260), and in part by China Postdoctoral Science Foundation (Grant No. 2018M642017).

Supporting information Tables A1–A3. The supporting information is available online at info.scichina.com and link.springer.com. The supporting materials are published as submitted, without typesetting or editing. The responsibility for scientific accuracy and content remains entirely with the authors.

References

1 Lin F, Phoa F. A performance study of parallel programming via CPU and GPU on swarm intelligence based evolutionary algorithm. In: Proceedings of International Conference on Intelligent Systems, Metaheuristics and Swarm Intelligence, 2017
2 Laili Y J, Zhang L, Tao F, et al. Rotated neighbor learning-based auto-configured evolutionary algorithm. Sci China Inf Sci, 2016, 59: 052101
3 Fan Q Q, Yan X F, Zhang Y L. Auto-selection mechanism of differential evolution algorithm variants and its application. Eur J Oper Res, 2018, 270: 636–653
4 Zhang J Q, Sanderson A C. JADE: adaptive differential evolution with optional external archive. IEEE Trans Evol Comput, 2009, 13: 945–958
5 Fan Q Q, Yan X F. Differential evolution algorithm with self-adaptive strategy and control parameters for P-xylene oxidation process optimization. Soft Comput, 2015, 19: 1363–1391