IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received March 1, 2020, accepted March 19, 2020, date of publication March 27, 2020, date of current version April 27, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.2983728

A Self-Adaptive Parameter Control Method Based on Entropy and Security Market Line for Scheduling Problems

SHENGJIE SUN AND HUI LU, (Senior Member, IEEE)
School of Electronic and Information Engineering, Beihang University, Beijing 100191, China
Corresponding author: Hui Lu (mluhui@vip.163.com)

This work was supported by the National Natural Science Foundation of China under under Grant 61671041 and Grant 61101153.

ABSTRACT Scheduling problems, as one of the classic combinatorial optimization problems are essential issues in many fields. Various meta-heuristic algorithms have been adopted to solve scheduling problems. However, parameter control problem is still crucial to the performance of algorithms. In this paper, we propose a self-adaptive parameter control method based on entropy and security market line, fully considering the characteristics scheduling problems. It consists of two key parts: locus-entropy strategy and parameter-control strategy. Firstly, the entropy on each genetic locus is calculated to accurately evaluate the population status of scheduling algorithms. Then, a parameter-control strategy based on the conception of security market line is proposed to address the issue that the nature of multipeak in scheduling problems makes algorithms fall into local optimal solutions. The strategy maintains the solutions of good quality and eliminate the solutions of poor quality by using locus-entropy as feedback. Through our method, the balance between exploitation and exploration is kept in algorithms to perform well in scheduling problems with different dimensions and characteristics. These strategies are tightly linked to adjust parameters adaptively without introducing new parameters, so that meta-heuristic algorithms equipped with the proposed approach are able to find a better solution. Moreover, our parameter control method is universal for meta-heuristic algorithms. The proposed approach is hybrid with genetic algorithm and particle swarm optimization. The hybrid algorithms are first compared with the standard algorithms in multipeak benchmark functions, then with other variants of the standard algorithms in real-world single and multi-objective scheduling problems. The results demonstrate that the proposed approach is valid for different kinds of algorithms to enhance the performance of solving a variety of scheduling problems.

INDEX TERMS Parameter control, meta-heuristic algorithms, scheduling problems, information entropy, security market line.

I. INTRODUCTION

Scheduling problems are a kind of combinatorial optimization problems encountered in various fields, whose solutions are widely applied to help decision-makers gain huge benefits at lower costs under various constraint conditions [1]–[3]. The problems play an important role in relevant industries seeking to improve task execution efficiency and optimizing resource allocation.

Meta-heuristic algorithms are generally adopted to solve scheduling problems [4], [5]. They combine the random method with a local search to solve complex multipeak problems in a short period of time. The idea of meta-heuristic algorithms is inspired from the survival and habit of animals in nature or the evolutionary process in human society [6].

However, according to the No Free Lunch (NFL) theory [7], there is not a single algorithm that can solve all the problems. Many factors should be considered in solving practical problems. For one thing, the decision space varies greatly in spatial distribution between different dimensions and different types of scheduling problems. The population status cannot be accurately evaluated, reducing the performance of scheduling algorithms. For another, the multipeak property, which makes algorithms fall into local optimal solutions, is one of the most important inherent characteristics of scheduling problems. Multiple schemes in the decision space

The associate editor coordinating the review of this manuscript and approving it for publication was Ligang He.

This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see https://creativecommons.org/licenses/by/4.0/

may correspond to the same solution in the objective space. As a result, it is difficult to find an optimal solution from all the permutations and combinations.

Therefore, the parameter control plays a vital role in scheduling problems. It is one crucial issue in determining the performance of scheduling algorithms to solve problems, which has attracted many researchers these years. The parameter control mechanism is clearly divided into deterministic parameter control, adaptive parameter control and self-adaptive parameter control [8]. Deterministic parameter control follows a preset rule to assign new parameter values, often associated with the running time without any feedback from the search. Adaptive parameter control monitors the performance when algorithms are running. The change of the performance is used as a feedback to guide the parameter adjustment. Self-adaptive parameter control combines the search of optimal parameters with the search of optimal solutions. The parameters evolve with the solutions simultaneously by the method.

The parameter control has been investigated for long, but there are still many unsolved questions. First, some studies have focused on using statistical properties, such as average and variance, as the feedback to adjust parameters [9]. However, these statistics are unstable in a single run of an algorithm. Therefore, using them as feedback will lower the accuracy of the parameter control. Second, only the status of whole population and each individual is taken into consideration. It is far from enough when facing the high-dimensional real-word problems. The status of some certain genetic loci must be paid attention to make algorithms perform better. Third, universal parameter control methods are rare. Most methods are tailored to specific algorithms. As a result, they have difficulties in generalization. When the algorithm is changed, the parameter control method must also be changed, which is troublesome.

In this paper, we investigate a self-adaptive parameter control method for scheduling problems. It consists of two key parts, including a locus-entropy strategy and a parameter-control strategy. The parameters are adjusted adaptively based on the entropy of each genetic locus and the conception of security market line (SML). It is a universal method, which can be hybrid with meta-heuristic algorithms. Through the proposed method, the status of algorithms is evaluated objectively and precisely, and the parameters are adjusted timely and efficiently. The balance between exploitation and exploration is kept in algorithms to perform well in scheduling problems with different dimensions and different characteristics. These strategies are tightly linked in adjusting parameters adaptively without introducing new parameters. As a result, algorithms equipped with the proposed approach release excellent performance and find a better solution to solve scheduling problems.

The main contributions of the paper are described below.

First, in order to track the status of population objectively and precisely when facing various scheduling problems with different dimensions and distribution, we propose a locus-entropy strategy to evaluate the conditions of different genetic loci. The entropy is a concept in statistics used to measure the uncertainty of a random event, and to describe the average amount of information contained in each message received [10]. In meta-heuristic algorithms, each gene in the next generation is updated by the alleles of several individuals in the current population. If an allele in the population does not converge or converges prematurely, it will not be conducive to a better solution in the next generation. Through the strategy, the phenomenon of prematurity and non-convergence of certain genetic loci can be prevented by the locus-entropy. The influence of the randomness of meta-heuristic algorithms can be avoided.

Second, in order to address the issue that the nature of multipeak in scheduling problems makes algorithms fall into local optimal solutions, a parameter-control strategy based on SML is proposed to adjust all parameters adaptively. SML reflects the systematic risk of portfolio investment in finance and represents the relationship between risk and return, which together determine the price of the security [11]. We convert the relationship among systemic risk, investor-required rate of return, and securities pricing in economics into the relationship among population volatility, the goal of convergence and the setting of parameters in our method. The exploitation ability of the genetic loci with less entropy is enhanced while the genetic loci with larger entropy focuses on exploration through SML. It can maintain the solutions of good quality and eliminate the solutions of poor quality. In the meantime, parameters are set based on SML, according to the performance of each individual.

A set of experiments are performed to verify the validity of the proposed self-adaptive parameter control method. The approach is hybrid with both genetic algorithm (GA) and particle swarm optimization (PSO). Firstly, the hybrid algorithms are compared with the standard algorithms in multipeak test functions with different dimensions which are similar to the scheduling problem in the attribute to verify validity. Then, the algorithms are compared with other variants of the standard algorithms in both single and multiobjective real-world problems, including test task scheduling problem (TTSP), flexible job-shop scheduling problem (FJSP) and parallel machine scheduling problem (PMSP). To compare the performance of these algorithms more fairly, we use the iterations and the maximum running time as the termination conditions separately. The results indicate that the approach helps meta-heuristic algorithms find better solutions in different types of scheduling problem regardless of the termination conditions. The larger the scale of the problems is, the more significant the effect of the improvement is.

The remainder of this paper is organized as follows. The related work is reviewed in Section 2. In Section 3, we introduce the concept of scheduling problem. In Section 4, the Framework of the self-adaptive method based on entropy and SML is described in detail. In Section 5, the performance is evaluated in multipeak benchmark functions, and both single and multiobjective scheduling problems.

II. RELATED WORK

There are a wide variety of meta-heuristic algorithms introduced to scheduling problems. Parameters control of these algorithms is one of the most long-standing big challenges and the research has never stopped since the last century. Generally, parameter setting of meta-heuristic algorithms can be divided into parameter tuning and parameter control [12]. In parameter tuning, the values of the parameters are static after being set up, while in parameter control, the parameter values are constantly changing when the algorithm runs [13], [14]. One of the major disadvantages of parameter tuning, compared to parameter control, is the lack of flexibility and relevance, as population changes dynamically during the search process while parameters remain unchanged, preventing the population from being updated accurately. Contrary to parameter tuning, parameter control is far from being thoroughly researched [15]. The methods of parameter control have generally been classified into three categories since 1999 [8], including deterministic, adaptive and self-adaptive methods. The classifying method has been widely adopted by scholars.

Deterministic parameter control refers to adjusting the parameters according to some deterministic rules, such as fixed schedules or iterations of algorithms [16], [17]. The rules adjust the parameters in a fixed and predetermined way without using any feedback from the search process. Due to the strong randomness of meta-heuristic algorithms, it is difficult to predict when the algorithms will converge before they run, so this method is simple and inaccurate but less timely and easier to control. Typically, as the number of iterations increases, the mutation demand is reduced in the genetic algorithm to promote the convergence of the algorithm. Joines and Houck proposed a dynamic penalty function, where the penalties are increased according to an external cooling scheme [18]. Costa *et al.* investigated the potential of a simplistic deterministic control schedule of monotonous growth or shrinkage to adjust the population size [19]. For scheduling problems, some deterministic methods inspired by the Boltzmann distribution were used to guarantee asymptotic convergence to the global optima [20]. The main disadvantages of deterministic parameter control are that the method requires a significant amount of manual setting and the values are dependent of the special problems [21].

Adaptive parameter control means that the parameters are adjusted by the state of the algorithm at runtime, such as the fitness of solutions [22], [23]. The direction and magnitude of the parameter adjustment are determined by taking one or more states as feedback. The parameter values are adjusted based on the feedback. The process is shown in Fig. 1. An interesting model of adaptive parameter control is presented, in which the optimization process and the control process are divided into four steps, including feedback collection, parameter effect assessment, parameter quality attribution and parameter value selection [9]. McGinley *et al.* used the diversity of the population as the feedback to adjust

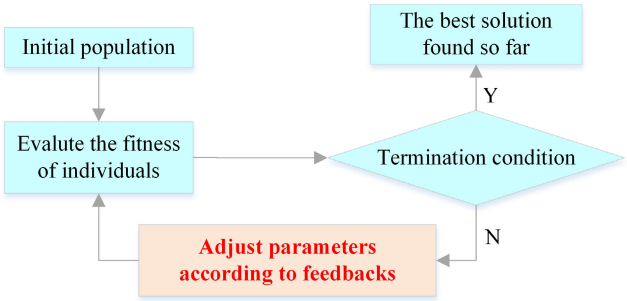

FIGURE 1. The basic process of adaptive parameter control.

the tournament size, reaching a bigger tournament size in the case of a high diversity and a smaller tournament size in the case of a low diversity [24]. Srinivas and Patnaik proposed an adaptive approach to control both mutation and crossover rates in genetic algorithms [25]. To ensure the diversity and convergence of the population at the same time, the differences in the parent's fitness with the best convergence and the grade of convergence were regarded as the feedback to define the crossover rates when the offspring is produced. The disadvantage of this method is that it is hard to establish a direct connection between the parameters in the next generation and the parameters in the current generation, leading to great changes of the parameters and instability of the algorithms.

Self-adaptive parameter control means that the search for the optimal solution is combined with parameter adjustment, compared to adaptive parameter control. It was originally introduced by Rechenberg and Schwefel in 1974 [26]. The method encodes the values of the parameters into the chromosomes and evolves with the solutions simultaneously [27], [28]. The feedback mechanism is the core of both adaptive and self-adaptive parameter control, directly affecting the results of parameter control. The feedback collection mechanisms can be divided into six categories, including collecting the information of the fitness function, collecting the difference among solutions based on entropy, collecting the solution components of a single solution, collecting the diversity of the solution components in a population of solutions, collecting the violations of constraints and collecting the computations in the optimization process. In the early stages of research, Davis proposed that the performance of the algorithm is estimated as the difference in fitness between a created solution and its parent [29]. In recent years, Maturana and Saubion used the weighted sum of the fitness improvement as the feedback of the algorithms [30]. The diversity-guided evolutionary algorithm used a distance-to-average-point measure to alternate between the phases of exploration and exploitation [31].

However, the parameter control methods still face several difficulties. First, some studies have focused on using statistical properties, such as average and variance, as the feedback to control parameters. According to Aldeida Aleti, more than 86% of the methods of adaptive parameter control use the

fitness improvement of the solutions as the feedback of the performance of the algorithm [9]. However, the statistics in these methods are unstable, so that these methods cannot completely avoid the effects of the randomness of the meta-heuristic algorithms. Second, failing to adjust the state of some certain genetic loci and only considering the whole population or each individual lead to a lack of diversity in the population. The performance of each gene will directly affect the quality of the final solution. If the genes of each individual are not taken seriously, the individual will not be close to the optimal solution once some of the genetic loci converge prematurely. Third, the methods of parameter control are always tailored. The generic control methods are rarely studied, because little is known about the joint effects of control mechanisms [21]. Although some scholars have combined the variation control with the selection control in the genetic algorithm so that the exploring individuals created under exploration mode with aggressive mutation are given a fair chance to survive and reproduce, the parameter control methods for different meta-heuristic algorithms remain to be studied [24].

Our method can effectively solve these difficulties to solve scheduling problems. The locus-entropy is calculated to accurately obtain the state of the population. It is targeted to individuals compared to deterministic and adaptive parameter control. Meanwhile, we link the random volatility of the population with systemic risk in finance and use market capital line theory to adjust the parameters adaptively. Furthermore, the method provides a trade-off between exploitation and exploration for algorithms to search for a better solution.

III. SCHEDULING PROBLEMS

Scheduling problems have been widely applied in various fields, such as the manufacturing industry, service industry, cloud computing and Internet of Things. They involve finding a grouping, selection, ordering, or assignment of a discrete, finite set of objects that satisfies given conditions, typically the maximum or minimum. The combination of task sequencing, instrument allocation, program selection and even task process in a certain scheduling problem produces a combined explosion effect, so meta-heuristic algorithms are often used to find near optimal solutions to these problems.

Three types of problems involved in the paper will be introduced in turn, including test task scheduling problem (TTSP), flexible job-shop scheduling problem (FJSP) and parallel machine scheduling problem (PMSP). Our target is to minimize the total time as much as possible through reasonable arrangements for all these problems.

The TTSP means that a fixed number of tasks are tested on corresponding instruments after being sorted and each being selected a certain scheme for [32], [33]. As the scheme is selected, the instrument the task occupies and the time it spends are determined. Since an instrument can only execute one task, the total time spent on testing this set of tasks can be calculated after the sequence of tasks and the selection of schemes are confirmed. In addition, each task must be completed without interruption once it starts and some instruments may be occupied at the same time for one test task.

The FJSP is similar to TTSP. A set of independent jobs is ready to be processed on certain machines. There are different sequences for each job, which can be executed on any machine [34]. The processing time of each sequence depends on the selected machine. The order of jobs and the choice of instruments for each sequence are two key factors that affect the total processing time.

Compared with the TTSP and FJSP, PMSP is relatively simple. There are a certain number of jobs and machines. For each job, only one process should be executed on any machine.

These three problems can be represented by a unified mathematical model as follows. The problem description, variable definitions and constraint conditions are taken into consideration [35].

A. SETS

The sets of the model are peresented as follows.

$J = \{J_i\}_{i=1}^{N}$: The set of jobs or tasks. (N is the number of jobs.)

$M = \{M_k\}_{k=1}^{H}$: The set of machines or instruments. (H is the number of machines.)

$S_i = \left\{s_i^n\right\}_{n=1}^{u_i}$: The scheme set that can be chosen by task i. (u_i is the scheme number of job or task i.)

$O_i = \left\{O_i^j\right\}_{j=1}^{v_i}$: The operation set of task i. (v_i is the operation number of task i.)

B. PARAMETERS AND VARIABLES

Parameters and variables of the model are peresented as follows.

$\left(i, s_i^n\right)$: The combination that task i adopts scheme s_i^n.

$Pt_{jk}^{\left(i,s_i^n\right)}$: The process time of the j^{th} operation of $\left(i, s_i^n\right)$ on machine k.

$a_{jk}^{\left(i,s_i^n\right)}$: The allocation variable. If the j^{th} operation of $\left(i, s_i^n\right)$ can be processed on machine k, $a_{jk}^{\left(i,s_i^n\right)}$ is equal to 1, otherwise 0.

$St_{jqk}^{\left(i,s_i^n\right)\left(r,s_r^n\right)}$: The setup time of machine k from the j^{th} operation of $\left(i, s_i^n\right)$ to the q^{th} operation of $\left(r, s_r^n\right)$.

$Ft_{jk}^{\left(i,s_i^n\right)}$: The completion time of the j^{th} operation on machine k.

$Y_{is_i^n}$: The adopted scheme of a task. If job i adopts scheme s_i^n, $Y_{is_i^n}$ is equal to 1, otherwise 0.

$X_{jk}^{\left(i,s_i^n\right)}$: If the j^{th} operation of $\left(i, s_i^n\right)$ is proposed on machine k, $X_{jk}^{\left(i,s_i^n\right)}$ is equal to 1, otherwise 0.

$R_{jqk}^{\left(i,s_i^n\right)\left(r,s_r^n\right)}$: The priority variable. If the j^{th} operation of $\left(i, s_i^n\right)$ precedes the q^{th} operation of $\left(r, s_r^n\right)$ on machine k, $R_{jqk}^{\left(i,s_i^n\right)\left(r,s_r^n\right)}$ is equal to 1, otherwise 0.

L: A large integer.

C. CONSTRAINT CONDITIONS

When the scheme s_i^n is chosen by the task i, the $(j+1)^{th}$ operation can be started only after the completion of the j^{th} operation. The formula of the Constraint can be expressed as follows.

$$Ft_{(j+1)k}^{(i,s_i^n)} - Ft_{jm}^{(i,s_i^n)} + L\left(1 - a_{(j+1)k}^{(i,s_i^n)} X_{(j+1)k}^{(i,s_i^n)}\right) \geq Pt_{(j+1)k}^{(i,s_i^n)},$$
$$i = 1, \ldots, N; \quad n = 1, \ldots, u_i; \; j = 1, \ldots, v_i - 1;$$
$$k = 1, \ldots, M; \quad m = 1, \ldots, H \tag{1}$$

$$Ft_{qk}^{(r,s_r^n)} - Ft_{jk}^{i,s_i^n} + L \cdot R_{jqk}^{(i,s_i^n)(r,s_r^n)} \geq Pt_{qk}^{(r,s_r^n)} X_{qk}^{(r,s_r^n)}$$
$$+ St_{jqk}^{(i,s_i^n)(r,s_r^n)} X_{jk}^{(i,s_i^n)}, \quad i = 1, \ldots, N-1;$$
$$r = i+1, \ldots, N; \quad s_i^n = s_i^1, \ldots, s_i^{u_i};$$
$$s_r^n = s_r^1, \ldots, s_r^{u_r}; \quad j = 1, \ldots, v_i; \; q = 1, \ldots, v_r;$$
$$k = 1, \ldots, H \tag{2}$$

$$Ft_{qk}^{(r,s_r^n)} - Ft_{jk}^{i,s_i^n} + L\left(1 - R_{jqk}^{(i,s_i^n)(r,s_r^n)}\right) \geq Pt_{qk}^{(r,s_r^n)} X_{qk}^{(r,s_r^n)}$$
$$+ St_{jqk}^{(i,s_i^n)(r,s_r^n)} X_{jk}^{(i,s_i^n)}, \quad i = 1, \ldots, N-1;$$
$$r = i+1, \ldots, N;$$
$$s_i^n = s_i^1, \ldots, s_i^{u_i}; \quad s_r^n = s_r^1, \ldots, s_r^{u_r}; \quad j = 1, \ldots, v_i;$$
$$q = 1, \ldots, v_r; \quad k = 1, \ldots, H \tag{3}$$

As shown in Eq. (2) and (3), any two operations cannot be processed on one machine at the same time. If the operations from two tasks are arranged on the same machine, the setup time of the machine is related to the sequence of operations.

Only one scheme can be adopted to solve each task while each operation of a scheme can only be assigned to one machine for processing. The formulas of the Constraint are presented as follows.

$$\sum_{n=1}^{u_i} Y_{is_i^n} = 1, \quad i = 1, \ldots, N \tag{4}$$

$$\sum_{n=1}^{u_i} a_{jk}^{(i,s_i^n)} X_{jk}^{(i,s_i^n)} = Y_{is_i^n}, \quad i = 1, \ldots, N; \; n = 1, \ldots u_i;$$
$$j = 1, \ldots, v_i; \; k = 1, \ldots H \tag{5}$$

$$Ft_{1k}^{(i,s_i^n)} \geq Pt_{1k}^{(i,s_i^n)} X_{1k}^{(i,s_i^n)}, \quad i = 1, \ldots, N;$$
$$n = 1, \ldots u_i; \quad k = 1, \ldots H \tag{6}$$

$$Ft_{jk}^{(i,s_i^n)} \leq L \cdot X_{jk}^{(i,s_i^n)}, \quad i = 1, \ldots, N;$$
$$n = 1, \ldots u_i; \quad j = 1, \ldots, v_i;$$
$$k = 1, \ldots H \tag{7}$$

Eq. (6) ensures that the completion time of the first operation of each task should be no less than its processing time. According to Eq. (7), if the j^{th} operation of one task is not assigned to machine k, the completion time of this operation on this machine is 0.

$$C_i \geq \sum_{k=1}^{H} \sum_{n=1}^{u_i} Ft_{v_ik}^{(i,s_i^n)} X_{v_ik}^{(i,s_i^n)}, \quad i = 1, \ldots, N \tag{8}$$

$$C_i \leq D_i, \quad i = 1, \ldots, N \tag{9}$$

$$Ft_{jk}^{(i,s_i^n)} \geq 0; \quad Y_{is_i} = 0, 1; \; X_{jk}^{(i,s_i^n)} = 0, 1,$$
$$i = 1, \ldots, N; \; n = 1, \ldots u_i; \; j = 1, \ldots, v_i;$$
$$k = 1, \ldots H \tag{10}$$

$$R_{jkq}^{(i,s_i^n)(r,s_r^n)} = 0, 1, \quad i = 1, \ldots, N-1;$$
$$r = i+1, \ldots, N; \quad s_i^n = s_i^1, \ldots, s_i^{u_i};$$
$$s_r^n = s_r^1, \ldots, s_r^{u_i};$$
$$j = 1, \ldots, v_i; \quad q = 1, \ldots, v_r;$$
$$k = 1, \ldots, H \tag{11}$$

Eq. (8) determines the completion time of each task based on other variables. The condition that each task can be completed before the deadline is ensured by Eq. (9). The range of variables is defined in Eq. (10) and (11).

D. OBJECTIVE FUNCTION

The makespan is the objective function and can be written as follws in Eq. (12).

$$F_{obj} = \min\{\max(C_i)\} \tag{12}$$

IV. THE FRAMEWORK OF THE SELF-ADAPTIVE PARAMETER CONTROL METHOD BASED ON ENTROPY AND SML

The framework of the self-adaptive parameter control method based on entropy and SML is shown in Fig. 2. The core of the approach includes a locus-entropy strategy and a parameter-control strategy based on SML. The state of different genetic loci is evaluated through the locus-entropy firstly to track the status of population. It avoids the impact of randomness of the algorithm and tracks the development of each gene in time to prevent premature and nonconvergence on a certain gene, because the entropy is a measure of uncertainty, not specific content. Then, the parameters are adjusted under the guidance of the conception of SML to maintain the solutions of good quality and eliminate the solutions of poor quality. In the strategy, the relationship among systemic risk, investor-required rate of return, and securities pricing in economics is converted into the relationship among population volatility, the goal of convergence and the setting of parameters in SML.

These strategies are tightly linked to adjust parameters adaptively without introducing new parameters according to the changes in population status during algorithm iteration. Meanwhile, the method is universal, which means it can be integrated with various meta-heuristic algorithms to improve the algorithms' capability to find multiple optimal solutions. Through these strategies, not only the convergence of population is accurately tracked, but also the diversity of the population is well preserved. The adjusted parameters can guide the population iteration to find better solutions more effectively in scheduling problems. The strategies of our method are proposed in turn in the following subsections, along with the meta-heuristic algorithm hybrid with the approach presented in Section 4.3.

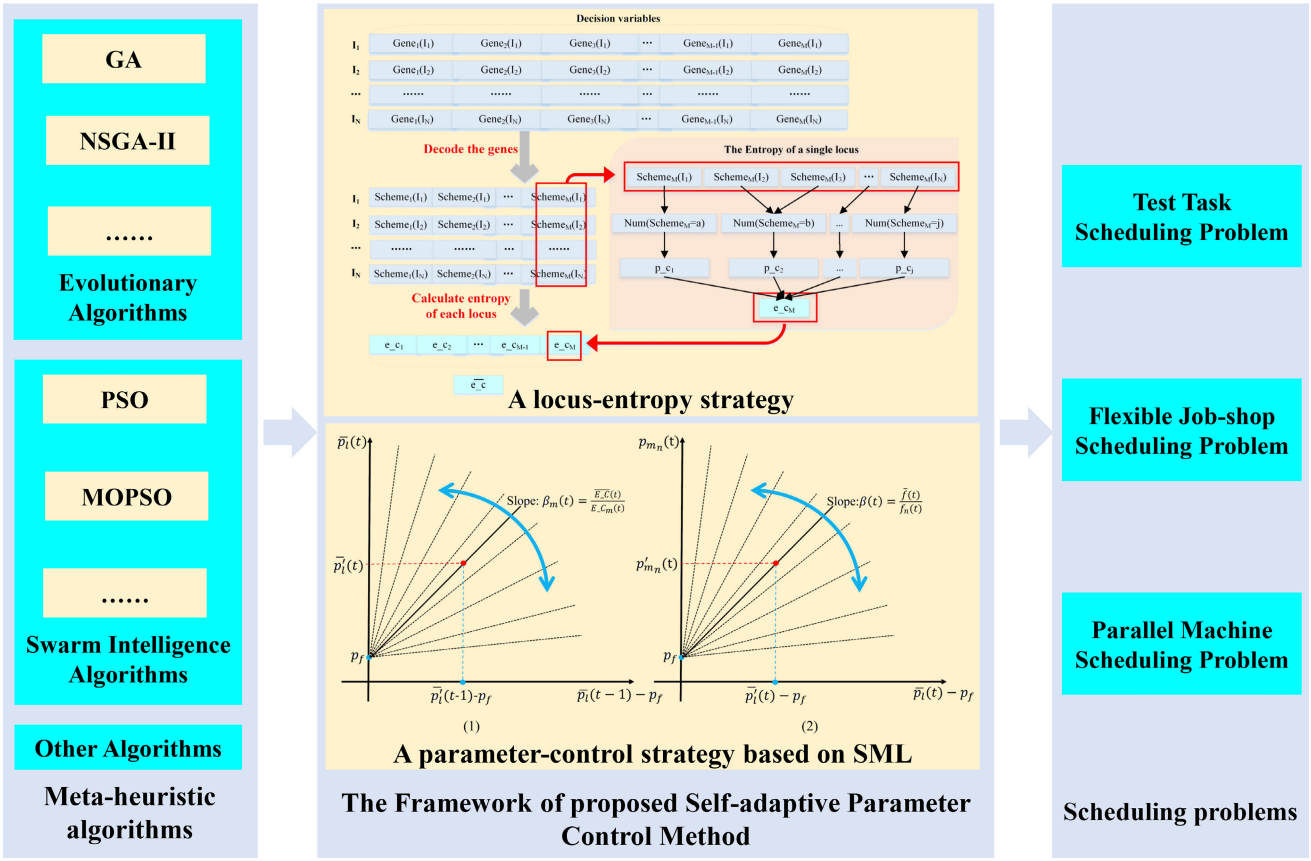

FIGURE 2. The framework of self-adaptive parameter control in meta-heuristic algorithm based on entropy and security market line.

A. A LOCUS-ENTROPY STRATEGY

The scheduling algorithms find solutions by searching, so the solutions of each generation are random. It is hard to evaluate the status of the population objectively when facing various scheduling problems with different dimensions and distribution. The lack of supervision and adjustment of some genes will affect the performance of the algorithm to solve scheduling problems. The information entropy provides a fairer way to assess the status of each genetic locus effectively and ensure the diversity of all loci in our method.

The concept of entropy originated in physics was used to measure the degree of disorder of a thermodynamic system. In information theory, entropy, as a measure of uncertainty, is the average amount of information contained in each message received. The probability distribution of events and the amount of information of each event constitute a random variable. The meaning of this random variable is the average amount of information generated by this distribution. The higher the entropy, the more information can be transmitted, and the lower the entropy, the less information is transmitted. Entropy only takes into account the probability of observing a specific event, so it can reflect the status of algorithms objectively. The information it encapsulates is the underlying probability distribution, not the meaning of the events themselves.

In scheduling problems, the information on a gene of an individual represents a scheme of a single task, the number of whose schemes is limited after the genes are decoded. A method of assessing the status of each genetic locus based on entropy is provided below. The flow chart is presented in Fig. 3.

Provided that there are M decision variables in each individual, $Scheme_{m^{th}}(i^{th}), (1 \leq m \leq M)$ is the scheme on the m^{th} gene of the i^{th} individual. It can be decoded according to specific problems. For example, if there are 20 tasks to be tested and 8 machines available in a test task scheduling problem, the decision variables of an individual are composed of 20 genes, as each gene locus corresponds to a task. After decoding, the scheme $Scheme_{m^{th}}(i^{th})$ on the m^{th} gene of the i^{th} individual contains the task information of the execution order and machine used.

According to the decoded schemes, the locus-entropy, is shown on the right side of Fig. 3. If j is the number of different schemes of all individuals on the m^{th} genetic locus, j different schemes can be denoted by $Num(Scheme_{m^{th}} = k), (1 \leq k \leq j)$. The probability $p\_c_k, (1 \leq k \leq j)$ of the scheme $Scheme_{m^{th}} = k$ can be easily calculated. The entropy

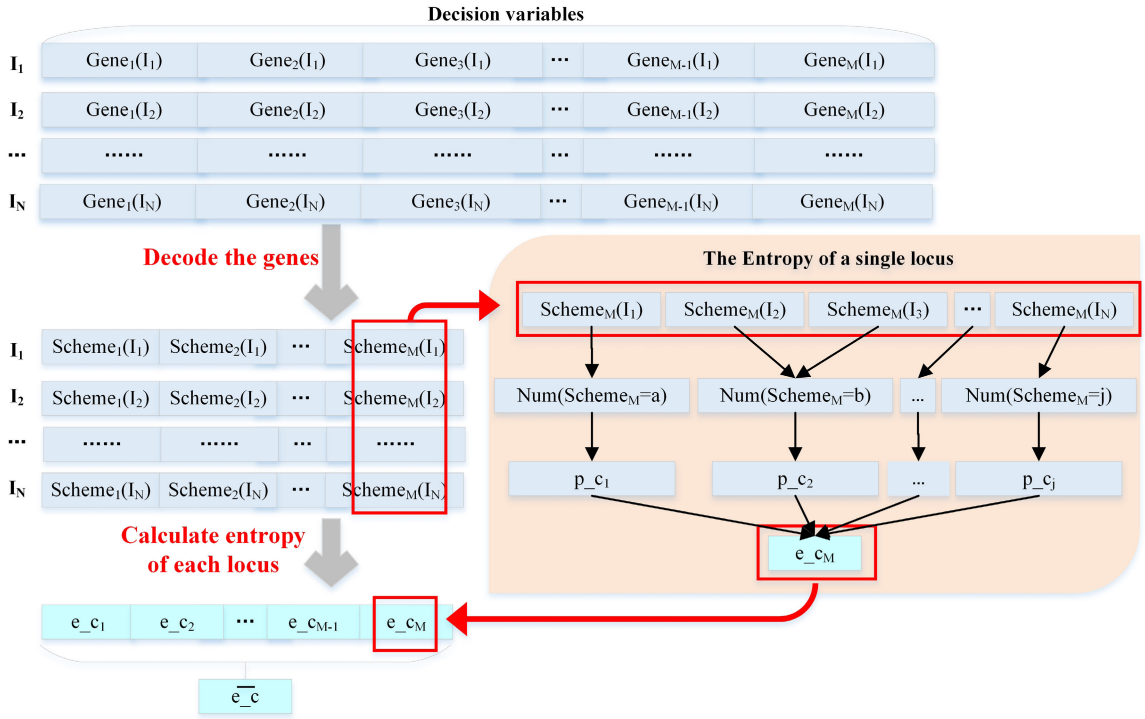

FIGURE 3. The flow chart of assessing the status of genetic loci based on entropy.

of m^{th} genetic locus can be represented by e_c_m.

$$e\_c_m = \sum_{k=1}^{j} p\_c_k \cdot \log_2(p\_c_k)$$

The average entropy of whole loci can be represented by $\overline{e\_c}$.

$$\overline{e\_c} = \frac{\sum_{m=1}^{M} e\_c_m}{M}$$

According to the definition of information entropy, the maximum of e_c_m is $log_2 j$, and the minimum is 0. In other words, once the number of schemes is determined, the range of e_c_m is also determined. Therefore, the value reflects the convergence degree accurately. If the value of e_c_m is lower than $\overline{e\_c}$, it means that the diversity on m^{th} genetic locus is poor and needs to be improved by increasing the exploring ability on the genetic locus. Otherwise, it means that the convergence on m^{th} genetic locus should be improved.

B. A PARAMETER-CONTROL STRATEGY BASED ON SML

After calculating the locus-entropy, the theory of security market line is applied to adjust parameters adaptively. The security market line is a conception in finance, which represents the relationship between risk and return, which determines the price of the security. In our method, SML is used to measure the relationship between system volatility and previous generation parameters, which determines the value of current parameters.

Security market line is the representation of the capital asset pricing model. It displays the expected rate of return of an individual security as a function of systematic, non-diversifiable risk. The risk of an individual risky security reflects the volatility of the return from security rather than the return of the market portfolio. The risk in the market portfolio reflects the systematic risk. SML is a line on which every point represents a certain portfolio composed of only risk-free assets and the market portfolio. The difference among these portfolios lies in the proportion invested in the market portfolio, which in turn determines the risk and return. Every correctly priced asset should find its risk and return profile in accordance with that of a point on SML. SML can be easily used to model and derive expected return from the assets or portfolio. The definition is shown in the following formula.

$$R_i = R_f - \beta_i(R_M - R_f)$$

R_i is the expected return on the security. R_f is the risk-free rate and represents the y-intercept of the SML. β_i is a non-diversifiable or systematic risk. It is the most important factor in SML. R_M is expected return on market portfolio. $R_M - R_f$ is known as market risk premium.

β_i, as an important factor in the SML equation, is a measure of the volatility, or systematic risk, of a security or a portfolio in comparison to the market as a whole. The market can

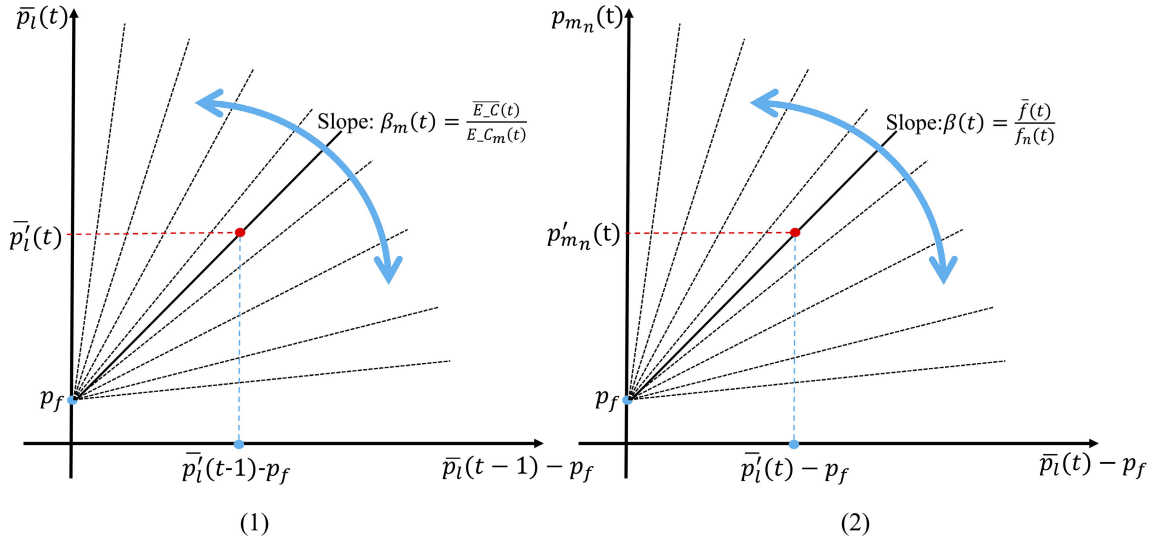

FIGURE 4. A parameter-control strategy based on SML.

be considered as an indicative market index or a basket of universal assets.

If $\beta_i = 1$, it means that the stock has the same level of risk as the market. A β_i greater than 1 represents an asset riskier than market. A β_i less than 1 represents an asset less risky than market. Although β_i provides a single measure to compare the volatility of an asset with the market, it does not remain constant with time.

In our method, we convert the relationship among systemic risk, investor-required rate of return, and securities pricing in economics into the relationship among population volatility, the goal of convergence and the setting of parameters. Based on the conception of SML, individuals with better fitness are more likely to be retained to the next generation in the population, while individuals with poor fitness are more likely to be changed or replaced in the algorithms, The strategy will be actively guided to find solutions. A parameter on a single locus is determined through the following two steps.

Firstly, the average parameter level of the whole population at each locus is calculated, as shown in Fig. 4(1). The formula is as follows.

$$\overline{p_l}(t) = p_f + \beta_m(t) \cdot \left[\overline{p_l}(t-1) - p_f\right]$$

$\overline{p_l}(t)$ is the average parameter level of the whole population at each locus in t^{th} generation. p_f is the basic value of the parameter, and each parameter cannot be less than p_f. $\beta_m(t)$ represents the convergence of the m^{th} gene locus.

$$\beta_m(t) = \frac{\overline{e\_c}(t)}{e\_c_m(t)}$$

$\overline{e\_c}(t)$ represents the average entropy of whole loci, while $e\_c_m(t)$ is entropy of the m^{th} genetic locus. If $\beta_m(t)$ is larger than 1, it means that the convergence on the gene locus is better and the diversity needs to be improved, so $\overline{p_l}(t)$ on the m^{th} genetic locus in the t^{th} generation will be much larger. Otherwise, if $\beta_m(t)$ is less than 1, it means that the diversity on the gene locus is better and the convergence needs to be improved, so $\overline{p_l}(t)$ will be smaller.

Secondly, the parameter value at the m^{th} locus of the n^{th} individual is calculated, which is shown in Fig. 4(2). The formula is as follows.

$$p_{m_n}(t) = p_f + \beta(t) \cdot \left[\overline{p_l}(t) - p_f\right]$$

$p_{m_n}(t)$ is the parameter value at the m^{th} locus of the n^{th} individual. $\beta(t)$ represents the volatility of the n^{th} individual. The formula is as follows.

$$\beta(t) = \frac{\bar{f}(t)}{f_n(t)}$$

$\bar{f}(t)$ is the average fitness of the t^{th} generation in the population, while $f_n(t)$ presents the fitness of the n^{th} individual in the t^{th} generation. The larger $\beta(t)$ is, the larger $p_{m_n}(t)$ will be. The strategy helps us keep best solutions of the scheduling problems and change the value of the poor solutions based on the information of entropy and fitness value. The method allows parameters to be adaptively adjusted, so the algorithms are more likely to find a better solution while maintaining the balance between exploration and exploitation.

C. META-HEURISTIC ALGORITHMS HYBRID WITH THE PROPOSED METHOD BASED ON ENTROPY AND SML

In this subsection, the process of meta-heuristic algorithms hybrid with the proposed method based on entropy and SML is presented. The pseudo-code of the whole process is given in Table 1. The flow chart is shown in Fig. 5.

The structure of the standard meta-heuristic algorithms is not changed by the proposed method. The proposed method only uses the information of decision variables and the quality of each individual for parameter adjustment. As a result,

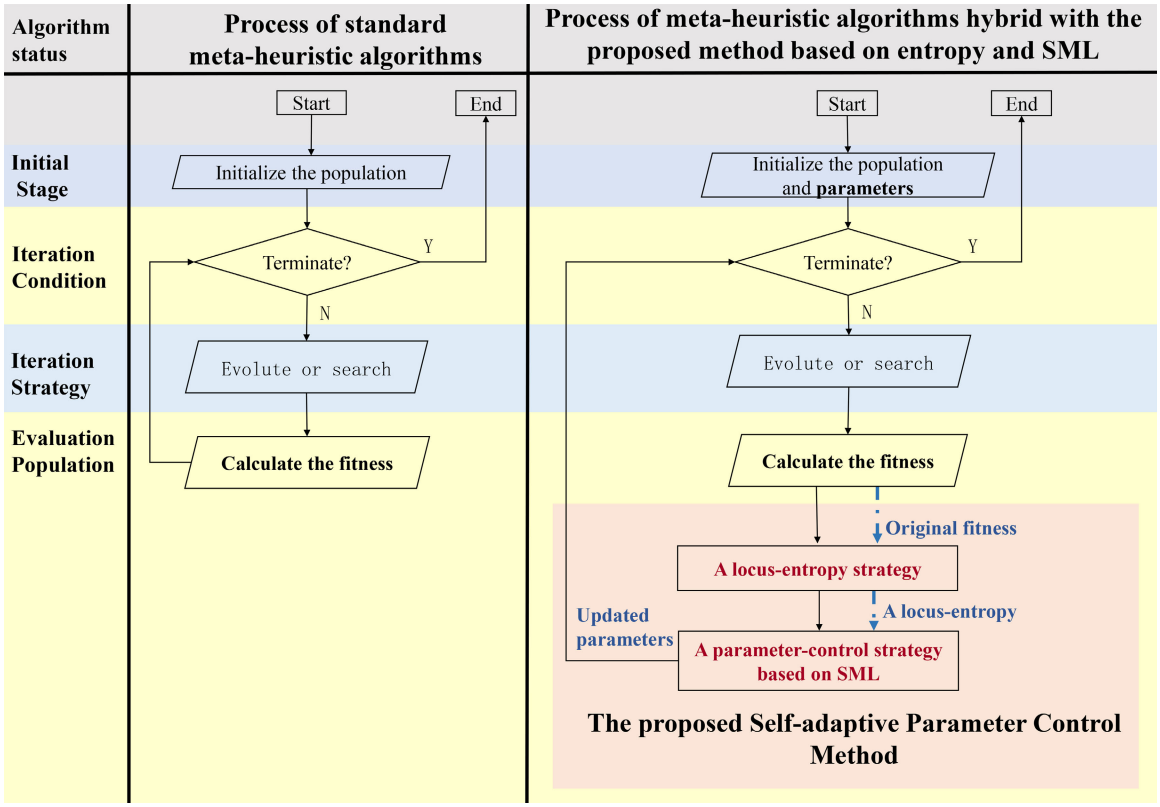

FIGURE 5. The flow chart of meta-heuristic algorithms hybrid with the proposed method based on entropy and SML.

TABLE 1. The pseudo code of meta-heuristic algorithms hybrid with the proposed method based on entropy and SML.

The whole process of meta-heuristic algorithms hybrid with the proposed method based on entropy and SML.
Step 1: Initialization.
Generate the initial population randomly.
Step 2: Iteration. (t is the label of the iteration.)
For $t = 1$ to N.
2.1 Update each individual according to special search operations for different algorithms.
2.2 Calculate the original fitness by the fitness function.
2.3 Calculate the entropy of each genetic locus and the average entropy of all loci.
2.4 Update the parameters based on security market line and the feedback.
End For
Step 3: Termination. Terminate the algorithm and output the best solution found so far, if the termination condition is satisfied.

the versatility of the method is guaranteed. Each step is tightly connected, enabling parameters to adaptively adjust according to the population status.

V. EXPERIMENTS

We have conducted a series of experiments on multipeak benchmark problems and scheduling problems to evaluate the proposed methods based on entropy and SML hybrid with GA and PSO (ESGA and ESPSO). The algorithms equipped with our method is first compared with the standard meta-heuristic algorithms in the benchmark problems to show that it is valid. Then, our method is compared with some other methods in three typical scheduling problems to verify the performance, including FJSP, PMSP and TTSP.

Since the application of meta-heuristic algorithms is very extensive, different users have different requirements. In other words, different users have different tolerances for time and performance. Therefore, to evaluate algorithms more objectively and fairly, all algorithms are compared at the same computational time and the same iterations in the following experiments.

All the noncontrolled parameters are fixed in the experiments. The population size is 100. If the stopping condition of the algorithm is the computational time, the maximum computational time is 1 second in the case of removing the decoding time due to different decoding times for different problems. If the number of generations is used as the stopping condition, the maximum number of generations is 1000. In addition, the mutation rate and crossover rate of the standard GA are 0.1 and 0.9, while the inertia weight and acceleration factor of the original PSO are 0.9 and 2.

All the experiments are conducted on a 64-bit computer with two Intel(R) Xeon(R) CPU E5-2620 @ 2.1 GHz and 128 GB of RAM on Windows10 Enterprise Edition. All the algorithms are implemented in MATLAB 2018a. They are all serial computing and no parallel computing is used.

A. EXPERIMENTS ON MULTIPEAK BENCHMARK PROBLEMS

Six benchmark problems are used to evaluate the performance of the proposed method. They are the Ackley function, Griewank function, Michalewicz function, Rastrigin function, Schwfel function and Levy function. The features of these functions are similar with scheduling problems, including scalability, separability, multimodality and local optimality. To measure the effectiveness in problems of different scales, the experiments are conducts on the benchmarks when the dimensions are 5, 10 and 30.

The function expression, search range and optimal value of each multipeak benchmark problem can be found in [36]. Functions become increasingly complicated as the number of dimensions increases. Here, we use the mean of all the best fitness values found in 500 times (MBF) as the evaluation index. ESGA is compared with the original GA, while ESPSO is compared with the original PSO.

To compare the performance of different algorithms more intuitively, the MBF at the same computational time and the same generation is fitted to a curve in the figures, and the mean of the best fitness found when meeting the termination condition is marked at the end of the curves. In each figure, the horizontal axis is the iterations or time and the vertical axis is the MBF. The solid red line represents the convergence curve of ESGA or ESPSO, while the blue dashed line represents the convergence curve of GA or PSO.

1) EXPERIMENT ON GA AND ESGA

When the dimension is 5, the comparisons of the MBFs between GA and ESGA are shown in Fig. 6 and Fig. 7. Regardless of using the time or number of iterations as the termination condition of the algorithm, ESGA always finds better solutions in all the problems than GA. In the meantime, the convergence speed of ESGA is faster than that of GA.

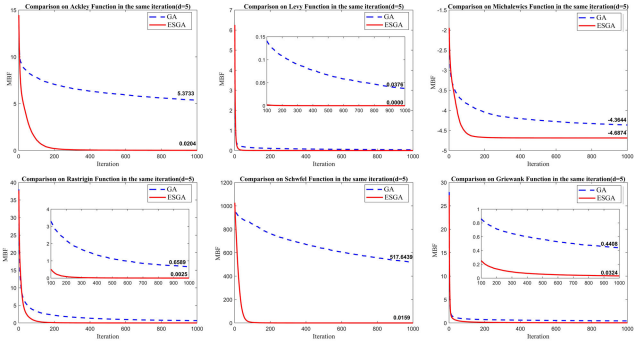

FIGURE 6. Comparison of MBFs in the same generation between GA and ESGA on benchmark problems when the dimension is 5.

In almost all the problems, ESGA approached the optimal solutions with less than 200 generations and 0.2 seconds. Among these problems, the Schwfel function exhibits largest difference in performance between ESGA and GA, while the Levy function shows the smallest.

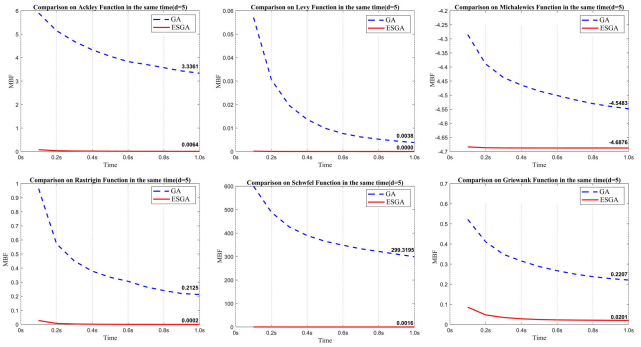

FIGURE 7. Comparison of MBFs in the same time between GA and ESGA on benchmark problems when the dimension is 5.

When the termination condition is the maximum iterations, the gaps between ESGA and GA are 5.3529, 0.0376, 0.323, 0.6564, 517.628 and 0.4084 in these functions. When the termination condition is the maximum computational time, the gaps between ESGA and GA are 3.3297, 0.0038, 0.1393, 0.2123, 299.3179 and 0.2006. Even though the dimension is not large, it is difficult for GA to jump out of the local optimal solutions.

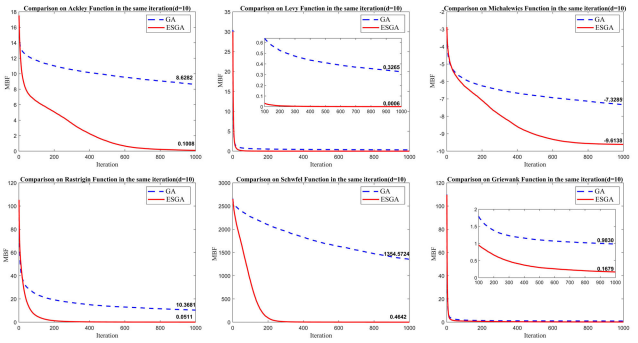

FIGURE 8. Comparison of MBFs in the same generation between GA and ESGA on benchmark problems when the dimension is 10.

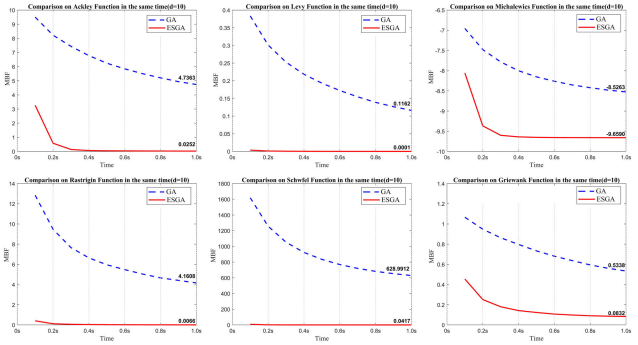

FIGURE 9. Comparison of MBFs in the same time between GA and ESGA on benchmark problems when the dimension is 10.

When the dimension is 10, the comparisons of MBFs between GA and ESGA are shown in Fig. 8 and Fig. 9. The MBFs of ESGA are still close to the optimal solutions for

these problems. It means that ESGA avoids falling into the local optimal solutions in most cases and constantly search for the global optimal solution. Meanwhile, our strategy will not suppress the convergence of the population. Although the dimension has become larger, ESGA can lock down to the optimal solutions and stay stable with less than 600 generations and 0.4 seconds. The gaps between ESGA and GA are further widened under all situations. For the Schwfel function, since the search space of the function is much larger than the other functions, ESGA performs especially better than GA. The differences of MBFs between them reach 1354.1082 in the same generation and 628.9495 in the same time. Although GA shows a downward trend, it is very difficult to find satisfactory solutions.

When the dimension is 30, the comparisons of MBFs between GA and ESGA are shown in Fig. 10 and Fig. 11. The difference between the performance of the two algorithms is apparent. The advantages of ESGA are further magnified in the high-dimensional test functions.

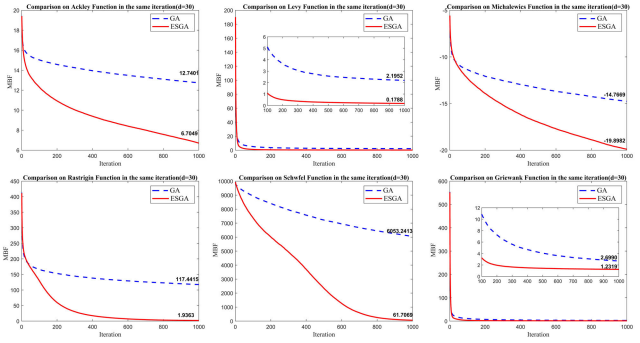

FIGURE 10. Comparison of MBFs in the same generation between GA and ESGA on benchmark problems when the dimension is 30.

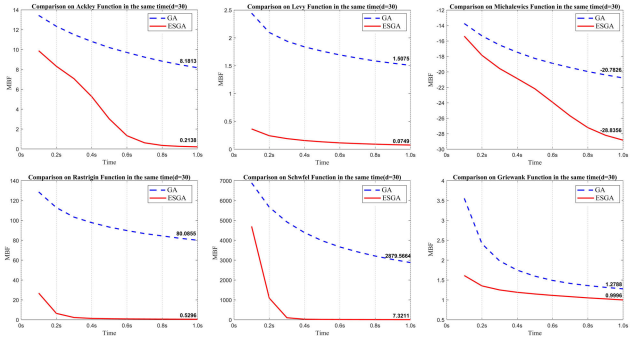

FIGURE 11. Comparison of MBFs in the same time between GA and ESGA on benchmark problems when the dimension is 30.

When the termination condition is the maximum iterations, the smallest gap between two algorithms is 1.4671, while the largest difference between the two algorithms is 5991.5344. When the termination condition is the maximum computational time, the smallest gap between two algorithms reaches 0.2792, while the largest difference between the two algorithms reaches 2872.2453. ESGA ensures the users find better solutions with less time.

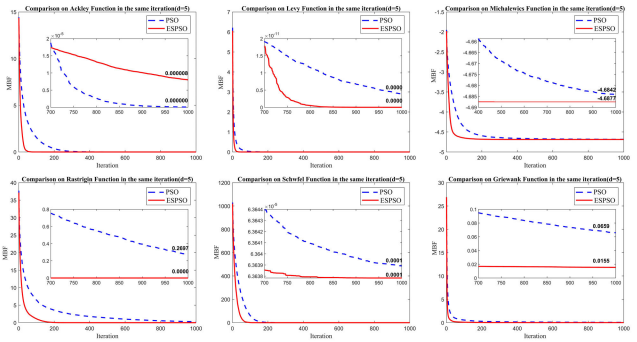

FIGURE 12. Comparison of MBFs in the same generation between PSO and ESPSO on benchmark problems when the dimension is 5.

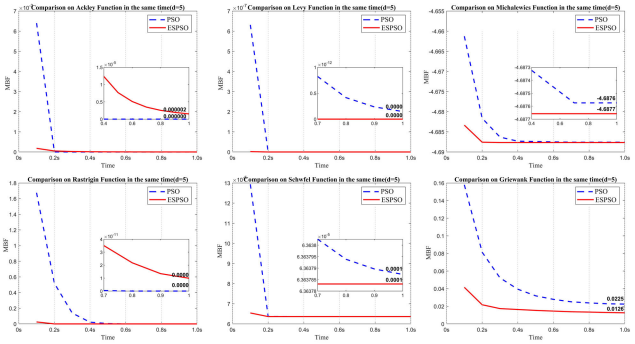

FIGURE 13. Comparison of MBFs in the same generation between PSO and ESPSO on benchmark problems when the dimension is 5.

2) EXPERIMENT ON PSO AND ESPSO

When the dimension is 5, the comparisons of MBFs between PSO and ESPSO are shown in Fig. 12 and Fig. 13. From the figures, we can see that the MBFs of both PSO and ESPSO are close to the optimal solutions. When the stopping condition is the maximum iterations, the MBF of ESPSO is better than that of PSO on five of these functions, while the performance of PSO is better on the Ackley function. Both algorithms converge quickly, but ESPSO is faster. The smallest gap between them is ten to the negative eleven.

When the stopping condition is the maximum time, ESPSO approaches the optimal solution faster than PSO. The difference between their MBFs is less than ten to the negative four on four of these functions. For the Ackley and Rastrigin functions, the performance of PSO surpasses the performance of ESPSO after 0.4 seconds, but the difference is very small. ESPSO still performs better on the other functions.

Tracing the reason why PSO performs better than ESPSO on the Ackley and Rastrigin functions, we think that the functions are simpler than the other functions and that the search ability of PSO is already strong. For simple problems, the requirements of diversity in the algorithms are not strict. ESPSO has a strong ability to jump out of the local optimal solution, so the algorithm will still ensure the diversity of the population during the convergence process. This will inevitably have a certain impact on ESPSO for simple problems.

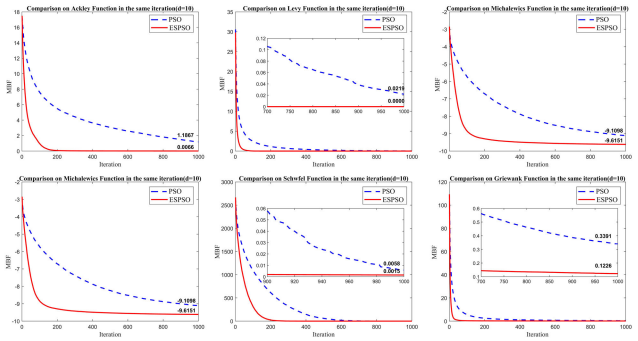

FIGURE 14. Comparison of MBFs in the same generation between PSO and ESPSO on benchmark problems when the dimension is 10.

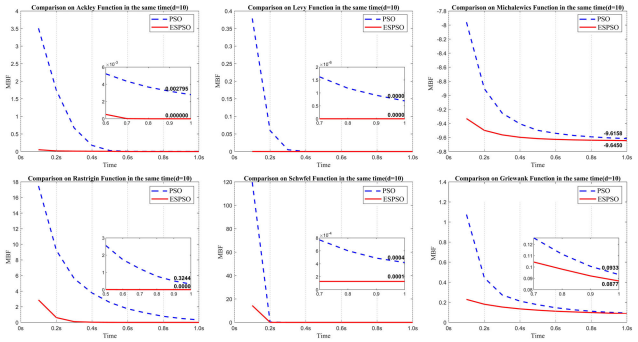

FIGURE 15. Comparison of MBFs in the same generation between PSO and ESPSO on benchmark problems when the dimension is 10.

When the dimension is 10, the comparisons of MBFs between PSO and ESPSO are shown in Fig. 14 and Fig. 15. All the results are consistent on these functions. The MBFs are improved by 1.1801, 0.0216, 0.5053, 0.5053, 0.0043 and 0.2165 through ESGA in the same iterations, while the MBFs are improved by 0.0028, 0.0000016, 0.0292, 0.3244, 0.0003 and 0.0056 in the same computational time.

The MBFs found by ESPSO tend to be steady on almost all functions in fewer than 200 generations and within 0.4 seconds, whereas PSO still does not find the satisfactory solutions to make the algorithm converge when the stopping condition is met.

When the dimension is 30, the comparisons of MBFs between PSO and ESPSO are shown in Fig. 16 and Fig. 17. The gap between ESPSO and PSO increases gradually as the number of dimensions becomes bigger.

The differences between the MBFs obtained by ESPSO and PSO in the same iterations are further expanded to 11.0212, 26.6251, 10.7784, 127.0589, 2421.7254 and 43.2527. The gap between ESPSO and PSO is enlarged as the time increases. Until the curve of PSO becomes stable, the difference between them is reduced but still large.

B. EXPERIMENTS ON DIFFERENT SCHEDULING PROBLEMS

The test task scheduling problem (TTSP), flexible job-shop scheduling problem (FJSP) and parallel machine scheduling problem (PMSP) are three different types of

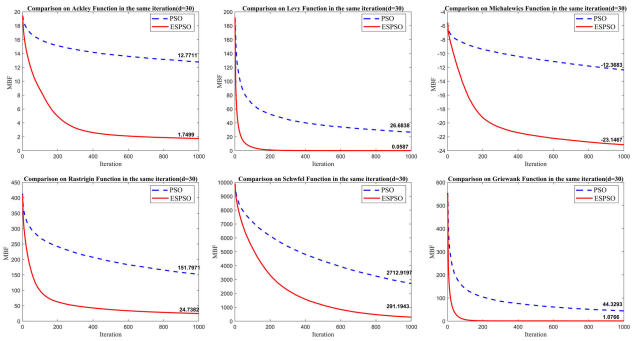

FIGURE 16. Comparison of MBFs in the same generation between PSO and ESPSO on benchmark problems when the dimension is 30.

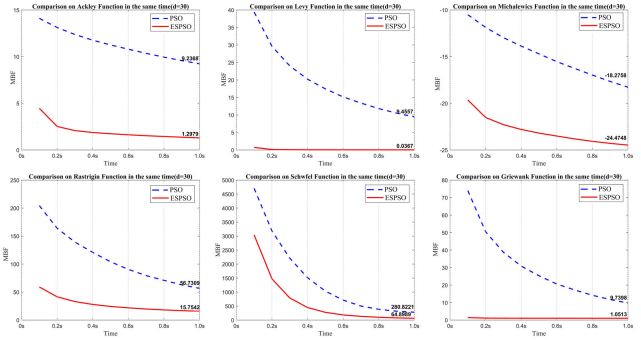

FIGURE 17. Comparison of MBFs in the same generation between PSO and ESPSO on benchmark problems when the dimension is 30.

scheduling problems. Both the single and multiobjective problems of these three problems are tested separately below. The mathematical model of TTSP is employed from [33], [37], [38], including the instances 20×8, 30×12, 40×12 and 50×15. The instances 15×3, 15×4, 20×3 and 20×4 of FJSP are from [38], [39]. Furthermore, the mathematical model of PMSP and the test instances are derived by [38], [40] including the instances 20×10, 30×10, 40×15 and 50×15. The instance $n\times m$ represents that this problem contains n jobs and m machines.

In the experiments on scheduing problems, ESGA is compared with not only the standard GA but also linear increasing crossover rate GA (LGA) and GA of entropy-based range parameter control (EARPCGA). In the same time, ESPSO is compared with the standard PSO, linear decreasing inertia weight PSO (LPSO) and PSO of entropy-based range parameter control (EARPCPSO).

LGA and LPSO are based on the same parameter control method [41]. They are the most widely applied GA and PSO variants, which tend to explore the search space at early stage of the evolution and exploit at later stage of the evolution. EARPCGA and EARPCPSO are based on another parameter control method [42]. Their main contribution is that parameters with good performance are clustered based on the entropy to determine the distribution of the parameter in next generation.

TABLE 2. Comparison of the performance among different algorithms on TTSP when the stopping condition is iterations.

	Instance 20 × 8			Instance 30 × 12			Instance 40 × 12			Instance 50 × 15		
	BV	SR	MBF	BV	SR	MBF	BV	SR	MBF	BV	SR	MBF
GA	30	0.02	32.05	36	0.06	37.95	45	0.01	48.09	63	0.01	68.09
LGA	30	0.03	31.91	36	0.07	37.90	45	0.01	48.02	63	0.03	67.05
EARPCGA	30	0.10	31.69	35	0.14	37.77	45	0.03	47.97	62	0.03	67.43
ESGA	29	0.11	31.51	33	0.18	37.31	43	0.08	47.28	62	0.08	66.74
PSO	30	0.06	31.83	36	0.02	38.90	46	0.01	48.91	62	0.02	67.48
LPSO	30	0.07	31.80	35	0.03	38.79	45	0.04	48.46	62	0.01	66.96
EARPCPSO	30	0.07	31.68	35	0.13	37.67	45	0.08	47.68	61	0.02	66.18
ESPSO	29	0.10	31.68	34	0.17	37.55	43	0.22	47.61	59	0.07	65.73

TABLE 3. Comparison of the performance among different algorithms on FJSP when the stopping condition is iterations.

	Instance 15 × 3			Instance 15 × 4			Instance 20 × 3			Instance 20 × 4		
	BV	SR	MBF	BV	SR	MBF	BV	SR	MBF	BV	SR	MBF
GA	79	0.01	88.10	69	0.02	77.79	122	0.01	133.67	147	0.01	160.75
LGA	78	0.02	87.85	68	0.02	77.30	121	0.02	132.52	146	0.02	160.43
EARPCGA	74	0.04	87.46	64	0.03	76.99	118	0.04	132.44	141	0.02	160.04
ESGA	74	0.10	84.66	61	0.20	73.63	114	0.09	128.65	138	0.13	155.73
PSO	79	0.01	87.64	73	0.01	83.64	125	0.01	138.27	139	0.05	158.37
LPSO	78	0.03	87.51	69	0.05	81.64	123	0.03	136.78	139	0.07	154.82
EARPCPSO	77	0.03	87.07	63	0.15	76.49	115	0.10	131.43	137	0.11	153.87
ESPSO	64	0.05	86.11	62	0.19	76.32	114	0.13	130.47	136	0.13	153.12

The crossover rate of LGA starts from 0.6 and decreases linearly to 1. The inertia weight of LPSO is set to 1.4 and linearly decreases to 0 when the algorithm stops running. Both of the crossover rate Pc and mutation rate Pc of EARPCGA are adjusted, their thresholds are $[0.5, 1]$ and $[0, 0.5]$. The inertia weight w and acceleration factor c of EARPCPSO are the control parameters, their thresholds $[0.5, 1.5]$ and $[1.5, 2.5]$.

1) SINGLE-OBJECTIVE SCHEDULING PROBLEMS

Minimizing the makespan of the scheduling problems is the optimization target of a single-objective problem. The evaluation metrics for the performance of algorithms on single-objective scheduling problems includes the best value (BV), the success rate to find the satisfactory solutions (SR) and the mean of all the best fitness values found in 500 times (MBF).

The comparison of the performance among different algorithms on TTSP when the stopping condition is iterations is shown in Table 2. In instance 20 × 8, the MBFs of these algorithms are very close, but the BV of ESGA and ESPSO is better than that of the other algorithms. The SRs of GA and LGA are poor, only 0.2 and 0.3, respectively. In instance 30 × 12, the BVs of ESGA and ESPSO are at least 2-unit time faster than those of other algorithms. The SR of ESGA is higher than that of GA, LGA and EARPCGA by 12%, 11% and 4%, respectively, while the SR of ESPSO is higher than that of PSO, LPSO and EARPCPSO by 15%, 14% and 4%, respectively. The performance of EARPCGA and EARPCPSO is not bad. In instance 40×12, the advantages of ESGA and ESPSO are significant in all indicators. the SR of ESPSO is especially outstanding. In instance 50×15, the BV of ESPSO is the only one less than 60. The SR of ESGA and ESPSO is at least 5% higher than the SR of other algorithms. As the examples become more complex, the superiority of ESGA and ESPSO becomes more significant.

The comparison of the performance among different algorithms on FJSP when the stopping condition is iterations is shown in Table 3. In instance 15 × 3, all indicators of ESGA and ESPSO are already ahead of the other algorithms. The BV of ESPSO is 64, better than the BVs of other algorithms by at least 10. In instance 15×4, the MBF of ESGA is higher than that of GA, LGA and EARPCGA by 4.16, 3.67 and 3.36, respectively, while the MBF of ESPSO is higher than that of PSO, LPSO and EARPCPSO by 7.32, 5.32 and 0.17, respectively. Although the performance of EARPCPSO is close to that of ESPSO, ESGA is still not as good as EARPCGA. In instance 20×3 and instance 20×4, ESGA is always the best algorithm, followed by ESPSO. EARPCPSO is close to ESPSO but not as good.

The comparison of the performance among different algorithms on PMSP when the stopping condition is iterations is shown in Table 4. The performance of ESGA and ESPSO is remarkable in all the instances. They have many advantages over other algorithms in all indicators. In instance 20 × 10, the SR of ESGA is higher than that of GA, LGA and EARPCGA by 53%, 50% and 43%, respectively, while the SR of ESPSO is higher than that of PSO, LPSO and EARPCPSO by 64%, 64% and 58%, respectively. In instance 30 × 10, the BV of ESGA is 21, and the BV of ESPSO is 22. However, the BVs of other algorithms are only between 26 to 28. The MBFs of the algorithms based on entropy and rules of nature are at least 2-unit time better than the other algorithms. In instance 40 × 15, although the BV of EARPCPSO is close to the BV of ESPSO, the performance of these algorithms is still far worse than the performance of ESGA and ESPSO in the other indicators. In instance 50×15,

TABLE 4. Comparison of the performance among different algorithms on PMSP when the stopping condition is iterations.

	Instance 20 × 10			Instance 30 × 10			Instance 40 × 15			Instance 50 × 15		
	BV	SR	MBF	BV	SR	MBF	BV	SR	MBF	BV	SR	MBF
GA	17	0.08	19.21	27	0.02	29.96	41	0.02	45.82	54	0.03	58.99
LGA	14	0.11	19.09	27	0.07	29. 79	40	0.03	45.53	52	0.06	58.82
EARPCGA	13	0.18	18.63	26	0.06	29. 65	38	0.09	44.74	50	0.08	58.23
ESGA	12	0.61	16.86	21	0.54	27.18	34	0.47	41.54	45	0.47	54.51
PSO	19	0.05	21.69	28	0.02	32.50	44	0.01	53.65	60	0.01	66.55
LPSO	17	0.05	21.28	27	0.02	31.45	44	0.03	50.55	55	0.16	63.66
EARPCPSO	15	0.11	20.71	26	0.17	31.12	38	0.18	47.63	53	0.23	58.40
ESPSO	13	0.69	18.70	22	0.36	29.04	37	0.43	44.52	47	0.53	57.53

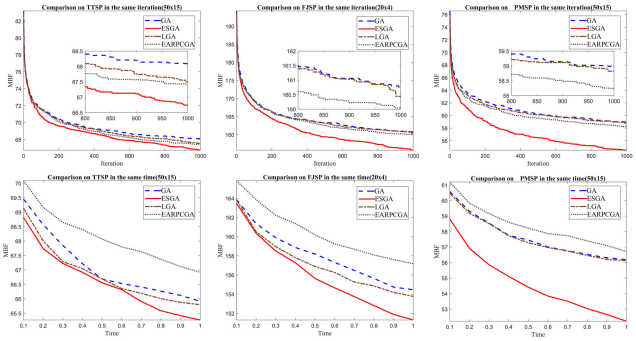

FIGURE 18. The MBFs of different genetic algorithms in the same generation and the same time.

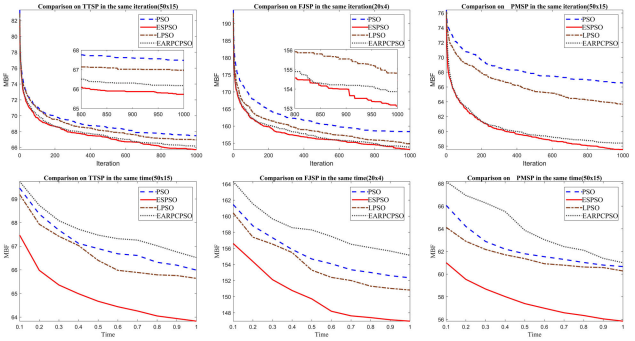

FIGURE 19. The MBFs of different particle swarm optimizations in the same generation and the same time.

only the values of BV for ESGA and ESPSO are less than 50 and their SR is approximately 50%.

The MBFs of different large-scale problems among different algorithms in the same generation and the same time are fitted into convergence curves and the figures are shown in Fig. 18 and Fig. 19. The experiments are carried out in the largest scale instances of the different scheduling problems. In Fig. 18, the blue line, red line, brown line and gray line represent the convergence curve of GA, ESGA, LGA and EARPCGA respectively. In Fig. 19, they represent the convergence curve of PSO, ESPSO, LPSO and EARPCPSO.

The conclusions of the two figures are consistent. When the termination condition is the number of iterations, the performance of ESGA and ESPSO is the best, which is followed by that of EARPCGA and EARPCPSO. The performance of GA and PSO is the worst. However, When the termination condition is the maximum computational time, EARPCGA and EARPCPSO are considered to be the worst algorithms. The performance of LGA and GA is similar, and so is the performance of LPSO and PSO. ESGA and ESPSO are still the best algorithms in any case.

When the termination condition changes, the performance of EARPCGA and EARPCPSO is greatly affected. This is because the principle of EARPCGA and EARPCPSO is complex. Clustering and other operations in the method are time consuming. Therefore, EARPCGA and EARPCPSO have no advantages over other algorithms at the same time when the termination condition is the maximum computational time. In contrast, the calculations in ESGA and ESPSO are very simple and include more targeted adjustments to parameters.

2) MULTI-OBJECTIVE SCHEDULING PROBLEMS

In addition to minimizing the makespan, the load on the machines is taken into account at this time. The load on the machines should be kept as small as possible while looking at minimizing the makespan. Therefore, the objectives of the problems are minimization of both the makespan and the average load. The evaluation metric for the performance of algorithms is the hypervolume (HV) indicator [43], [44]. For solving the multiobjective problems, NSGA-II and MOPSO are used as the multiobjective forms of GA and PSO. NSGA-II hybrid with the proposed methods based on entropy and SML (ESNSGA-II) is compared with the original NSGA-II, linear increasing crossover rate NSGA-II (LNSGA-II) and NSGA-II of entropy-based range parameter control (EARPCNSGA-II) in different scheduling problems. MOPSO hybrid with the proposed methods based on entropy and SML (ESMOPSO) is compared with the original MOPSO, linear increasing crossover rate MOPSO (LMOPSO) and MOPSO of entropy-based range parameter control (EARPCMOPSO).

The results of the HV metric are presented in Tables 5 and 6. The comparison of different genetic algorithms in the same generation in different scheduling problems is shown in Table 5. When the termination condition is iterations, ESNSGA-II performs the best, followed by EARPCNSGA-II. The original NSGA-II is the worst algorithm among them. All experimental results are consistent.

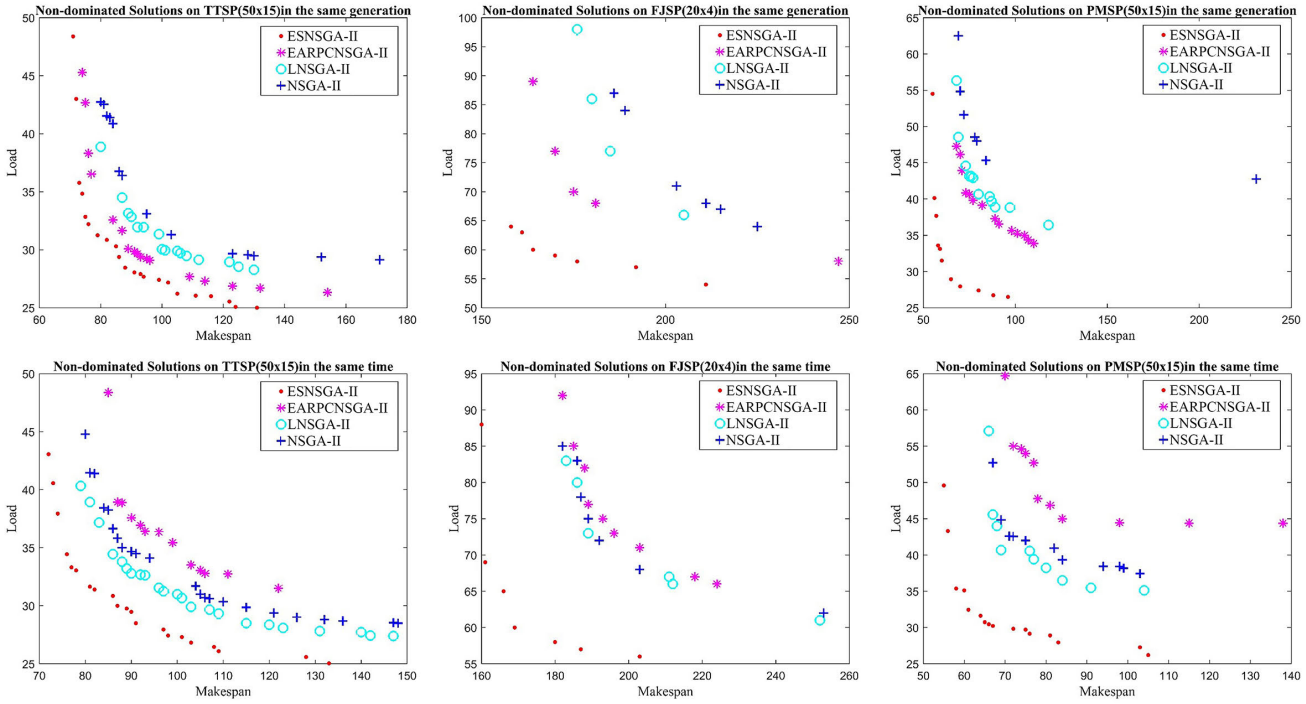

FIGURE 20. Nondominated solutions of different NSGA-II algorithms in the same generation and the same time.

TABLE 5. HV of different NSGA-II algorithms in the same generation in different scheduling problems.

	Instance	NSGA-II	LNSGA-II	EARPCNSGA-II	ESNSGA-II
T	20×8	1990.2	2381.4	2790.5	**3068.4**
T	30×12	2634.9	3212.7	4067.8	**4334.7**
S	40×12	3815.2	4091.3	4575.7	**5002.5**
P	50×15	3635.6	4163.2	4974.3	**5764.0**
F	15×3	4027.3	4109.7	4639.2	**5469.6**
J	15×4	4389.1	4527.8	4865.5	**5651.5**
S	20×3	7157.5	7654.6	8417.1	**9663.4**
P	20×4	6824.1	7595.7	8170.6	**9317.0**
P	20×10	7303.6	7796.4	7903.1	**8811.7**
M	30×10	5454.4	5915.7	6327.6	**7321.1**
S	40×15	28847.5	32179.0	34084.6	**42060.7**
P	50×15	22382.6	34953.7	39029.7	**44185.8**

TABLE 6. HV of different MOPSO algorithms in the same generation in different scheduling problems.

	Instance	NSGA-II	LNSGA-II	EARPCNSGA-II	ESNSGA-II
T	20×8	2058.7	2170.9	2581.4	**2848.3**
T	30×12	2539.6	2875.4	3424.3	**4060.2**
S	40×12	3635.6	4127.6	4792.9	**5011.9**
P	50×15	4015.2	4467.9	5135.0	**5664.8**
F	15×3	3810.2	4165.5	4819.6	**5005.7**
J	15×4	4693.3	4810.6	5440.1	**5762.0**
S	20×3	7659.7	8096.1	8807.6	**8944.9**
P	20×4	7388.2	8100.4	8823.9	**9258.8**
P	20×10	7023.3	7657.5	8348.6	**8678.7**
M	30×10	5899.9	6105.9	6743.5	**6943.3**
S	40×15	30953.4	33234.8	36446.8	**38087.1**
P	50×15	27612.9	31353.2	35306.3	**40737.8**

As the scale of the problem increases, the gap between the algorithms gradually widens. On TTSP, the HV of ESNSGA-II is better than that of other algorithms by 1078.2, 678 and 277.9 in instance 20×8, while their gap has widened to 2128.4, 1600.8 and 789.7 in instance 50×15. On FJSP, the gap has changed from 1442.3, 1359.9 and 830.4 to 2492.9, 1721.3 and 1146.4. On PMSP, the gap has changed from 1508.1, 1015.3 and 908.6 to 21803.8, 9232.1 and 5156.1. Although EARPCNSGA-II is not bad, compared with ESNSGA-II, the gap is still very obvious. It means that the set of solutions found by ESNSGA-II is the best from both convergence and diversity viewpoints.

The comparison of different particle swarm optimizations in the same generation in different scheduling problems is shown in Table 6. We can obtain the same result as in Table 5. The solutions obtained by ESMOPSO is of higher quality than by the other algorithms. On TTSP, the value of HV in ESMOPSO is 789.6, 677.4 and 266.9 more than that in MOPSO, LMOPSO and EARPCMOPSO, respectively, in instance. When the scale increases to, the differences between them become 1649.6, 1196.9 and 529.8 respectively. The difference between ESMOPSO and EARPCMOPSO is not very big on FJSP and MPSP, but the HV of ESMOPSO is the best in all instances.

The nondominated solutions of different algorithms in the same generation and the same time are shown in Fig. 20 and Fig. 21. The experiments are also carried out in the largest scale instances of the different scheduling problems. The abscissa of these figures is the shortest completion time, and the ordinate is the load. Red dots, magenta snowflakes, cyan

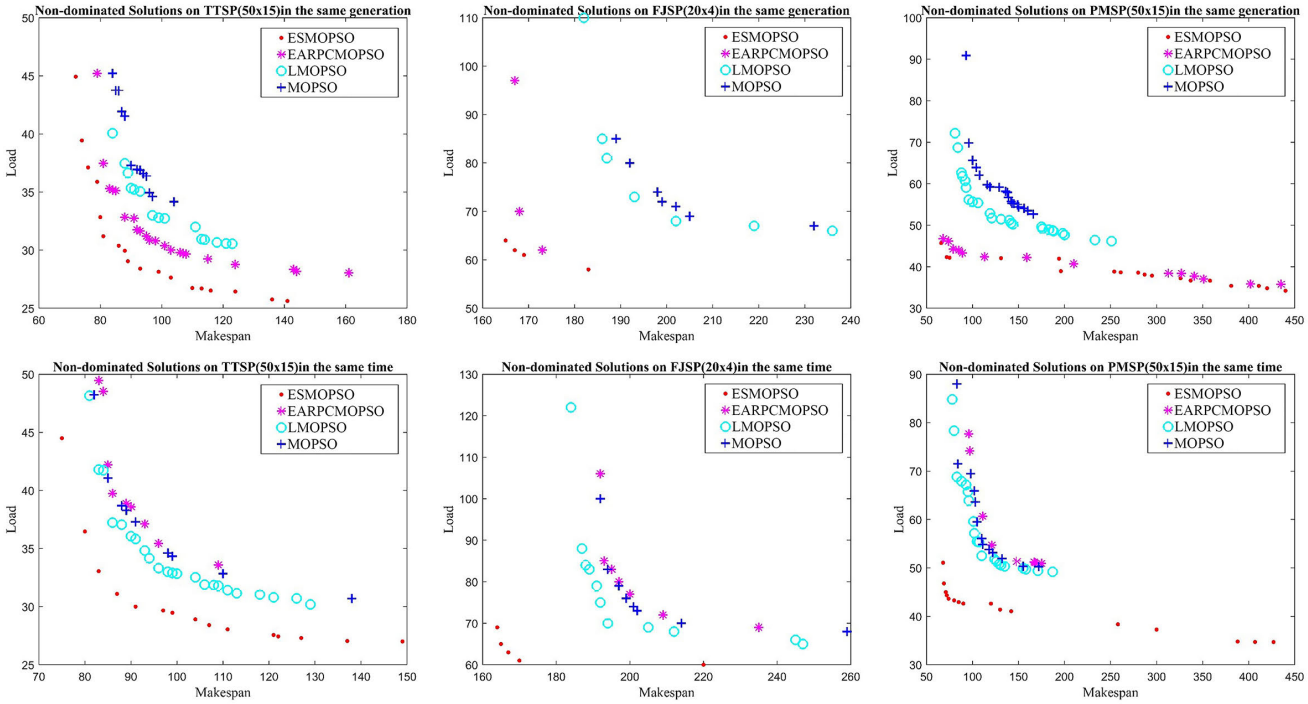

FIGURE 21. Nondominated solutions of different MOPSO algorithms in the same generation and the same time.

circles, and blue plus are used to indicate the Pareto frontier points found by ESNSGA-II, EARPCNSGA-II, LNSGA-II and NSGA-II in Fig. 20. They are used to indicate the Pareto frontier points found by ESMOPSO, EARPCMOPSO, LMOPSO and MOPSO in Fig. 21.

The nondominated solutions found by ESNSGA-II and ESMOPSO are far better than the solutions by the other algorithms. EARPCMOPSO is close to ESMOPSO only on FJSP and PMSP when the stopping condition is iterations. However, when the stopping condition is computational time, the performance of EARPCNSGA-II and EARPCMOPSO becomes very poor. They are not even as good as the original algorithms in most of the instances. We believe that the reason for this result is that the process of their method is too time consuming and not as effective as our method.

C. SUMMARY

Although different algorithms have different performances when facing different types of problems, the algorithms hybridized with our parameter control strategy perform better than the standard algorithms and other two parameter adjustment algorithms. The advantages of our method are not evidently reflected in low-dimensional problems, but the superiority becomes obvious as dimension goes higher. More importantly, our method performs well under different termination conditions of algorithms, bringing great convenience to different kinds of algorithm users. The reason is that our method analyzes the status of not only the whole population but also each gene locus, which makes parameter adjustment more targeted and on time. In addition, our method has a very simple calculation process, so it does not consume too much time.

For almost all benchmark problems, the results indicate that different kinds of meta-heuristic algorithms hybrid with the proposed method based on entropy and SML are better than other algorithms. Our proposed method is suitable for different dimension and different type of benchmark functions. In addition, when the dimensions of the test functions are small, the gap between them is not obvious. However, as the number of dimensions increases, the performance of our method is better than others. When facing complex functions, our self-adaptive method of parameter control makes it easy for the algorithms to jump out of the local optimal solution and approximate the global optimal solution. In the meantime, the convergence speed of our algorithm is very fast, so that a good solution is guaranteed to be found in a short time.

For scheduling problems, the results illustrate the differences in the quality of solutions for different algorithms. The proposed approach is more competitive than other methods in all the metrics. Our method can find satisfactory solutions in a short time and keep exploring at any stage of algorithms, so it is very valuable in practical applications. In multiobjective problems, the nondominated solutions found by our method are more forward and more balanced. Our method can solve the realistic scheduling problem well.

VI. CONCLUSION

In this paper, we proposed a self-adaptive parameter control method based on entropy and security market line for

scheduling problems. It adjusts the parameters according to the theory of security market line in finance and uses the entropy of each gene locus as feedback when adjusting the parameters. It brings the following advantages. First, since the value of the information entropy is independent of the content of the information, it provides a possibility to objectively evaluate the status of the population and adjust the parameters self-adaptively when the randomness of the algorithms is strong. Second, the status information for each locus can be tracked by the entropy so that the imbalance in the development of each gene position can be effectively solved. The status in algorithms can be measured accurately in different type of scheduling problems with different dimensions. Third, the theory of security market line is applied to address the issue that the nature of multipeak in scheduling problems makes algorithms fall into local optimal solutions. It maintains the solutions of good quality and increase the possibility that the solutions of poor quality change. An analogy is drawn between the system risk of the portfolio in finance and the diversity of the population in the algorithm, and the relationship between the parameters and the population state is established. Fourth, our method is universal, so that it can be hybrid with various meta-heuristic algorithms. This brings convenience to the users of the algorithms. Furthermore, our method will not consume too much time, so the satisfactory solutions will be found in a short time, whether the termination condition of the algorithms is the number of iterations or computational time. The experiments illustrate that our method is applicable to different meta-heuristic algorithms. In the meantime, it can solve different scheduling problems.

Further work will focus on validating our method on more algorithms to test the versatility. According to the current experiments, it also improves the performance of other algorithms, but the extent of the improvement needs to be verified. In addition, more real-world problems should be taken into consideration. Furthermore, exploring other kinds of parameter control methods is a trend deserving further research. Finally, visualizing the whole process of parameter control is also essential for users to keep track of the status of algorithms.

REFERENCES

[1] X. Ji, H. Ye, J. Zhou, Y. Yin, and X. Shen, "An improved teaching-learning-based optimization algorithm and its application to a combinatorial optimization problem in foundry industry," *Appl. Soft Comput.*, vol. 57, pp. 504–516, Aug. 2017.

[2] H. Wang, A. Mansouri, and J.-C. Créput, "Cellular matrix model for parallel combinatorial optimization algorithms in Euclidean plane," *Appl. Soft Comput.*, vol. 61, pp. 642–660, Dec. 2017.

[3] M. A. Lopes Silva, S. R. de Souza, M. J. Freitas Souza, and M. F. de F. Filho, "Hybrid metaheuristics and multi-agent systems for solving optimization problems: A review of frameworks and a comparative analysis," *Appl. Soft Comput.*, vol. 71, pp. 433–459, Oct. 2018.

[4] R. Babapour, R. Naghdi, I. Ghajar, and Z. Mortazavi, "Forest road profile optimization using meta-heuristic techniques," *Appl. Soft Comput.*, vol. 64, pp. 126–137, Mar. 2018.

[5] A. F. Nematollahi, A. Rahiminejad, and B. Vahidi, "A novel physical based meta-heuristic optimization method known as lightning attachment procedure optimization," *Appl. Soft Comput.*, vol. 59, pp. 596–621, Oct. 2017.

[6] K. K. H. Ng, C. K. M. Lee, F. T. S. Chan, and Y. Lv, "Review on meta-heuristics approaches for airside operation research," *Appl. Soft Comput.*, vol. 66, pp. 104–133, May 2018.

[7] D. H. Wolpert and W. G. Macready, "No free lunch theorems for optimization," *IEEE Trans. Evol. Comput.*, vol. 1, no. 1, pp. 67–82, Apr. 1997.

[8] A. E. Eiben, R. Hinterding, and Z. Michalewicz, "Parameter control in evolutionary algorithms," *IEEE Trans. Evol. Comput.*, vol. 3, no. 2, pp. 124–141, Jul. 1999.

[9] A. Aleti and I. Moser, "A systematic literature review of adaptive parameter control methods for evolutionary algorithms," *ACM Comput. Surv.*, vol. 49, no. 3, pp. 56:1–56:35, 2016.

[10] A. L. C. Runehov, L. Oviedo, and N. P. Azari, Eds., *Encyclopedia of Sciences and Religions*. Amsterdam, The Netherlands: Springer, 2013.

[11] P. H. Dybvig and S. A. Ross, "Differential information and performance measurement using a security market line," *J. Finance*, vol. 40, no. 2, pp. 383–399, Jun. 1985.

[12] O. Kramer, *Self-Adaptive Heuristics for Evolutionary Computation*, vol. 147. Springer, 2008.

[13] S. Aine, R. Kumar, and P. P. Chakrabarti, "Adaptive parameter control of evolutionary algorithms to improve quality-time trade-off," *Appl. Soft Comput.*, vol. 9, no. 2, pp. 527–540, Mar. 2009.

[14] R. Tanabe and H. Ishibuchi, "An analysis of control parameters of MOEA/D under two different optimization scenarios," *Appl. Soft Comput.*, vol. 70, pp. 22–40, Sep. 2018.

[15] S. Bagheri, W. Konen, M. Emmerich, and T. Bäck, "Self-adjusting parameter control for surrogate-assisted constrained optimization under limited budgets," *Appl. Soft Comput.*, vol. 61, pp. 377–393, Dec. 2017.

[16] S. W. Leung, S. Y. Yuen, and C. K. Chow, "Parameter control system of evolutionary algorithm that is aided by the entire search history," *Appl. Soft Comput.*, vol. 12, no. 9, pp. 3063–3078, Sep. 2012.

[17] S.-H. Liu, M. Mernik, D. Hrnčič, and M. Črepinšek, "A parameter control method of evolutionary algorithms using exploration and exploitation measures with a practical application for fitting Sovova's mass transfer model," *Appl. Soft Comput.*, vol. 13, no. 9, pp. 3792–3805, Sep. 2013.

[18] J. A. Joines and C. R. Houck, "On the use of non-stationary penalty functions to solve nonlinear constrained optimization problems with GA's," in *Proc. 1st IEEE Conf. Evol. Comput., IEEE World Congr. Comput. Intell.*, Orlando, FL, USA, Jun. 1994, pp. 579–584.

[19] J. C. Costa, R. Tavares, and A. Rosa, "An experimental study on dynamic random variation of population size," in *Proc. IEEE SMC99 Conf., IEEE Int. Conf. Syst., Man, Cybern.*, Tokyo, Japan, Oct. 1999, pp. 607–612.

[20] A. Dukkipati, M. Narasimha Murty, and S. Bhatnagar, "Cauchy annealing schedule: An annealing schedule for Boltzmann selection scheme in evolutionary algorithms," in *Proc. Congr. Evol. Comput.*, Portland, OR, USA, 2004, pp. 55–62.

[21] G. Karafotias, M. Hoogendoorn, and A. E. Eiben, "Parameter control in evolutionary algorithms: Trends and challenges," *IEEE Trans. Evol. Comput.*, vol. 19, no. 2, pp. 167–187, Apr. 2015.

[22] C. M. Fu, C. Jiang, G. S. Chen, and Q. M. Liu, "An adaptive differential evolution algorithm with an aging leader and challengers mechanism," *Appl. Soft Comput.*, vol. 57, pp. 60–73, Aug. 2017.

[23] A. Aleti, I. Moser, and S. Mostaghim, "Adaptive range parameter control," in *Proc. IEEE Congr. Evol. Comput.*, 2012, pp. 1–8.

[24] B. McGinley, J. Maher, C. O'Riordan, and F. Morgan, "Maintaining healthy population diversity using adaptive crossover, mutation, and selection," *IEEE Trans. Evol. Comput.*, vol. 15, no. 5, pp. 692–714, Oct. 2011.

[25] M. Srinivas and L. M. Patnaik, "Adaptive probabilities of crossover and mutation in genetic algorithms," *IEEE Trans. Syst., Man, Cybern.*, vol. 24, no. 4, pp. 656–667, Apr. 1994.

[26] H. P. Schwefel, "Mechanismen in der biologischen Evolution und ihr Einfluss auf die Evolutionsgeschwindigkeit," *Interner Bericht Arbeits-Gruppe Bionik Evolutionstechnik Inst. Mess-und Regelung–Stechnik* vol. 215, 1974.

[27] M. H. Maruo and H. S. Lopes, "Self-adapting evolutionary parameters: Encoding aspects for combinatorial optimization problems," in *Proc. Eur. Conf. Evol. Comput. Combinat. Optim.* Berlin, Germany: Springer, 2005, pp. 154–165.

[28] Q. Fan and X. Yan, "Self-adaptive differential evolution algorithm with discrete mutation control parameters," *Expert Syst. Appl.*, vol. 42, no. 3, pp. 1551–1572, Feb. 2015.

[29] L. Davis, *Handbook of Genetic Algorithms*. New York, NY, USA: Van Nostrand, 1991.

[30] J. Maturana and F. Saubion, "A compass to guide genetic algorithms, expert systems with applications," in *Proc. Int. Conf. Parallel Problem Solving Nature*. Berlin, Germany: Springer, 2008, pp. 256–265.
[31] R. K. Ursem, *Models for Evolutionary Algorithms and Their Applications in System Identification and Control Optimization*. Russia, Brazil: BRICS, 2003.
[32] J. Shi, H. Lu, and K. Mao, "Solving the test task scheduling problem with a genetic algorithm based on the scheme choice rule," in *Proc. Int. Conf. Swarm Intell.* Cham, Switzerland: Springer, 2016, pp. 19–27.
[33] H. Lu, R. Niu, J. Liu, and Z. Zhu, "A chaotic non-dominated sorting genetic algorithm for the multi-objective automatic test task scheduling problem," *Appl. Soft Comput.*, vol. 13, no. 5, pp. 2790–2802, May 2013.
[34] H.-L. Lu, X.-S. Wen, L. Lan, Y.-Z. An, and X.-P. Li, "A self-adaptive genetic algorithm to estimate JA model parameters considering minor loops," *J. Magn. Magn. Mater.*, vol. 374, pp. 502–507, Jan. 2015.
[35] H. Lu, J. Shi, Z. Fei, Q. Zhou, and K. Mao, "Analysis of the similarities and differences of job-based scheduling problems," *Eur. J. Oper. Res.*, vol. 270, no. 3, pp. 809–825, Nov. 2018.
[36] (2013). *Virtual Library of Simulation Experiments: Test Functions and Datasets, Optimization Test Problems*. Accessed: Feb. 28, 2020. [Online]. Available: https://www.sfu.ca/~ssurjano/index.html
[37] H. Lu, Z. Zhu, X. Wang, and L. Yin, "A variable neighborhood MOEA/D for multiobjective test task scheduling problem," *Math. Problems Eng.*, vol. 2014, pp. 1–14, Apr. 2014.
[38] R. Zhou, H. Lu, and J. Shi, "Framework Based on Packet Scheduling and Dispatching Rule for Job-Based Scheduling Problems," in *Proc. Int. Conf. Sens. Imag.* Cham, Switzerland: Springer, 2018, pp. 202–211.
[39] P. Brandimarte, "Routing and scheduling in a flexible job shop by tabu search," *Ann. Oper. Res.*, vol. 41, no. 3, pp. 157–183, Sep. 1993.
[40] E. Vallada and R. Ruiz, "A genetic algorithm for the unrelated parallel machine scheduling problem with sequence dependent setup times," *Eur. J. Oper. Res.*, vol. 211, no. 3, pp. 612–622, Jun. 2011.
[41] H. Shi and R. Eberhart, "A modified particle swarm optimizer," in *Proc. IEEE World Congr. Comput. Intell.*, Anchorage, AK, USA, May 1998, pp. 69–73.
[42] A. Aleti and I. Moser, "Entropy-based adaptive range parameter control for evolutionary algorithms," in *Proc. 15th Annu. Conf. Genetic Evol. Comput. Conf. (GECCO)*, Amsterdam, The Netherlands, 2013, pp. 1501–1508.
[43] E. Zitzler and L. Thiele, "Multiobjective optimization using evolutionary algorithms—A comparative case study," in *Proc. Int. Conf. Parallel Problem Solving Nature*. Berlin, Germany: Springer, 1998, pp. 292–301.
[44] E. Zitzler and L. Thiele, "Multiobjective evolutionary algorithms: A comparative case study and the strength Pareto approach," *IEEE Trans. Evol. Comput.*, vol. 3, no. 4, pp. 257–271, Nov. 1999.

SHENGJIE SUN received the B.Sc. degree from the School of Electronic and Information Engineering, Baihang University, Beijing, China, in 2018, where he is currently pursuing the master's degree. His main research areas include data mining and intelligent optimization.

HUI LU (Senior Member, IEEE) received the Ph.D. degree in navigation, guidance and control from Harbin Engineering University, Harbin, China, in 2004. She is currently a Professor with Beihang University, Beijing, China. Her research interests include information and communication systems and intelligent optimization.