Li *et al. EURASIP Journal on Wireless Communications and Networking* (2018) 2018:256
https://doi.org/10.1186/s13638-018-1259-2

EURASIP Journal on Wireless Communications and Networking

RESEARCH Open Access

Task scheduling algorithm based on fireworks algorithm

Jingmei Li[1], Qiao Tian[1*], Guoyin Zhang[1], Weifei Wu[1], Di Xue[1], Lanting Li[1], Jiaxiang Wang[1] and Lei Chen[2]

Abstract

To give full play to the high efficiency and parallelism of multi-processor systems, the fireworks algorithm (FWA) is improved, and a multi-processor task scheduling algorithm based on improved FWA, named IMFWA, is proposed. IMFWA maps continuous space to discrete space by designing the fireworks location coding method, improves the Gaussian mutation process, and sets adaptive dimensions to accelerate the convergence speed of the algorithm. At the same time, in order to reduce the time complexity of the algorithm and shorten the time finding the optimal task scheduling sequence, the fitness-based tournament selection strategy is used instead of the rule based on Euclidean distance. Finally, IMFWA is compared with the basic fireworks algorithm and the genetic algorithms on the Matlab platform for performance analysis. The results show that the IMFWA has advantages in the convergence speed, and the negative impact of the number of tasks is also lower than the fireworks algorithm and genetic algorithm.

Keywords: Task scheduling, Fireworks algorithm, Gaussian mutation, Adaptive

1 Introduction

The task scheduling problem belongs to the combinatorial optimization problem and cannot be solved within the polynomial time complexity [1]. It has been proved to be an NP-hard problem [2, 3]. It is easy for multiple independent tasks to be scheduled on homogeneous multiprocessors, by only scheduling the task with the shortest completion time to the processor. The representative task scheduling scheme on heterogeneous multi-processors is more complex. The performance parameters, such as execution efficiency and total time of completion, need to be considered [4–7]. Research on task scheduling strategies for heterogeneous multi-processors has drawn much attention in recent years [8–10]. At present, the combination optimization algorithm, which is widely used in task scheduling problem, is genetic algorithm (GA) [11–13]. Nevertheless, the parameters of genetic algorithm are complicated to configure, and the effectiveness of the crossover and mutation operations decreases when the number of tasks increases, and the "premature" phenomenon can be easily triggered by the population initialization of the individuals [12, 14–18].

The fireworks algorithm (FWA) is a swarm intelligence optimization algorithm based on non-living population proposed by Ying Tan in 2010 [1, 19, 20]. The fireworks algorithm can provide the global optimal solution for complex problems [1, 20, 21] and has low definition for solving the problem. Since proposed, it has been widely applied to non-negative matrix factorization calculation, directional characteristic matrix measure, group robot multi-target search, fertilization problem, traveling salesman, among others [20].

Based on the above literature and background, a heterogeneous multi-processor task scheduling algorithm, IMFWA, is proposed, for making the fireworks algorithm suitable for the discrete multi-processor task scheduling problem and for efficiently and quickly obtaining the optimal task scheduling sequence. Compared with the basic fireworks and genetic algorithm on the Matlab simulation platform, it is proved that IMFWA has noticeable advantages in convergence speed, and the impact on the number of tasks is also significantly lower than genetic algorithm.

The paper is organized as follows: Section 1 introduces the research background for the problem. Section 2 describes the basic task scheduling problem. Section 3 presents the fireworks algorithm and its improvement strategy. The proposed algorithm is presented in

*Correspondence: tianqiao@hrbeu.edu.cn
[1]College of Computer Science and Technology, Harbin Engineering University, Harbin, China
Full list of author information is available at the end of the article

Section 4. Section 5 provides the experiment results and analysis, and Section 6 concludes this paper.

2 Description of task scheduling problem

Task scheduling is to obtain the optimal solution of the task sequence according to an algorithm, and tasks are assigned to the corresponding multi-processor according to the optimal solution order, so that the task completion time is the shortest [22–24]. For convenience, the system contains n heterogeneous multi-processors, denoted by $P = \{P_1, P_2, \ldots, P_n\}$. Divide any application into m independent tasks, represented by $T = \{T_1, T_2, \ldots, T_m\}$. $n \geq m$ means there are more processors than tasks to be scheduled. The next task is scheduled to an idle processor which has the shortest execution time according to the first-come-first-service mechanism. When $n \prec m$, the number of processors is smaller than the number of tasks, and the tasks need to be assigned following the scheduling scheme. This paper only considers the case of $n \prec m$. $exet_{ij}$ is the execution time of task T_i on processor P_j. The execution time of all tasks under the scheduling scheme is $CP(s)$. $\min(CP(s))$ indicates the time used to execute all tasks under the optimal scheduling scheme f. The task scheduling problem can be expressed as Eq. (1).

$$CP(f) = \min(CP(s)) \tag{1}$$

3 Fireworks algorithm and its improvement

3.1 The fireworks algorithm

The fireworks algorithm is a parallel explosive optimization algorithm proposed by Tan Ying, inspired by the natural fireworks exploding behavior [20]. In the fireworks algorithm, the sparks generated by fireworks and their explosions and mutations resemble feasible solutions, and the explosion process simulates the process of optimizing in the current feasible solution domain [25, 26]. In nature, high-quality fireworks explosions produce more and more sparks, while the inferior fireworks explosions produce much fewer sparks. The fireworks algorithm follows the natural rules: the fitness value is good, the fireworks quality is excellent, the explosion range is small, and the sparks generated are in large quantity. On the other hand, when the fitness value is not good, the fireworks quality is inferior, the explosion range is large, and the sparks generated are limited [1, 20, 21, 27]. The fireworks algorithm includes explosive sparks, mutation sparks, mapping rules, selection strategies, and other elements [20].

The explosion operator produces explosion sparks, which play a key role in the fireworks algorithm, whose metrics include explosion intensity, explosion amplitude, and displacement operation [28, 29]. The explosion intensity is the core of the explosion operator, indicating the quantity of sparks generated during the explosion [30, 31]. The fireworks based on better fitness value can produce more sparks and avoid them swinging around the optimal value during optimization. The calculation method for the number of sparks is shown in Eq. (2).

$$S_i = S \cdot \frac{F_{\max} - f(X_i) + \epsilon}{\sum_{i=1}^{N} \left(F_{\max} - f\left(X_i\right)\right) + \epsilon} \tag{2}$$

S_i indicates the number of subgeneration sparks in the ith fireworks. S represents the maximum number of sparks in the fireworks subgeneration. $F_{\max}$ indicates the worst fitness value of this generation. $f(X_i)$ represents the fitness value of the ith fireworks. ϵ is a minimal constant, avoiding the divide-by-zero error.

In order to control the number of subgeneration sparks of high quality fireworks and inferior fireworks, fireworks i is limited, as shown in Eq. (3).

$$S_i = \begin{cases} \text{round}(a \cdot S) & S_i \prec a \cdot S \\ \text{round}(b \cdot S) & S_i \succ b \cdot S \\ \text{round}(S) & \text{else} \end{cases} \tag{3}$$

a and b are given constants. round is the integral function, following the principle of rounding.

The explosion amplitude is set to explore the optimum value within a certain range around this generation of fireworks [32, 33]. The fireworks with poor fitness value can produce sparks in a wider range and avoid "premature" phenomenon [34, 35]. The calculation method of explosion amplitude is shown in Eq. (4).

$$R_i = R \cdot \frac{f(X_i) - F_{\min} + \epsilon}{\sum_{i=1}^{N} \left(f\left(X_i\right) - F_{\min}\right) + \epsilon} \tag{4}$$

R_i indicates the range of the explosion amplitude of the ith fireworks. R shows the maximum range of the fireworks explosion. $F_{\min}$ represents the best fitness value of the current generation of fireworks.

The displacement operation is performed on each dimension of the fireworks i according to the explosion intensity and the explosion amplitude [36, 37], as shown in Eq. (5).

$$\Delta X_i^k = X_i^k + \text{rand}(0, R_i) \tag{5}$$

X_i^k indicates the value of the location vector of fireworks i in the kth dimension. ΔX_i^k represents the value of the location vector of the sparks generated by fireworks i explosion in the kth dimension. $\text{rand}(0, R_i)$ indicates the random number between 0 and R_i.

In the fireworks algorithm, the diversity of the population is further improved by the mutation spark. The Gauss distribution is used to preform Gauss mutation on any dimensions of fireworks in the population [38–40]. The calculation method is shown in Eq. (6).

$$\nabla X_i^k = X_i^k \cdot n \tag{6}$$

∇X_i^k indicates the value of the location vector of the sparks produced by fireworks i Gauss mutation in the kth dimension. n obeys the Gauss distribution of the mean value of 1 and the variance of 1, as shown in Eq. (7).

$$n \sim N(1,1) \tag{7}$$

If fireworks i is near the boundary of the feasible domain, it may produce sparks across the boundary. Therefore, we use the rule of modular operation to map it back to the feasible domain, as shown in Eq. (8).

$$X_i^k = X_{\min}^k + \left|X_i^k\right| \bmod \left(X_{\max}^k - X_{\min}^k\right) \tag{8}$$

$X_{\max}^k$ and $X_{\min}^k$ indicate the upper and lower bounds of the location vector of fireworks i in the kth dimension, respectively.

According to the explosion operator and Gauss mutation operator, the current population contains this generation of fireworks, the explosion sparks, and the Gauss mutation sparks. The individuals with the best fitness values are retained to the next generation with probability 1. Then, the rest $N-1$ individuals are selected according to the Roulette rule with the probability in Eq. (9).

$$P_i = \frac{\sum_{j\in K} D_{ij}}{\sum_{i\in K}\sum_{j\in K} D_{ij}} \tag{9}$$

P_i indicates the probability that fireworks i is selected. K is a set of fireworks, explosion sparks, and Gauss mutation sparks. D_{ij} represents the Euclidean distance between fireworks i and fireworks j, as shown in Eq. (10):

$$D_{ij} = \sum_{j=1}^{K} ||X_i - X_j|| \tag{10}$$

3.2 The process of adaptive Gaussian mutation

The basic fireworks algorithm increases the diversity of the population through Gauss mutation and uses Gauss distribution to mutate any of the multiple dimensions of the fireworks in the population [20]. The actual effect of Gauss mutation can be easily influenced by the selected Gauss mutation fireworks and mutation dimensions. Therefore, for having a good fireworks population diversity and short convergence time, the process of Gauss mutation is redesigned.

With a poor fitness value of Gauss mutation, the contribution of the poor fireworks to the mutant becomes too large, thereby reducing the convergence speed of the algorithm. According to Pareto's rule, the most important part of anything is only 20%, and the remaining 80% are secondary, although they are the majority [41]. Therefore, in order to ensure that the algorithm has a fast convergence speed, the improved fireworks algorithm randomly selects one of the fireworks for Gauss mutation among the top 20% of the fitness value.

When the fireworks with better fitness value are selected for Gauss mutation, if the mutation dimension constantly remains large, despite the improved diversity of the fireworks population, the contribution of fireworks with better fitness value to the population is reduced, thereby slowing down the convergence speed of the algorithm. In contrast, if the mutation dimension constantly remains small, although the information of the fireworks with better fitness value is retained, the diversity of the fireworks population is also reduced and consequently makes the algorithm fall into the local optimum. Therefore, to guarantee a fast convergence speed while not falling into the local optimal, the adaptive Gauss mutation dimension is presented. The value of the dimension is shown in the Eq. (11).

$$z(t+1) = \lceil w(t) \times z(t) \rceil \tag{11}$$

w is a nonlinear function that decreases by the iterations t, as shown in Eq. (12). Considering that the value of Gauss mutation dimension should be a positive integer, the value of $z(t+1)$ is rounded up to $w(t) \times z(t)$.

$$w(t) = e^{-t} \tag{12}$$

At the beginning of iteration, w is relatively large, and the corresponding mutation dimension z is also large, which helps improve the diversity of fireworks population and enhances the algorithm with global search. In the later stage of the iteration, the value of w decreases gradually along with the iterations. The corresponding mutation dimension z decreases, the Gauss mutation dimension decreases gradually, and the information of the fireworks with better fitness value is preserved, so that the algorithm achieves optimization in the later stage of the iterations. The improved fireworks algorithm fully plays the role of Gauss mutation, avoiding the waste of resources caused by improper selection of the mutation fireworks and the improper determination of the mutation dimension. Meanwhile, the algorithm improves the ability of global search and convergence speed.

4 IMFWA task scheduling algorithm

4.1 Coding strategy

Fireworks algorithm is an explosive optimization algorithm for continuous space, and task scheduling is a discrete problem. Therefore, according to the characteristics of heterogeneous multi-processor task scheduling, IMFWA encodes every generation of fireworks and maps the continuous search space to the discrete search space, so that fireworks algorithm can be applied to the task scheduling problem.

The fireworks or spark in the population represents a possible way of task scheduling. The processors and

tasks are numbered respectively: the processor numbers are $1, 2, \ldots, n$, and the task numbers are $1, 2, \ldots, m$. Let the location vector of fireworks or sparks be an m-dimensional vector X_i, as shown in Eq. (13).

$$X_i = \left[x_1, x_2, \ldots, x_j, \ldots, x_m\right], x_j = \text{rand}(1, n) \tag{13}$$

$j = \{1, 2, 3, \ldots, m\}$, and $\text{rand}(1, n)$ indicates a random value between 1 and n, then:

(1) The jth task is distributed to the x_j processor.
(2) The requirements of task scheduling for heterogeneous multi-processors are met.
(3) Each task can and can only be executed by one processor.
(4) Each processor can execute multiple tasks.

In this coding scheme, the fireworks or spark location vector is retained as a multi-dimensional vector, and only the meaning and value represented by each dimension are limited. In order to apply the coding scheme to the heterogeneous multi-processor task scheduling problem, the displacement operation formula Eq. (5) is updated to Eq. (14).

$$X_j = X_j + \text{rand}(0, R_i) \tag{14}$$

When applying Eq. (14) to the execution of displacement operation, the value of x_j may be beyond the range, and the mapping rule Eq. (8) needs to be updated to Eq. (15).

$$X_j = 1 + |X_j| \text{mod}(n - 1) \tag{15}$$

The coding scheme has four features and, therefore, is suitable for heterogeneous multi-processor task scheduling:

(1) The encoded mode is simple and clear and easy to understand and implement.
(2) The requirements of task scheduling for heterogeneous multi-processors are met.
(3) Contains all possible task scheduling schemes.
(4) The unique mapping of location vector and task scheduling sequence for fireworks or sparks.

4.2 Fitness value

In the fireworks algorithm, the quality of each generation of fireworks and their offspring is evaluated by the fitness value. IMFWA uses $CP(s)$, the time used to execute all tasks under the task scheduling sequence s, as the fitness evaluation standard. A smaller value of $CP(s)$ indicates faster task execution, and a smaller fitness value presents better quality of the fireworks or sparks. On the contrary, a larger value of $CP(s)$ indicates slower task execution, and a higher fitness value presents worse quality of the fireworks or sparks. According to the coding scheme, $CP(X_i)$ is the completion time of the task scheduling sequence corresponding to fireworks i or spark, and the fitness value calculation method is as shown in Eq. (16).

$$f(X_i) = CP(X_i) \tag{16}$$

$CP(X_i)$ is calculated via Eq. (17).

$$CP(X_i) = \sum_{j=1}^{m} \text{exet}_{x_j} \tag{17}$$

The fitness value calculation method is shown in Eq. (18).

$$f(X_i) = \sum_{j=1}^{m} \text{exet}_{x_j} \tag{18}$$

4.3 Tournament selection strategy

The basic fireworks algorithm uses the Roulette rules based on Euclidean distance to select the next generation of fireworks; the higher the distance from the other fireworks or sparks, the higher the probability of being selected. Although this selection strategy allows the algorithm to avoid the local optimal solution, the time overhead increases greatly when the dimension of the location vector X_i of the fireworks increases. This leads to long execution time of the algorithm in actual applications [1]. Therefore, IMFWA improves the basic fireworks algorithm and adopts the fitness-based tournament selection strategy instead of the Roulette rule based on Euclidean distance [1, 20, 21]. This reduces the time cost of the algorithm on the basis of ensuring the population diversity.

Fitness-based tournament selection strategy first selects a certain number of fireworks or sparks from the population to form the next generation fireworks candidate set in each iteration. Then, it chooses the best fireworks or sparks in the candidate set to enter the next generation according to the fitness value. The specific process is as follows:

(1) To determine the percentage of fireworks or sparks in the next generation of fireworks candidates out of the total number of fireworks or sparks in the contemporary population.
(2) In accordance with the percentage determined in (1), the contemporary fireworks population is randomly selected to form the next generation of fireworks candidates set.
(3) According to fitness values, the best fireworks or sparks are retained in the next generation within the candidate set determined in (2).

Based on the above, the selection strategy of IMFWA ensures that the individuals with the optimal fitness value still remain in the next generation with the probability of 1. Then the remaining $N - 1$ individuals are selected

according to the fitness-based tournament strategy. The probability of each fireworks or spark being selected is shown in Eq. (19).

$$P_i = \frac{r\left(F'_{\max} - f\left(X_i\right)\right)}{\sum_{i\in Q}\left(F'_{\max} - f\left(X_i\right)\right)} \tag{19}$$

Q represents the next generation of fireworks candidates selected according to a certain percentage of r. $F_{\max}$ indicates the maximum fitness value of an individual in set Q. IMFWA adopts this fitness-based tournament selection strategy to reduce the computational cost of selecting each generation of fireworks and shorten the actual execution time of the algorithm.

4.4 The process of IMFWA

The process of the IMFWA scheduling algorithm is as follows:

(1) Initialize the number of fireworks N, the basic explosion spark number S, the basic explosion amplitude R, the number of tasks m, the multi-processors number n, and the maximum iteration number I.
(2) Initialize the fireworks location and convert the fireworks location into a task scheduling sequence according to the encoding scheme in Section 4.1.
(3) Calculate the fitness value, explosion intensity, and explosion range of the fireworks.
(4) Start the explosion, generate ordinary sparks, and calculate the fitness value.
(5) Calculate the Gaussian mutation dimension according to Section 3.2, generate the Gaussian mutation spark, and calculate its fitness value.
(6) Select the next generation of fireworks according to the tournament selection strategy in Section 4.3.
(7) Determine if the maximum number of iterations has been reached. If yes, the algorithm stops and outputs the task scheduling sequence corresponding to the fireworks or sparks with the smallest the fitness value. Otherwise, the algorithm returns to step 3, and the algorithm continues to execute.

The flow of algorithm is shown in Fig. 1.

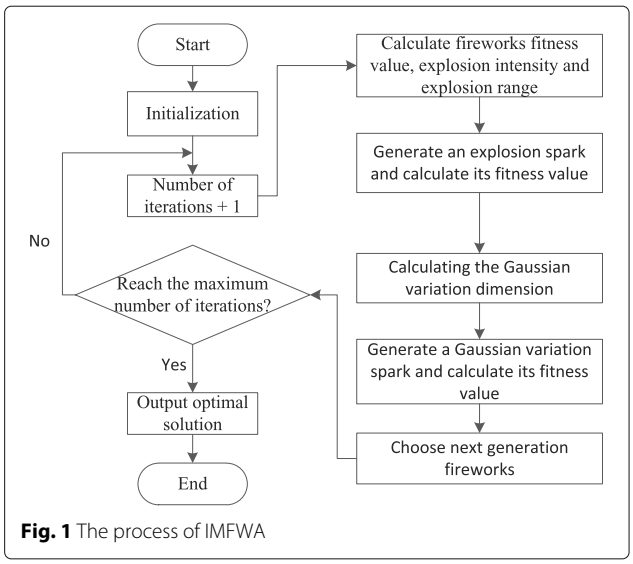

Fig. 1 The process of IMFWA

5 Experiment results and analysis

5.1 Experimental methods

For evaluating the performance of IMFWA algorithm, this research designed the comparative experiments with the basic fireworks algorithm and genetic algorithm which is widely used in task scheduling. The fireworks algorithm is distributed and parallel. Compared with the chromosome information sharing mechanism of genetic algorithm, the fireworks algorithm adopts a distributed information sharing mechanism. The population is not uniformly to the optimal region, but the explosion and sparks are determined according to the fitness value of the fireworks distributed in different regions. Although the selection strategy of fireworks algorithm and genetic algorithm both introduce the idea of immune concentration, the fireworks algorithm has more mechanisms to avoid falling into the local optimal solution.

The experimental platform is Matlab R2014b and the operating system is Windows 10. In order to objectively analyze the performance of the algorithms, several comparative experiments are set up to test the performance of the three algorithms under the conditions of eight processors and 50, 100, 150, 200, and 250 tasks.

To eliminate the contingency of random data, the total task completion time obtained is the average of 20 experiments. The specific parameter settings of IMFWA, GA, and FWA are shown in Table 1. Parameter N indicates the initial quantity of fireworks per generation of the population in IMFWA and FWA. In GA, it indicates the quantity

Table 1 Parameter settings of three algorithms

Algorithm	Parameter	Value
IMFWA, FWA	Number of fireworks N	100
	Basic explosion spark number S	80
	Basic explosive amplitude R	300
	Maximum number of iterations I	500
	Constant a	0.80
	Constant b	0.04
GA	Individual number N	100
	Cross probability	0.80
	Mutation probability	0.04
	Maximum number of iterations I	500

of initial individuals per generation with the value of 100. The maximum number of iterations I is 500.

5.2 Results and discussion

Figure 2 shows the scenario when $m = 200$ and $n = 8$, where the average performance curve of all task completion time-iteration times after 200 tasks are executed on eight heterogeneous processors. In Fig. 2, among the three algorithms, IMFWA has the least number of iterations and the fastest convergence speed. As the number of iterations increases, the task completion time of IMFWA is consistently shorter than that of FWA and GA. It can be seen that the accuracy of IMFWA and FWA is superior to that of the genetic algorithm. This is because the crossover and mutation operations of genetic algorithm increase the time complexity of the algorithm, and it tends to miss the optimal solution. While the iterative process of IMFWA is relatively simple, the search speed is faster and the precision is higher.

Figure 3 shows the algorithm execution time-task number curve for different task numbers when there are eight processors. In Fig. 3, with the increased number of tasks, IMFWA still maintains lower task execution time when compared with FWA and genetic algorithm. It can be seen that IMFWA is less affected by the number of tasks than FWA, and both algorithms are less affected by the number of tasks than genetic algorithm. This is because the cross-mutation operation of genetic algorithm reduces its effectiveness in large dimensions and is most affected by the dimension. IMFWA reduces the time complexity of the algorithm by improving the Gauss mutation and the selection strategy.

The experimental results and analysis show that IMFWA has the feasibility and efficiency in solving the task scheduling problem. It retains the characteristics of the basic fireworks algorithm's accuracy in finding the optimal or suboptimal task scheduling sequence. Meanwhile, it shortens the overall completion time of tasks on multi-processor, where its parallelism and heterogeneity can be fully utilized.

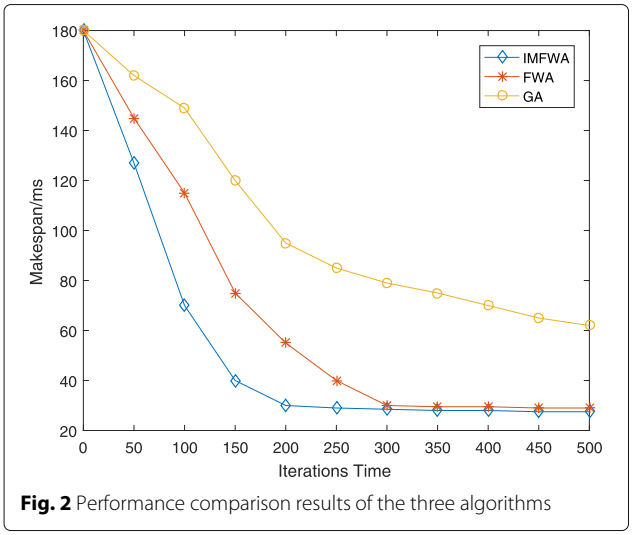

Fig. 2 Performance comparison results of the three algorithms

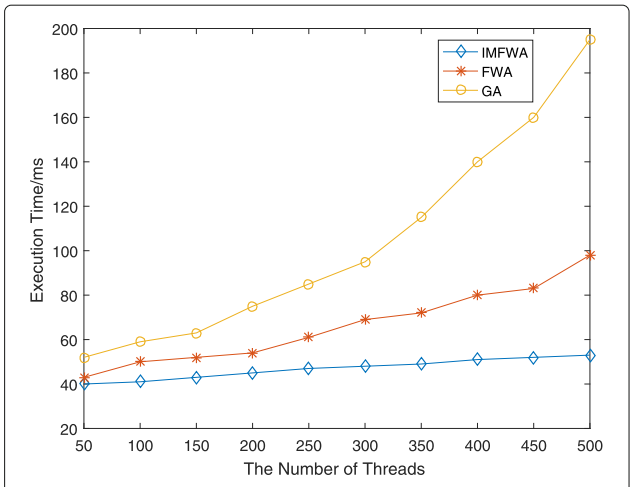

Fig. 3 Execution results of the three algorithms

6 Conclusions

To exploit the parallelism and efficiency of heterogeneous multi-processor systems, a task scheduling algorithm based on improved fireworks algorithm is proposed. By setting the coding method of the fireworks location, the solution space of the fireworks algorithm is converted from continuous to discrete, so that it is suitable for task scheduling problem. At the same time, IMFWA improves the Gauss mutation process of the basic fireworks algorithm, effectively accelerating the convergence speed of the algorithm. The algorithm introduces a fitness-based tournament selection strategy, which reduces the time complexity of the algorithm. On the Matlab simulation platform, compared with the basic fireworks algorithm and genetic algorithm applied to the task scheduling problem, it is proved that IMFWA can find the optimal solution within fewer iteration times. The convergence speed is faster, and the execution time of the algorithm is also shorter. When the number of tasks increases, IMFWA can still maintain a good optimization speed and solution accuracy.

Abbreviations

FWA: Fireworks algorithm; GA: Genetic algorithm; IMFWA: Improved fireworks algorithm; NP: Non-deterministic polynomial

Funding

This work was supported by the National Key Research and Development Plan of China (No.2016YFB0801004).

Availability of data and materials

The authors declare that all the data and materials in this manuscript are available.

Authors' contributions
The contributions of all authors are equal in this manuscript, and all authors read and approved the final manuscript.

Authors' information
Jingmei Li received her M.S. degree and Ph.D. degree from Harbin Engineering University, Harbin, China. She is currently a professor working in College of Computer Science and Technology, Harbin Engineering University. Her research focuses on Computer Architecture Performance Optimization, Big Data and Cloud Computing, Network and Information Security and Embedded Technology (E-mail:lijingmei@hrbeu.edu.cn).
Qiao Tian is currently a Ph.D. student at College of Computer Science and Technology, Harbin Engineering University. Her main research includes System Parallel Optimization and Computer Architecture (E-mail:tianqiao@hrbeu.edu.cn).
Guoyin Zhang received his Ph.D. degree from Harbin Engineering University. He is currently a professor and Ph.D. supervisor at College of Computer Science and Technology, Harbin Engineering University. His current interests include Embedded System, Network Technology and Information Security (E-mail:zhangguoyin@hrbeu.edu.cn).
Weifei Wu received his M.S. degree at College of Computer Science and Technology, Harbin Engineering University. He is currently studying for his Ph.D. degree at the same institution (E-mail:wuweifei@hrbeu.edu.cn).
Di Xue is currently a Ph.D. student at College of Computer Science and Technology, Harbin Engineering University (E-mail:dixue@hrbeu.edu.cn).
Lanting Li is currently a Master studying at College of Computer Science and Technology, Harbin Engineering University.
Jiaxiang Wang is currently a professor at College of Computer Science and Technology, Harbin Engineering University.
Lei Chen is currently a professor of Georgia Southern University, USA.

Competing interests
The authors declare that they have no competing interests.

Publisher's Note
Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Author details
[1]College of Computer Science and Technology, Harbin Engineering University, Harbin, China. [2]Georgia Southern University, Georgia, USA.

Received: 25 July 2018 Accepted: 28 September 2018
Published online: 30 October 2018

References

1. N. Bacanin, M. Tuba, in *IEEE Congress on Evolutionary Computation, CEC 2015*. Fireworks algorithm applied to constrained portfolio optimization problem (IEEE, New York, 2015), pp. 1242–1249. https://doi.org/10.1109/CEC.2015.7257031
2. J.D. Ullman, Np-complete scheduling problems. J. Comput. Syst. Sci. **10**(3), 384–393 (1975). https://doi.org/10.1016/S0022-0000(75)80008-0
3. C.H. Papadimitriou, M. Yannakakis, Towards an architecture-independent analysis of parallel algorithms. SIAM J. Comput. **19**(2), 322–328 (1990). https://doi.org/10.1137/0219021
4. d.e.O.liveira. L.L., F.reitas. A.A., T.inós. R., Multi-objective genetic algorithms in the study of the genetic code's adaptability. Inf. Sci. **425**, 48–61 (2018). https://doi.org/10.1016/j.ins.2017.10.022
5. A.S. Pillai, K. Singh, V. Saravanan, A. Anpalagan, I. Woungang, L. Barolli, A genetic algorithm-based method for optimizing the energy consumption and performance of multiprocessor systems. Soft Comput. **22**(10), 3271–3285 (2018). https://doi.org/10.1007/s00500-017-2789-y
6. H. Topcuoglu, S. Hariri, M. Wu, Performance-effective and low-complexity task scheduling for heterogeneous computing. IEEE Trans. Parallel Distrib. Syst. **13**(3), 260–274 (2002). https://doi.org/10.1109/71.993206
7. G. Park, B.A. Shirazi, J. Marquis, in *Solving Irregularly Structured Problems in Parallel, 4th International Symposium, IRREGULAR '97*. Comparative study of static scheduling with task duplication for distributed systems (Springer, Berlin, 1997), pp. 123–133. https://doi.org/10.1007/3-540-63138-0_12
8. Z. Zheng, N. Saxena, K.K. Mishra, A.K. Sangaiah, Guided dynamic particle swarm optimization for optimizing digital image watermarking in industry applications. Futur. Gener. Comput. Syst. (2018). https://doi.org/10.1016/j.future.2018.05.027
9. X. Shi, Z. Zheng, Y. Zhou, H. Jin, L. He, B. Liu, Q.S. Hua, Graph processing on GPUs: a survey. ACM Comput. Surv. **50**(6), 1–35 (2018)
10. M.A. Al-Mouhamed, Lower bound on the number of processors and time for scheduling precedence graphs with communication costs. IEEE Trans. Softw. Eng. **16**(12), 1390–1401 (1990). https://doi.org/10.1109/32.62447
11. N. Zhang, X. Yang, M. Zhang, Y. Sun, K. Long, A genetic algorithm-based task scheduling for cloud resource crowd-funding model. Int. J. Commun. Syst. **31**(1) (2018). https://doi.org/10.1002/dac.3394
12. F.A. Omara, M.M. Arafa, Genetic algorithms for task scheduling problem. J. Parallel Distrib. Comput. **70**(1), 13–22 (2010). https://doi.org/10.1016/j.jpdc.2009.09.009
13. Z. Dou, C. Shi, Y. Lin, W. Li, Modeling of non-Gaussian colored noise and application in CR multi-sensor networks. EURASIP J. Wirel. Comm. Netw. **2017**, 192 (2017). https://doi.org/10.1186/s13638-017-0983-3
14. A.S.A. Beegom, M.S. Rajasree, in *Distributed Computing and Internet Technology - 11th International Conference, ICDCIT 2015*. Genetic algorithm framework for bi-objective task scheduling in cloud computing systems (Springer-Verlag, Berlin, 2015), pp. 356–359. https://doi.org/10.1007/978-3-319-14977-6_38
15. S.G. Ahmad, C.S. Liew, E.U. Munir, T.F. Ang, S.U. Khan, A hybrid genetic algorithm for optimization of scheduling workflow applications in heterogeneous computing systems. J. Parallel Distrib. Comput. **87**, 80–90 (2016). https://doi.org/10.1016/j.jpdc.2015.10.001
16. Z. Li, in *International Conference on Information Science and Engineering*. Optimization for the parallel test task scheduling based on GA (IEEE, 2010), pp. 5223–5226. https://doi.org/10.1109/ICISE.2010.5689193
17. B. Kruatrachue, T. Lewis, Grain size determination for parallel processing. IEEE Softw. **5**(1), 23–32 (1988). https://doi.org/10.1109/52.1991
18. I. Ahmad, Y. Kwok, On exploiting task duplication in parallel program scheduling. IEEE Trans. Parallel Distrib. Syst. **9**(9), 872–892 (1998). https://doi.org/10.1109/71.722221
19. K. Ding, S. Zheng, Y. Tan, in *Genetic and Evolutionary Computation Conference, GECCO '13, Amsterdam, The Netherlands, July 6-10, 2013*. A GPU-based parallel fireworks algorithm for optimization, (2013), pp. 9–16. https://doi.org/10.1145/2463372.2463377
20. Y. Tan, Y. Zhu, in *Advances in Swarm Intelligence, First International Conference, ICSI 2010, June 12-15, 2010, Proceedings, Part I*. Fireworks algorithm for optimization (Springer, Beijing, 2010), pp. 355–364. https://doi.org/10.1007/978-3-642-13495-1_44
21. S. Zheng, A. Janecek, Y. Tan, in *Proceedings of the IEEE Congress on Evolutionary Computation, CEC 2013, June 20-23*. Enhanced fireworks algorithm (IEEE, 2013), pp. 2069–2077. https://doi.org/10.1109/CEC.2013.6557813
22. T. Yang, A. Gerasoulis, DSC: scheduling parallel tasks on an unbounded number of processors. IEEE Trans. Parallel Distrib. Syst. **5**(9), 951–967 (1994). https://doi.org/10.1109/71.308533
23. B. Cirou, E. Jeannot, in *30th International Workshops on Parallel Processing (ICPP 2001 Workshops), 3-7 September 2001*. Triplet: a clustering scheduling algorithm for heterogeneous systems (IEEE, Valencia, 2001), pp. 231–236. https://doi.org/10.1109/ICPPW.2001.951956
24. A. Gerasoulis, T. Yang, A comparison of clustering heuristics for scheduling directed acyclic graphs on multiprocessors. J. Parallel Distrib. Comput. **16**, 276–291 (1992). Elsevier. https://doi.org/10.1016/0743-7315(92)90012-C
25. S. Kim, J. Browne, in *Proceedings of the International Conference on Parallel Processing*. General approach to mapping of parallel computations upon multiprocessor architectures, vol. 3, (1988), pp. 1–8
26. A. Dogan, F. Özgüner, in *31st International Conference on Parallel Processing (ICPP 2002), 20-23 August 2002*. LDBS: a duplication based scheduling algorithm for heterogeneous computing systems (IEEE, Vancouver, 2002), pp. 352–359. https://doi.org/10.1109/ICPP.2002.1040891
27. B. Zhang, M. Zhang, Y. Zheng, in *Advances in Swarm Intelligence - 5th International Conference, ICSI 2014, October 17-20, 2014, Proceedings, Part I*. Improving enhanced fireworks algorithm with new Gaussian explosion and population selection strategies (Springer, Cham, 2014), pp. 53–63. https://doi.org/10.1007/978-3-319-11857-4_7
28. S. Chen, Y. Liu, L. Wei, B. Guan, PS-FW: A hybrid algorithm based on particle swarm and fireworks for global optimization. Comp. Int. Neurosc. **2018**, 6094685–1609468527 (2018). https://doi.org/10.1155/2018/6094685

29. Z. Zheng, A.K. Sangaiah, T. Wang, Adaptive communication protocols in flying ad hoc network. IEEE Commun. Mag. **56**(1), 136–142 (2018). https://doi.org/10.1109/MCOM.2017.1700323
30. Y. Tu, Y. Lin, J. Wang, Semi-supervised learning with generative adversarial networks on digital signal modulation classification. CMC-Comput. Mater. Continua. **55**(2), 243–254 (2018)
31. J.T. Zhou, H. Zhao, X. Peng, M. Fang, Z. Qin, R.S.M. Goh, Transfer hashing: from shallow to deep. IEEE Trans. Neural Netw. Learn. Syst. **PP**(99), 1–11 (2018). https://doi.org/10.1109/TNNLS.2018.2827036
32. Y. Lin, X. Zhu, Z. Zheng, The individual identification method of wireless device based on dimensionality reduction and machine learning. J. Supercomput. **5**, 1–18 (2017). https://doi.org/10.1007/s11227-017-2216-2
33. C. Shi, Z. Dou, Y. Lin, W. Li, Dynamic threshold-setting for RF-powered cognitive radio networks in non-Gaussian noise. Phys. Commun. **27**, 99–105 (2018). https://doi.org/10.1016/j.phycom.2018.02.001
34. J. Sun, W. Wang, L. Kou, Y. Lin, L. Zhang, Q. Da, L. Chen, A data authentication scheme for uav ad hoc network communication. J. Supercomput. **8**, 1–16 (2017). https://doi.org/10.1007/s11227-017-2179-3
35. Y. Lin, C. Wang, J. Wang, Z. Dou, A novel dynamic spectrum access framework based on reinforcement learning for cognitive radio sensor networks. Sensors. **16**(10), 1675 (2016). https://doi.org/10.3390/s16101675
36. Y. Lin, C. Wang, C. Ma, Z. Dou, X. Ma, A new combination method for multisensor conflict information. J. Supercomput. **72**(7), 2874–2890 (2016). https://doi.org/10.1007/s11227-016-1681-3
37. Q. Wu, Y. Li, Y. Lin, The application of nonlocal total variation in image denoising for mobile transmission. Multimedia Tools Appl. **76**(16), 17179–17191 (2017). https://doi.org/10.1007/s11042-016-3760-0
38. H. Wang, L.I. Jingchao, L. Guo, Z. Dou, Y. Lin, R. Zhou, Fractal complexity-based feature extraction algorithm of communication signals. Fractals-Compl. Geom. Patterns Scaling Nat. Soc. **25**(5), 1740008 (2017). https://doi.org/10.1142/S0218348X17400084
39. T. Liu, Y. Guan, Y. Lin, Research on modulation recognition with ensemble learning. EURASIP J. Wirel. Comm. Networking. **2017**, 179 (2017). https://doi.org/10.1186/s13638-017-0949-5
40. M.A. Khan, Scheduling for heterogeneous systems using constrained critical paths. Parallel Comput. **38**(4-5), 175–193 (2012). https://doi.org/10.1016/j.parco.2012.01.001
41. S.U. Jingnai, Pareto's principle and the optimization distribution of documentary information resources in colleges and universities. J. Fuqing Branch Fujian Normal Univ., 110–112 (2006)