A Boltzmann-Based Estimation of Distribution Algorithm for a General Resource Scheduling Model

Xinle Liang, Huaping Chen, and Jose A. Lozano, *Member, IEEE*

***Abstract*—Most researchers employed common functional models when managing scheduling problems with controllable processing times. However, in many complicated manufacturing systems with a high diversity of jobs, these functional resource models fail to reflect their specific characteristics. To fulfill these requirements, we apply a more general model, the discrete model. Traditional functional models can be viewed as special cases of such model. In this paper, the discrete model is implemented on a problem of minimizing the weighted resource allocation subject to a common deadline on a single machine. By reducing the problem to a partition problem, we demonstrate that it is NP-complete, which addresses the difficult issue of the guarantee of both the solution quality and time cost. In order to tackle the problem, we develop an estimation of distribution algorithm based on an approximation of the Boltzmann distribution. The approximation strategy represents a tradeoff between complexity and solution accuracy. The results of the experiments conducted on benchmarks show that, compared with other alternative approaches, the proposed algorithm has competitive behavior, obtaining 74 best solutions out of 90 instances.**

***Index Terms*—Boltzmann distribution, controllable processing times, discrete model, estimation of distribution algorithm (EDA), scheduling.**

I. Introduction

RESOURCE efficiency is a determinant factor in manufacturing industries. Rapid changes in production systems with resource allocations have attracted the principal attention of contributors. The problem of improving resource efficiency when fulfilling a given production plan has become one of the major concerns in this field [1]–[4].

Manuscript received April 2, 2014; revised September 5, 2014; accepted November 14, 2014. Date of publication December 18, 2014; date of current version November 25, 2015. This work was supported in part by the 973 Program of China under Grant 2011CB707006, in part by the National Natural Science Foundation of China under Grant 61329302 and Grant 71171184, and in part by the European Union Seventh Framework Programme under Grant 247619. The work of J. A. Lozano was supported in part by the Basque Government Research Groups 2013–2018 under Grant IT-609-13 and in part by the Ministerio de Economia y Competitividad under Grant TIN2013-41272P.

X. Liang and H. Chen are with the School of Computer Science and Technology, University of Science and Technology of China, Hefei 230027, China (e-mail: shindler@mail.ustc.edu.cn).

J. A. Lozano is with the Intelligent Systems Group, Department of Computer Science and Artificial Intelligence, University of the Basque Country, Leioa 20018, Spain.

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TEVC.2014.2382135

When dealing with resource allocation problems, researchers often associate processing times with resource allocation. In many production systems, the processing times are controllable by resource allocations, such as money, overtime, energy, fuel, and catalysts and additional manpower. Interesting applications related to controllable processing times can be found in [5]–[7].

In this paper, a scheduling problem with resource-dependent processing times is investigated. Studies in the subfield of scheduling with controllable processing times originate in [8] and [9]. Surveys of related results were provided by Nowicki and Zdrzałka [10] and Shabtay and Steiner [11]. Research dealing with related problems can be found in [12]–[17]. In these studies, different approaches for achieving a trade-off between the production objectives and the total processing cost are made to reach the most efficient performance.

Scheduling problems with controllable processing times are essentially summarized as a problem with two criteria: 1) scheduling and 2) resource allocation criteria [11], which derives four different problem variants. The first two variants consist in optimizing one criterion subject to a constraint on another. The third one considers an integrated objective [17], [18], and the fourth one is to identify the Pareto-optimal schedules given a specialized production environment [19].

The problem studied in this paper belongs to the category of the first two variants stated above, i.e., to minimize the resource allocation criterion subject to the given scheduling constraints. The scheduling criterion employed is to minimize the total weighted resource allocation subject to a common deadline for all jobs. A solution is considered feasible if all jobs are completed ahead of the deadline. Deadline-related criteria are among the most commonly investigated in scheduling problems [12], [13], [15], [20], since more companies are concerned about the satisfaction of the customer's demand with respect to deadline. Tardy jobs may result in customer dissatisfaction, contract penalties, loss of sale and reputation among other issues.

To depict the one-to-one mappings among resource allocations and processing times, practically all the previous studies employed functional models, such as linear models [13], [15], [20], or convex decreasing models [21]–[23], among others. In each functional model, a uniform formulation is constructed for all jobs. However,

Yedidsion *et al.* [24], [25] noted that many applications in real-life systems are in contradiction with these models. For example, in a complicated manufacturing environment, jobs may be of different characteristics and no uniform functional formulations exist. They then showed that the researches employing more general models can be implemented on much larger scale of problems. Following their ideas, we apply one of such models, the discrete model, which is proposed in [12].

Instead of using mathematical functionals, the discrete model maintains a processing time set for each job. Each processing time in the set corresponds to a discrete resource allocation. The discrete model presents a new perspective for scheduling problems with discrete resource allocations. Traditional functional models for discrete resource allocation can be viewed as special cases of the discrete model.

The problem formulation in this paper is similar to the settings in [15]. The only difference is that in their paper, the authors employed the linear resource model. They proposed several approaches based on the linearity of such model, which makes them unfit for the problem in this paper.

In this context, we present a transformation procedure for the complexity analysis and then develop a novel approach for the considered problem. Through the complexity analysis, it is found that the problem can be reduced to a partition problem [26], which is a known NP-complete problem. We show that the constraints and the objective in the considered problem can be represented by the elements in the corresponding partition problem. It is suggested that the transformation and representation procedures between the instances (of the partition problem and the considered problem) can be completed in polynomial time. Following the theory of NP-completeness [26], the problem in this paper is NP-complete.

The NP-completeness of the considered problem reflects the difficult issue of obtaining the optimal solutions. For the instances with sufficiently-large search spaces, computation exhaustive algorithms or local search procedures often fail to provide the optimal (or even high-quality) solutions in reasonable time. Therefore, it is of practical meaning to develop other approaches to maintain a tradeoff between complexity and solution accuracy.

To tackle the investigated problem, we present an estimation of distribution algorithm (EDA). EDAs share the advantages of evolutionary algorithms (EAs) i.e., they are widely applicable and easily implementable. Besides, EDAs were claimed effective when dealing with applications with multiconstraints and multicriteria decision making processes [27]–[32]. These characteristics encourage us to make an attempt to tackle the problem with EDAs. To associate the distribution with the fitness values, we build an approximation of the Boltzmann distribution. The Boltzmann distribution is capable of reflecting the differences in fitness values, which can be used for generating more promising solutions in the next generation. Proper learning procedures for the distribution can lead the solutions to optimal regions. To further improve the approach, two approximation methods for learning the distribution parameters are provided.

In order to demonstrate the effectiveness of the proposed algorithm, an experimental study is conducted. Classical versions of several available approaches are introduced as comparison algorithms. The approaches compared are specifically introduced from three categories: 1) exhaustive algorithms; 2) local-search-based approaches; and 3) EAs. We emphasize on the convergence analysis and performance comparisons on small- and large-scale problems. The results showed that the proposed EDA has competitive convergence speed and obtains the best results among all the compared approaches.

The remainder of this paper is organized as follows. Section II presents the problem formulation, which is followed by the complexity analysis. In Section III, our novel EDA is presented for solving the problem under investigation. Section IV is devoted to the analysis of the convergence of the proposed algorithm and the comparisons of the proposed EDA with the alternative algorithms. The conclusion with future research directions is given in Section V.

II. Scheduling With Discrete Model

This section presents a detailed description and complexity analysis of the problem under consideration. In the first part, a mathematical formulation of the considered problem is proposed. Then, in the second part, the complexity analysis is conducted by reducing the problem to a partition problem, which demonstrates its NP-completeness.

A. Problem Formulation

The mathematical formulation of the considered problem is presented as follows.

1) Notation:

n	Number of jobs.
$\mathcal{W}$	$(w_1, w_2, \ldots, w_n)$ is the weight vector.
PT_i	Actual processing time of job i.
C_i	Completion time of job i.
d	Common deadline for all jobs.
r_i	Resource allocation for job i.
$\mathcal{R}$	$(r_1, r_2, \ldots, r_n)$ is the resource allocation vector.

2) Constraints: Each schedule is subject to the following resource allocation and time constraints.

a) Resource allocation constraints: The resource allocation for any job i is discrete and has finite possibilities

$$r_i \in \{1, 2, \ldots, k\},\ i = 1, 2, \ldots, n \tag{1}$$

where k represents the number of resource allocation possibilities for all jobs.

b) Time constraints:

1) In the discrete model, each job i has k possible processing times $\{p_{i1}, p_{i2}, \ldots, p_{ik}\}$, where p_{ij} is the processing time of job i with j unit resources and $p_{i1} > p_{i2} > \cdots > p_{ik}$. The actual processing time of job i is determined by its resource allocation

$$PT_i(r_i) = p_{ir_i},\ i = 1, 2, \ldots, n. \tag{2}$$

2) The completion time of the last job i should not exceed d

$$\sum_{j=1}^{n} PT_j \le d. \tag{3}$$

For each solution, if the completion time of the last job is smaller than d, we say it is feasible. Moreover, for each job, its processing time is independent with its position in the schedule. Therefore, the job sequence is irrelevant to the solution quality. In order to simplify the representations, we use sequence $(1, 2, \ldots, n)$ in each solution. If we denote by $C_i = \sum_{j=1}^{i} PT_{[j]}$ the completion time of the first ith jobs ($PT_{[j]}$ denotes the processing time of the job in the jth position), then the constraint can be written as

$$C_n \le d. \tag{4}$$

3) Objective: The objective is to minimize the sum of the total weighted resource consumption

$$\min \qquad S(\mathcal{R}) = \mathcal{W}\mathcal{R}^T. \tag{5}$$

4) Decision Variables: The solution for the problem is to determine the resource allocation $\mathcal{R} = (r_1, r_2, \ldots, r_n)$ for all jobs, where each r_i is chosen from the set $\{1, 2, \ldots, k\}$.

B. Complexity Analysis

It is shown in [11] that there is no appearing evidence whether the adoption of different resource models can change the complexity of the original scheduling problems. Therefore, before seeking for effective solving approaches, it is believed necessary to analyze the complexity. It is proven that a simplified version of the problem is NP-complete in the following.

Theorem 1: The problem is NP-complete even if $k = 2$ and $p_{i1} = p$ for $i = 1, 2, \ldots, n$.

Proof: We show that the simplified version of the problem can be transformed into a partition problem, which is a known NP-complete problem. The partition problem can be stated as follows: given positive integers $e_1, e_2, \ldots, e_n$, is there a binary vector $\mathcal{Y} = (y_1, y_2, \ldots, y_n)$, $y_i \in \{0, 1\}$ such that $\sum_{i=1}^{n} y_i e_i = 0.5T$, where $\sum_{i=1}^{n} e_i = T$?

Given an instance of the partition problem, the following instance of the considered problem is constructed: $w_i = e_i$, $p - p_{i2} = e_i$, $p = 0.5T$, and $d = 0.5(n-1)T$. We will show that there exists a binary sequence $\mathcal{Y} = (y_1, y_2, \ldots, y_n)$ for which $\sum_{i=1}^{n} y_i e_i = 0.5T$ if and only if there exists a corresponding solution $\mathcal{R} = (r_1, r_2, \ldots, r_n)$ for which $C_n \le d$ and $S(\mathcal{R}) = 1.5T$. It is obvious that the representation procedure can be completed in polynomial time.

The transformation procedures between $\mathcal{Y}$ and $\mathcal{R}$ are presented as follows: for each y_i in $\mathcal{Y}$, if $y_i = 0$, then set $r_i = 1$ in $\mathcal{R}$; otherwise if $y_i = 1$, set $r_i = 2$ in $\mathcal{R}$. Similarly, an inverse procedure can produce an unique $\mathcal{Y}$ from $\mathcal{R}$. It is obvious that such procedures form a one-to-one mapping between $\mathcal{Y}$ and $\mathcal{R}$. Both transformation procedures can be completed in polynomial time.

Let $\eta = \sum_{y_i=1} e_i - 0.5T$ and calculate the parameters, for job n

$$\begin{aligned} C_n &= \sum_{i=1}^{n} p_{ir_i} \\ &= 0.5(n-1)T - \eta \end{aligned} \tag{6}$$

and the objective value is

$$\begin{aligned} S(\mathcal{R}) &= \sum_{i=1}^{n} w_i r_i \\ &= 2\sum_{y_i=1} e_i + \sum_{y_i=0} e_i \\ &= 1.5T + \eta. \end{aligned} \tag{7}$$

Discussion: If $\eta > 0$, the solution $\mathcal{R}$ is feasible with the objective value greater than $1.5T$. If $\eta < 0$, $\mathcal{R}$ is infeasible since $C_n > d$. If $\eta = 0$, $\mathcal{R}$ is feasible with the objective value $1.5T$. It follows that if and only if there exists a solution $\mathcal{Y}$ satisfying $\sum_{i=1}^{n} y_i e_i = 0.5T$, the solution $\mathcal{R}$ can achieve the minimal objective value, which completes our proof. ■

III. EDA Based on the Boltzmann Distribution

The NP-completeness of the considered problem reflects the difficulty to obtain the optimal solutions for the problem. For large-scale problems, exhaustive algorithms spent an unaffordable amount of computing resources and local search procedures often cannot provide the optimal (or even high-quality) solutions in reasonable time. Therefore, it is of practical meaning to develop other approaches to maintain a trade-off between solution quality and time cost. In this section, we propose an approach based on EDAs to tackle the problem.

This section is devoted to introducing the EDA approach to be proposed. First, the general framework for EDA is presented. Second, the probabilistic model for the EDA approach is presented, which is followed by the corresponding learning and sampling procedures. At last, we conduct the complexity analysis of the proposed approach and then evaluate the approximation methods in the learning procedure.

A. EDA Framework

EAs adopt the evolution of species in the natural world, which is carried out by selection and random changes as working principle. Compared with classical methods, EAs are widely applicable while making few assumptions about the optimization problem being solved. Many variants of such as genetic algorithms (GAs) [32]–[35], evolution strategies [36], and evolutionary programming [37]–[40] have been applied in the solution of scheduling problems.

EDAs [27], [28], [41], [42] are EAs that construct an explicit probability model of the set of selected solutions and sample this probability model in order to obtain the new generation of offsprings. In EDAs, the classical genetic recombination operators are replaced by probability estimation and stochastic sampling techniques. The new population of individuals

Algorithm 1: General Framework for EDAs

$D_0 \leftarrow$ Generate N individuals (the initial population) at random

Repeat for $l = 1, 2, \ldots$ until a stopping criterion is met

$D_{l-1}^{Se} \leftarrow$ Select $M \leq N$ individuals from D_{l-1} according to the selection method

$p_l(\mathbf{x}) = p(\mathbf{x}|D_{l-1}^{Se}) \leftarrow$ Estimate the probability distribution of the selected individuals

$D_l \leftarrow$ Sample N individuals (the new population) from $p_l(\mathbf{x})$

End

is sampled from a probability distribution that is estimated from the database of the selected individuals from the previous generation. The interrelations among the variables of the optimization problem are captured somehow in the probabilistic model. Effective learning and sampling methods can lead to the promising areas near the optimal solution. In the literature, there are numerous versions of EDA approaches, which have been applied to many real-life applications [27]–[31], [43]–[50], etc. Algorithm 1 shows a brief framework of EDAs.

B. Probabilistic Model

Our proposal is based on the Boltzmann estimated distribution algorithm (BEDA) [51]–[53]. The main idea of BEDA is the use of the Boltzmann distribution in the selection and sampling of individuals. However, BEDA is a conceptional algorithm, because the calculation of the distribution requires a sum over exponential many terms. Our choice of BEDA as departing model comes motivated by its converge properties [53]. These good properties encourage us to make an attempt to build an EDA based on an approximation of the Boltzmann distribution.

The general form of the Boltzmann distribution can be expressed as

$$p(X = x) = \frac{\exp(-\alpha E(x))}{\tau(\alpha)} \tag{8}$$

where $E(x)$ is the energy of the configuration x, α is the inverse temperature, and $\tau(\alpha)=\sum_{x'} \exp(-\alpha E(x^{'}))$ is the normalization constant.

In this paper, we approach the Boltzmann distribution as a factorization of univariate marginals. In this sense, we assume that each component in the solution for Boltzmann-distribution-based EDA (BDEDA) is independent with each other. The factorization of the probabilistic model is

$$P(\boldsymbol{\mathcal{R}}) = \prod_{i=1}^{n} p(r_i) \tag{9}$$

where each $p(r_i)$ follows (8). The following part shows how to build an approximation of the Boltzmann distribution for each $p(r_i)$.

The following notations are employed in the following parts.

1) $r_{l,i}^0$ (a real value to be estimated) is the center point for r_i in the lth generation.
2) $d(x, r_{l,i}^0)$ is the distance between any resource possibility x and $r_{l,i}^0$.
3) $\mathcal{R}_{l,i}$ is the set of r_i of the selected individuals in the lth generation.
4) $H_{l,i}(x)$ represents the best estimation of the objective value of all the individuals with $r_i = x$ in D_{l-1}^{se}.

The smaller $d(x, r_{l,i}^0)$, the greater the probability value for the appearance of x in the $(l+1)$th generation. The best estimation is calculated by

$$H_{l-1,i}(x) = \begin{cases} \min\left\{S(\boldsymbol{\mathcal{R}})|r_i = x \& \boldsymbol{\mathcal{R}} \subset D_{l-1}^{se}\right\}, & x \in \mathcal{R}_{l-1,i} \\ \max\left\{S(\boldsymbol{\mathcal{R}})|\boldsymbol{\mathcal{R}} \subset D_{l-1}^{se}\right\}, & \text{otherwise.} \end{cases} \tag{10}$$

$H_{l-1,i}(x)$ is the minimum objective value obtained from the selected individuals in the $(l-1)$th generation with $r_i = x$ (for the case there is no $r_i = x$ existing, the maximum objective value in the database is assigned). The smaller $H_{l-1,i}(x)$, the greater the probability value for the appearance of x in the next generation.

The energy parameter employed in the probabilistic model consists of the distance parameter and the best estimation values. Accordingly, the energy model for the r_i in the lth generation is presented as

$$E_{l-1,i}(x) = d\left(x, r_{l,i}^0\right) H_{l,i}(x). \tag{11}$$

Therefore, following (8), the approximation of the Boltzmann distribution is expressed as:

$$p_l(r_i = x) = \frac{\exp(-\alpha E_{l-1,i}(x))}{\tau_{l,i}(\alpha)}. \tag{12}$$

The parameter α is restricted to $\alpha > 0$, since in the case of $\alpha = 0$ a uniform distribution is achieved and an anti mode can be obtained with $\alpha < 0$.

C. Learning and Sampling

In this paper, we use the principle of maximum likelihood estimation in the probabilistic model learning procedure as follows.

For a sample of M solutions, we denote by $\mathcal{R}_{l-1,i}$ the values of the ith component of the individuals for $i = 1, 2, \ldots, n$. Then the solutions can be divided into n set $\Omega_{l-1} = \{\boldsymbol{\mathcal{R}}_{l-1,1}, \boldsymbol{\mathcal{R}}_{l-1,2}, \ldots, \boldsymbol{\mathcal{R}}_{l-1,n}\}$, and the log likelihood of $\boldsymbol{\mathcal{R}}_l^0$ for the proposed model is given by

$$\begin{aligned} \ln L_l\left(\Omega_{l-1}|\boldsymbol{\mathcal{R}}_l^0\right) &= \sum_{i=1}^{n} \ln L_l\left(\boldsymbol{\mathcal{R}}_{l-1,i}|r_{l,i}^0\right) \\ &= \sum_{i=1}^{n} \ln \prod_{s=1}^{M} p_l\left(r_{l-1,i}^s|r_{l,i}^0\right). \end{aligned} \tag{13}$$

Since the estimation of the parameter for each component is independent with other components, the learning process can be factorized as to minimize n likelihood functions, each of

which can be expressed as

$$\begin{aligned}\ln \prod_{s=1}^{M} p_l\left(r_{l-1,i}^s | r_{l,i}^0\right) &= \ln \prod_{s=1}^{M} \frac{\exp\left(-\alpha\left(d\left(r_{l-1,i}^s, r_{l,i}^0\right) H_{l-1,i}\left(r_{l-1,i}^s\right)\right)\right.}{\tau_{l,i}(\alpha)} \\ &= -\alpha \sum_{s=1}^{M} d\left(r_{l-1,i}^s, r_{l,i}^0\right) H_{l-1,i}\left(r_{l-1,i}^s\right) \\ &\quad - M \ln \sum_{s=1}^{M} \exp\left(-\alpha d\left(r_{l-1,i}^s, r_{l,i}^0\right) H_{l-1,i}\left(r_{l-1,i}^s\right)\right)\end{aligned} \tag{14}$$

where $r_{l,i}^s$ denotes the resource allocation for job i of the sth ($s \in \{1, 2, \ldots, M\}$) solution in the lth generation.

Deriving (14) with respect to $r_{l-1,i}^s$ and making it equal to zero, we obtain the following formula:

$$\begin{aligned}&\sum_{s=1}^{M} \eta_i\left(r_{l-1,i}^s\right) \sum_{s=1}^{M} \exp\left(-\alpha d\left(r_{l-1,i}^s, r_{l,i}^0\right) H_{l-1,i}\left(r_{l-1,i}^s\right)\right) \\ &- M \sum_{s=1}^{M} \eta_i\left(r_{l-1,i}^s\right) \exp\left(-\alpha d\left(r_{l-1,i}^s, r_{l,i}^0\right) H_{l-1,i}\left(r_{l-1,i}^s\right)\right) = 0\end{aligned} \tag{15}$$

where

$$\eta_i\left(r_{l-1,i}^s\right) = \left(d\left(r_{l-1,i}^s, r_{l,i}^0\right) H_{l-1,i}\left(r_{l-1,i}^s\right)\right)'. \tag{16}$$

Unfortunately, the solution for (15) does not have a closed form expression. Therefore, we introduce two approximation methods.

1) The first one is to employ the Newton–Raphson algorithm (ANR), whose complexity is closely related to the initial solution. With a good initial solution approximation, its close time-complexity is $O(\log(h)F(h))$, where $F(h)$ is the complexity for computing the $f(x)/f'(x)$ with a h-digit precision [54]. Since the deriving procedures cost $O(M)$ time, calculating the n approximations of the center points has an overall time complexity of $O(Mn + n\log(h)F(h))$.
2) The second one is a rough approximation of the solution of (15). The geometric mean $GM_{l,i} = (\prod_{j=1}^{M} r_{l-1,i}^j)^{1/M}$, can be considered as a fast but imprecise estimation of $r_{l,i}^0$, whose calculation complexity is $O(M)$.

In the remainder of this paper, we use approximation by ANR and GM to represent the results of the methods, respectively.

The process for sampling the next population of individuals are presented as follows. Given the center vector $\mathcal{R}_l^0 = (r_{l,1}^0, r_{l,2}^0, \ldots, r_{l,n}^0)$, for each component, the probability value can be calculated according to (12). Then the sampling process for each component in the solution can be carried out in a straightforward way from these probability values.

Algorithm 2: Pseudo Code of BDEDA

Begin
Step 1: Set the initial population $D_0 = \emptyset$, the selected population $D_0^{Se} = \emptyset$, generation counter $Ge = 0$.
Step 2: *Initialization*: generate $D_0 = \{\mathcal{R}^1, \mathcal{R}^2, \ldots, \mathcal{R}^N\}$ at random and make correction procedure.
Step 3: **While** $Ge < Ge_{\max}$
(1) *Evaluation*:
For each solution $\mathcal{R}$ in D_{Ge}, calculate $S(\mathcal{R}) = \mathcal{W}\mathcal{R}^T$.
(2) *Selection*:
Sequence the individuals in $S(\mathcal{R})$ non-increasing order.
$D_{Ge}^{Se} \leftarrow$ select the top M individuals.
(3) *Parameters learning*:
For $i = 1, 2, \ldots, n$
ANR: Construct and solve (15) with Newton-Raphson algorithm.
or GM: Calculate the geometric mean.
EndFor
(4) *Sampling*:
For $i = 1, 2, \ldots, n$
Compute the probabilities following (12).
EndFor
$D_{Ge+1} \leftarrow$ sample $N - 1$ individuals.
Make correction procedure.
(5) Set $Ge = Ge + 1$.
EndWhile
Step 4: The best individuals in D_{Ge} is the final solution.
End

D. Algorithm Presentation

The initial population for the algorithm is generated randomly. In order to deal with the solutions that do not comply with the deadline constraint, a correction procedure is proposed as follows.

Step 1: Initial $j \leftarrow 1$.
Step 2: For each component r_i in one solution $\mathcal{R}$, calculate the pseudo utility $(p_{ir_i} - p_{ik})/(k - r_i)$ (if $r_i = k$, then the pseudo utility is set to 0).
Step 3: Order the components in the pseudo utility nonincreasing order.
Step 4: Set the corresponding resource allocation to k of the job in the jth position in the order.
Step 5: Calculate C_n: if $C_n \le d$ then the procedure ends; otherwise, set $j \leftarrow j + 1$ and go to step 4.

The procedure tries to improve the current solution by adding maximal resource to the job with the largest pseudo utility. In the worst case, if all resource allocations are set to k and the solution still cannot comply with the constraint, then the problem has no feasible solution. Note that each C_n can be obtained by calculating the current reduction of processing time $p_{ir_i} - p_{ik}$, which requires $O(n)$ time. Steps 2 and 4 also can be completed in $O(n)$ time. The ordering operation in step 3 has a complexity of $O(n \log n)$. Thus the correction procedure for each solution has a complexity of $O(n \log n)$.

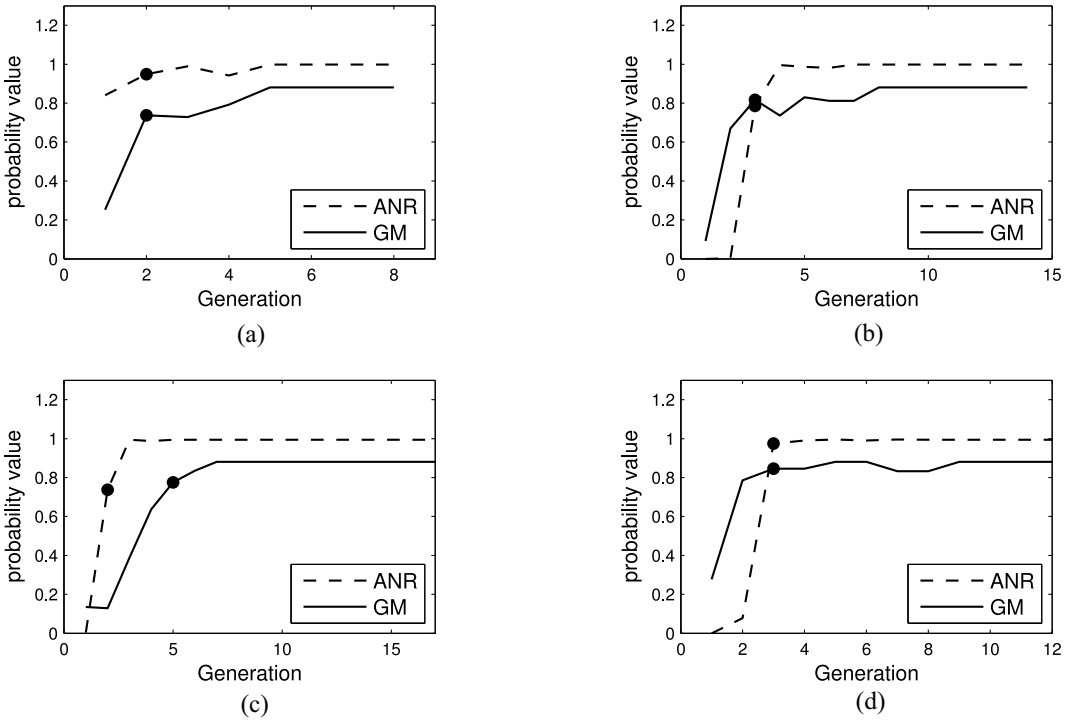


Fig. 1. Performance comparisons between ANR and GM on different instances. The curves show the changes of the probability of the optimal value of the first component. The dots denote the first generation where one component can obtain its final probability value. Results for the (a) 5×5 instance, (b) 5×10 instance, (c) 10×5 instance, and (d) 10×10 instance.

With the probabilistic model and the processes of learning and sampling, BDEDA is proposed as follows.

Unless stated specifically, the parameter settings employed for BDEDA in the following experiments are as follows.

1) *Population:* The population size is set to $N = 10n$.
2) *Selection Mechanism:* The truncation selection mechanism is applied, which chooses the best 10% individuals in each generation, $M = 0.1N$.
3) *Elitism Criteria:* The elitism criteria is employed such that in each generation $10n - 1$ individuals are sampled.
4) *Stopping Criterion:* The stopping criterion is set to be maximum evaluations. For an instance of configuration $n \times k$, the maximum generation is set to $n \times k \times 5$, where $n \times k$ denotes the configuration of n jobs with k resource allocation possibilities. The pseudo code of BDEDA is presented in Algorithm 2.

E. Computational Cost Analysis

In this subsection, the computational cost of BDEDA is analyzed following the steps in Algorithm 2.

1) Initialization: The initial solutions are generated at random with the time complexity of $O(Nn)$. To make each solution feasible, correction procedures are employed which has a complexity of $O(Nn \log n)$. Therefore, the total complexity of initialization is $O(Nn \log n)$.

2) Evaluation: For each individual, computing the objective values costs $O(n)$ time and calculating the total completion time requires $O(n)$ time. Therefore, the total complexity of evaluation is $O(Nn)$.

3) Selection: The selection process can be carried out by ordering the solutions according to the objective values which has a complexity of $O(N \log N)$.

4) Parameter Learning:

a) *ANR:* As was mentioned, calculating ANRs has a close complexity to $O(Mn + n \log(h)F(h))$. In this paper, h is set to $h = 10^{-3}$.

b) *GM:* Calculating the geometric mean of selected solutions takes $O(Mn)$ time.

5) Sampling: Calculating the distance parameters costs $O(nk)$ time. Obtaining the best fitness value in the last generation has a complexity of $O(M)$. Computing the probability values costs $O(Nkn)$ time. Generating the solutions according to these probability values can be done in $O(nk)$ time. For each individual, verifying whether it complies with the deadline constraint costs $O(n^2k)$ time. The employed correction procedure has a complexity of $O(Nn \log n)$. Therefore, the sampling process in each generation has a total complexity of $O(Nn^2k)$.

In the conclusion, each iteration of BDEDA with ANR has a close complexity to $O(Nn^2k + n \log(h)F(h))$ and BDEDA with GM has a complexity of $O(Nn^2k)$.

F. Evaluation of ANR and GM

In this section, we carry out a comparison between ANR and GM. Given that there is no available datasets in the literature for the discrete model, we have generated benchmark instances using the procedure explained in the Appendix. The generated instances will be used in this and posterior sections. In this experimentation, the stopping criterion is specifically designed: for each $n \times k$ instance, the stopping criterion is set to be maximum execution time $n \times k \times 5$ s.

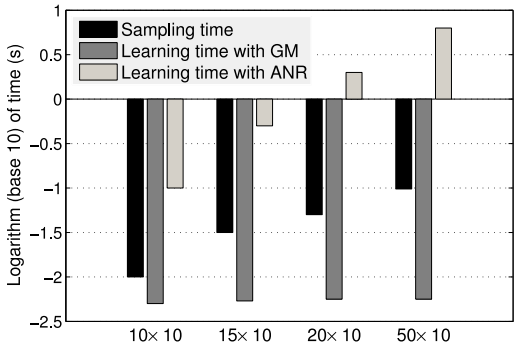


Fig. 2. Logarithm (base 10) of the average sampling and learning time on each configuration. In order to better compare the learning and sampling time, the logarithm values are used.

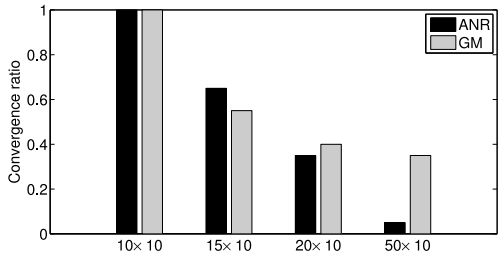


Fig. 3. Convergence ratio on each configuration.

For the comparison of the accuracy and time cost of ANR and GM, the following experiment is conducted. The configurations employed in this experiment are 5×5, 5×10, 10×5, 10×10, and one instance is generated for each configuration. Let $\mathcal{R}^{n\times k} = (r_1^{n\times k}, \ldots, r_n^{n\times k})$ denote the optimal solution obtained by an exhaustive algorithm of the corresponding instance. Each EDA (with ANR or GM) was executed for 5 times on each instance. The results that were kept track include: 1) the average probability of the optimal value of the first component in each generation, $p(r_1 = r_1^{n\times k})$ and 2) the first generation where one component obtains its final probability value. The results can be seen in Fig. 1.

In Fig. 1, for all the instances, ANR achieved higher probability values and had a faster convergence speed than GM. The generations in which the first convergence happens for all components were either 2 or 3, which makes it better than GM from this aspect. GM had a reasonable convergence speed. Except for the 10×10 case, ties happened between ANR and GM with respect to the first generation where one component can obtain its final probability value.

The further experiments were conducted to validate the suitability of ANR and GM on medium- and large-scale problems. 5 instances were generated for the configurations, 10×10, 15×10, 20×10, and 50×10. For each instance, BDEDA was executed with ANR and GM for ten times, respectively. During the executions, the average learning and sampling time were recorded for each configuration. The average sampling time was calculated from all executions for each configuration, since no change was made to the sampling procedure. The results are shown in Figs. 2 and 3.

Note that in Fig. 2, the average learning time with ANR was relatively huge compared with GM as well as with the average sampling time. The ANR gradually became the most time-consuming process when applied to medium-scale problems. As to the convergence ratios, for small-scale instances (with 10 or 15 jobs), ANR had advantages over GM. However, when the job numbers exceeds 20, the convergence ratio was not competitive, since the learning procedure took up too much computation time which resulted in the phenomenon that only a small amount of evaluations were executed.

In the conclusion, the ANR parameter learning is more suitable than GM if sufficient execution time is allowed. Although being more accurate than GM, ANR costs too much computation time. Therefore, when applied to medium- and large-scale problems, ANR may lose its advantages. Though GM has some shortcomings (relatively low optimal probability value and convergence ratio, for example), BDEDA with GM is capable of maintaining a tradeoff between time-complexity and accuracy for large-scale problems. On the basis of the idea of the improvement of the performance of BDEDA, the following strategy is employed. ANR is applied for the small-scale problems with less than 20 jobs and GM for the medium- and large-scale problems.

IV. Experimental Study

Since the problem is a new one, there is no state-of-the-art algorithms. However, there are numerous comparative approaches in literature capable of being employed. We have used standard versions of several algorithms in order to compare our proposal. The experiments are divided into four parts. Firstly, we compare Boltzmann distribution with multinomial distribution based on information entropy. Secondly, the convergence speed for different algorithms are compared under different problem settings. Thirdly, the performance of all the algorithms on small-scale instances are compared, for which the optimal solutions can be given by an exhaustive algorithm. At last, the performance of these algorithms on large-scale problems are evaluated.

The benchmarks employed in this section are generated at random and the corresponding parameter settings are presented in the Appendix. Note that the parameter $\alpha = 0.1$ is fixed in the experimental study. All the approaches were coded in MATLAB R2013b (8.2.0.701) and executed on a 2.2 GHz, Core II, 2 GB RAM PC with Win8 OS (64-bit).

A. Comparative Approaches

This subsection is devoted to the introduction of several comparative approaches. Note that there exist many optimization algorithms that can be adopted for solving the considered problem. We introduce the comparative approaches from the following categories: 1) exhaustive algorithms; 2) local-search-based approaches; and 3) EAs.

Considering the fact that for small-scale problems, the optimal solution is accessible, an exhaustive algorithm can generate the optimal solution, so that all approaches can be evaluated based on the objective gaps. The local-search-based approaches are introduced as attempts to solve the problem based on the neighborhood structure. It is also necessary to

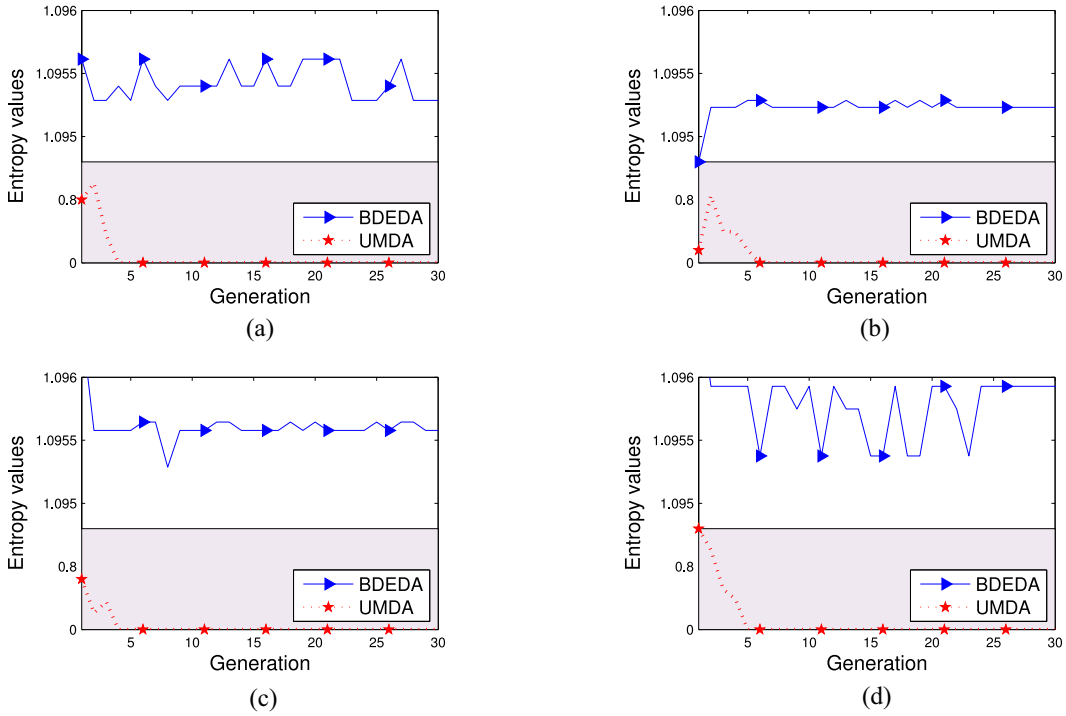


Fig. 4. Entropy values for all the components of BDEDA and UMDA on the first 4×3 instance. The vertical axis is set to be nonuniform in order to better illustrate the changes in the curves. The areas colored white and gray have different coordinate scales. Entropy values for (a) r_1, (b) r_2, (c) r_3, and (d) r_4.

compare BDEDA with other EAs and thus several classical and yet powerful EAs are introduced. We think that with this wide range of proposal a fair and complete comparison can be conducted.

1) Exhaustive Algorithm: In this part, a branch and bound algorithm is developed. A depth-first search is adopted in the branching procedure, which allocates resources to each job in a forward manner. It begins with an empty resource allocation set $(0, 0, \ldots, 0)$, allocating one unit resource allocation to the first position so that the nodes starts from $(1, 0, \ldots, 0)$, and finally arrives at $(k, k, \ldots, k)$ with the maximum resource allocation in each position. Suppose in one step we are at the node $(r_1, r_2, \ldots, r_i, -, \ldots,)$ and its descendants are $(r_1, r_2, \ldots, r_i, 1, \ldots, -)$ to $(r_1, r_2, \ldots, r_i, k, \ldots, -)$, when the first resource allocation that satisfies the deadline constraint d is found, then all the proceeding nodes are pruned. If a new node is generated, a lower bound for the weighted resource consumption is calculated. If it is greater than or equal to the initial solution, then the node and all the nodes beyond it in that branch are eliminated. If a completed node has a weighted resource consumption less than the initial solution, then it is substituted as the new solution. Otherwise, it is eliminated. The step continues until all the nodes are explored.

2) Local-Search-Based Approaches: In this part, two local-search-based approaches are introduced: simple tabu search (STS [55], [56]) and shotgun hill climbing (SHC [57]). The neighborhood structure for STS and SHC is the same: for each solution $\mathcal{R}=(r_1, r_2, \ldots, r_n)$, the solution $\mathcal{R}'$ is considered its neighborhood, if $\mathcal{R}'$ is obtained by decreasing only one unit to any component in $\mathcal{R}$. That is

$$N_u(\mathcal{R}) = \left\{\mathcal{R}' | r_s = r'_s - 1 \text{ and } r'_i = r_i, \forall\, i \neq s\right\}. \tag{17}$$

Tabu search has been extensively employed in scheduling problems and achieved considerable performance. The initial solution of tabu search is provided by a greedy search, which starts from the solution $(1, 1, \ldots, 1)$, and each time adds one unit resource allocation to job i with the lowest $w_i/(p_{ir_i} - p_{ir_{i+1}})$. If the completion time of the last job is smaller than d or all jobs are allocated k units resources, the search ends.

3) EAs: Three meta-heuristics are provided in this part.

Univariate marginal distribution algorithm (UMDA) was firstly introduced in [43], it considers a model that factorizes a product of univariate marginal distributions. UMDA belongs to the simplest and yet powerful EDAs in [27]. UMDA was initially defined for binary problems. The version of UMDA employed hereafter shares the same assumption with the standard UMDA that all the components are independent. Since each variable in the solution takes more than two values, they are modeled by means of a multinomial distribution. Therefore, UMDA and BDEDA differ in the probability distributions that are used to model the variables: UMDA employs the multinomial distribution and BDEDA follows an approximation of the Boltzmann distribution.

The second EA approach, M_k-EDA [30], [31], [58] is based on l-order Markov model. In this probabilistic model, the configuration of the variable x_i depends on the configurations of the previous l variables, where l is a parameter given in advance. In this paper, the two-order Markov model is employed.

The third EA approach employed is the ant colony system (ACS [59], [60]). The ACS parameter settings in [59] are employed.

TABLE I
AVERAGE ENTROPY GAPS OF BDEDA AND UMDA

	Generation			
Instance	1	5	10	30
4×3	0.224	1.050	1.093	1.095
10×5	0.044	0.980	1.602	1.602
20×5	-0.120	0.923	1.601	1.602
30×5	0.070	0.880	1.134	1.601

Our last comparative approach is the conventional GA (CGA [61]–[64]). For a fair comparison, the stop criterion for all the EAs are consistent with that of BDEDA.

B. Comparing BDEDA With UMDA

This part is aimed at the comparison of the Boltzmann and multinomial distribution. We employ the concept of information entropy to carry out the comparisons.

Entropy is a measure of unpredictability of information content [65], [66]. The higher the entropy, the higher the uncertainty of a distribution. In this section, we will provide experimentation evidences to show that the Boltzmann distribution has the ability of obtaining more diverse populations than the multinomial distribution. Maintaining a more diverse population can make BDEDA have a high probability of detecting more promising regions in the search space.

The experimentation was conducted on configurations 4×3, 10×5, 20×5, and 20×10. For each configuration, five instances were generated. In each generation, the entropy value for each component are recorded. Fig. 4 shows the results on the first 4×3 instance. Note that in Fig. 4, in order to show the changes in the entropy curves and make comparisons in each subfigure, the vertical axis is set to be nonuniform. The parts colored white and gray have different scales.

It can be easily seen from Fig. 4 that in almost all cases, the Boltzmann distribution can achieve greater entropy values than the multinomial distribution. For each component, the entropy value of the multinomial distribution quickly converged to nearly 0, while the entropy values of Boltzmann distribution stayed in a reasonable level until the recorded generation.

In order to show the results of all instances, the average entropy value of r_1 were calculated in the 1st, 5th, 10th, and 20th generations. Table I shows the gaps (the average entropy values of the Boltzmann distribution minus those of the multinomial distribution) of all instances in the experimentation.

It can be easily seen, that except the first generation of configuration 20×5, the Boltzmann distribution achieved high entropy values than the multinomial distribution. This means that the Boltzmann model obtains more diverse populations than the multinomial distribution in these cases. On the other hand, due to the number of parameters of the multinomial distribution, their entropy values quickly reduce to nearly 0, which can be considered as a reasonable explanation for why UMDA has a very fast convergence speed in Section IV-C and obtains relatively noncompetitive solutions in Sections IV-D and IV-E.

C. Convergence Analysis

This part is devoted to the analysis of the convergence of the compared approaches under different problem settings. For the problem under investigation, the value of deadline parameter determines the feasible solution space. For the same instance, changing the deadline to a smaller one constrains the feasible solution space.

This experimentation is carried out on the configurations 10×5, 10×10, and 20×10. Ten instances are generated for each configuration. To ensure comparable search spaces, three deadline parameters d_1, d_2, and d_3 are generated with $\varepsilon_1 = 0.2$, $\varepsilon_2 = 0.5$, and $\varepsilon_3 = 0.8$ by the following expression:

$$d_i = \sum_{j=1}^{n} p_{jk} + \varepsilon_i \left(\sum_{j=1}^{n} p_{j1} - \sum_{j=1}^{n} p_{jk} \right). \tag{18}$$

The EA approaches, ACS, CGA, $M_k EDA_2$, UMDA, and BDEDA were evaluated. For each instance, each algorithm was executed for five times. Note that in order to compare ACS with other EAs, the number of evaluations was converted into a parameter corresponding to the generations of other EAs. The generation where the final result first appears was recorded for each execution. Since all the approaches share the same stopping criterion, the generation can reflect the convergence speed. The spans among the recorded generations for each case show the stability of the corresponding approaches. Fig. 5 presents the box plots of the results for all the compared approaches.

The convergence analysis was conducted based on the convergence speed and stability. Note that UMDA had clear advantages with respect to these two aspects. Therefore, the following analysis was conducted within ACS, CGA, M_k-EDA_2, and BDEDA.

1) Convergence Speed: For the configuration 10×5, considering the median generations, no clear differences can be found for BDEDA, CGA, ACS, and M_k-EDA_2 in Fig. 5(a). However, for the configuration 10×10 BDEDA had obvious advantages over CGA, ACS, and M_k-EDA_2. For the configuration 20×10, BDEDA and ACS had higher convergence speed than CGA and M_k-EDA_2. The median generation of BDEDA for each case of the configurations 10×10 and 20×10 was lower than the corresponding result of either CGA or M_k-EDA_2. The expansion of the solution space had less influence on BDEDA than on CGA and M_k-EDA_2. No obvious differences can be found between ACS and BDEDA with respect to convergence speed in this experimentation.

2) Convergence Stability: The span of each box reflects the stability of the corresponding approach. As can be seen from Fig. 5, for all the configurations, BDEDA had advantages over CGA, since each box span for BDEDA are smaller compared with the corresponding result of CGA. Compared with M_k-EDA_2, BDEDA was more stable on the configurations 10×5 and 10×10. However, no obvious differences between BDEDA and M_k-EDA_2 can be found in Fig. 5(c) with respect to the box spans. BDEDA achieves smaller spans on the configuration 10×10 than ACS. However, no corresponding differences can be found on configurations 10×5 and 20×10.

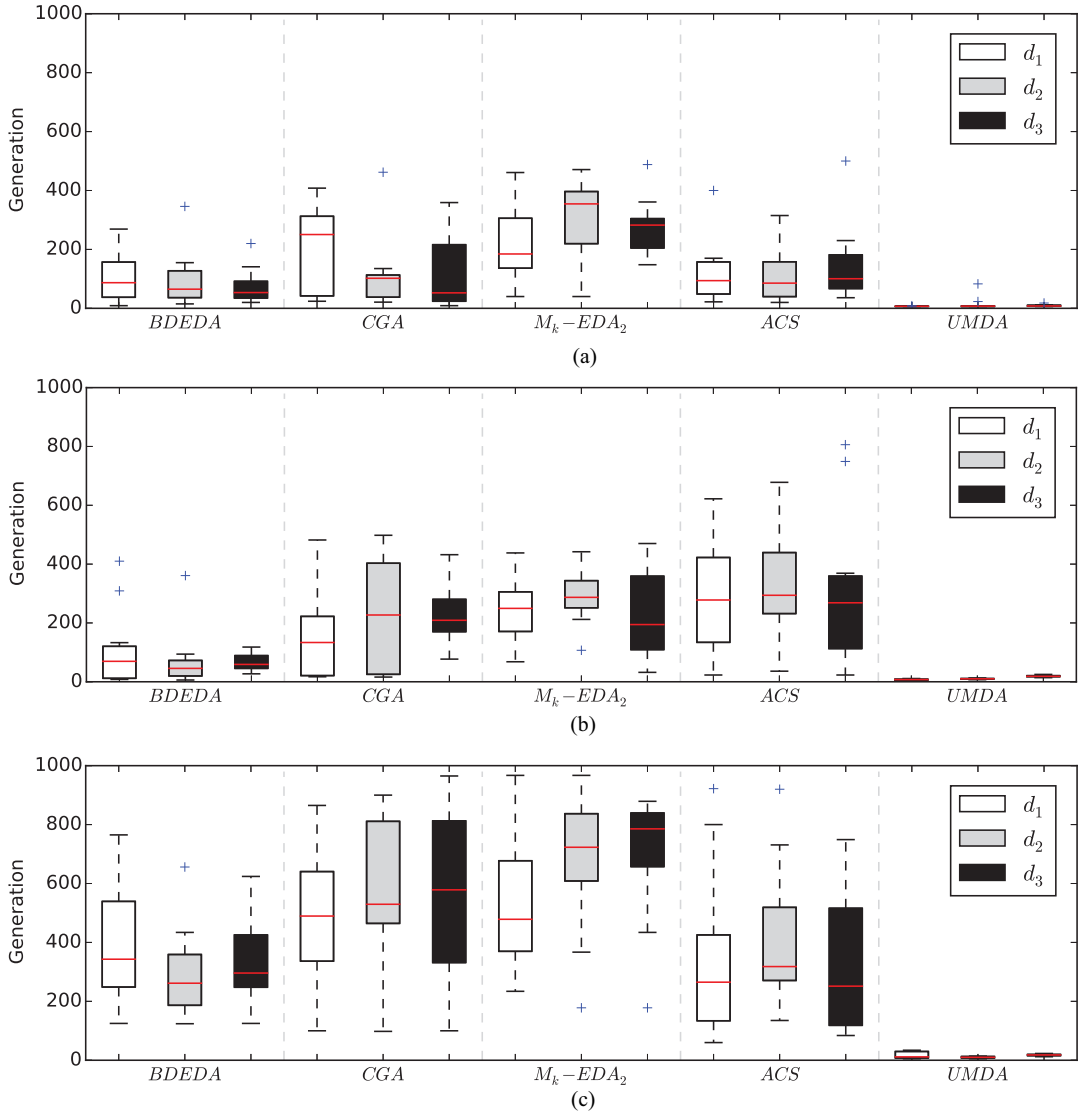


Fig. 5. Box plots of the generations where the final results first appear for the compared approaches. Each box displays the results of BDEDA, CGA, M_k-EDA$_2$, ACS, and UMDA with d_1, d_2, and d_3 on the configuration of (a) 10×5, (b) 10×10, and (c) 20×10. The spacings between the different parts of the box help indicate the degree of dispersion (spread) and skewness in the results for each case, which in this paper represents the stabilities of the relevant approaches. The band inside the box is the median generation for each case.

In the conclusion, BDEDA achieves competitive convergence speed among all the compared EAs. Besides, BDEDA has advantages over CGA and ACS with respect to the convergence stability.

D. Performance Comparison on Small-Scale Instances

In this section, the performance of each algorithm is compared on small-scale instances for which the developed exhaustive algorithm can present the optimal solutions in reasonable time. The configurations employed range from 4×10 to 10×10.

For each configuration, five instances are generated. Each EA algorithm was executed for 10 times for each instance. The parameters recorded are $AD_1 \pm \text{STD}$, where

$$AD_1 = \left(\sum_{i=1}^{10} \frac{A_i - \text{Opt}}{\text{Opt}} \right) \Big/ 10 \tag{19}$$

stands for the average deviation, A_i denotes the solution found by each algorithm during its ith execution, Opt represents the optimal solution gained by the exact algorithm and STD represents the standard deviation of the solutions obtained by ten times executions for each instance.

As can be seen in Table II, the efficiency of the compared algorithms varied along with the sizes of the instances. As to the lowest AD_1, BDEDA had obvious advantages over other approaches, which achieved 44 best AD_1s out of total 60 instances. STS can obtain five best AD_1s, ACS 20, SHC 3, CGA 7, M_k-EDA$_2$ 4, and UMDA 3. Moreover, other than the exhaustive algorithm, BDEDA is the only approach that can obtain the optimal solutions for all instances. There is no obvious differences among SHC, CGA, M_k-EDA$_2$, and UMDA with respect to the ability to achieve the optimal solutions.

In the conclusion, for small-scale instances, BDEDA can obtain all the optimal solutions within ten execution times. As

TABLE II

PERFORMANCE COMPARISON ON SMALL-SCALE INSTANCES. RESULTS IN BOLD DENOTES THE ALGORITHM ACHIEVES THE LOWEST AD_1 AMONG ALL THE ALGORITHMS. RESULTS MARKED WITH $*$ REPRESENTS THAT THE ALGORITHM OBTAINED AT LEAST ONCE THE OPTIMAL SOLUTION

Instance	STS	ACS	SHC	CGA	M_k-EDA$_2$	UMDA	BDEDA	Instance	STS	ACS	SHC	CGA	M_k-EDA$_2$	UMDA	BDEDA
4×10	**104***	**0.00**±0.00*	0.01±0.00*	0.02±0.01*	**0.00**±0.00*	0.10±0.04*	**0.00**±0.00*	8×6	267	0.01±0.00*	0.03±0.01*	0.09±0.02*	0.06±0.03*	0.05±0.02*	**0.00**±0.00*
	170	**0.00**±0.00*	0.03±0.01*	0.03±0.01*	0.01±0.00*	0.13±0.06*	**0.00**±0.00*		228	**0.01**±0.00*	0.04±0.02*	**0.01**±0.01*	0.02±0.01*	0.02±0.01*	**0.01**±0.00*
	249	0.01±0.00*	**0.00**±0.00*	0.02±0.01*	0.02±0.01*	0.10±0.04*	**0.00**±0.00*		337	0.01±0.00*	0.02±0.01*	0.03±0.01*	0.01±0.01*	0.01±0.01*	**0.00**±0.00*
	76*	**0.00**±0.00*	0.02±0.00*	0.06±0.04	0.05±0.02*	0.10±0.04*	**0.00**±0.00*		239	0.02±0.01*	0.03±0.01*	0.01±0.01*	0.01±0.01*	0.01±0.01*	**0.00**±0.00*
	265	**0.00**±0.00*	0.03±0.00*	0.03±0.02*	0.02±0.01*	0.10±0.05*	0.01±0.01*		279	0.01±0.00*	0.04±0.01*	0.02±0.01	0.01±0.01*	0.01±0.00*	**0.00**±0.00*
4×15	**75***	**0.00**±0.00*	0.03±0.01*	0.04±0.02*	0.02±0.01*	0.28±0.16*	**0.00**±0.00*	8×8	160	**0.00**±0.00*	0.03±0.00*	0.03±0.02*	0.04±0.03*	0.05±0.03*	0.01±0.00*
	160	**0.01**±0.00*	0.02±0.00*	0.12±0.07*	0.06±0.01*	0.16±0.09*	0.04±0.01*		347	0.03±0.00*	0.04±0.02*	**0.01**±0.00*	0.15±0.07	0.04±0.02*	0.02±0.00*
	313	**0.00**±0.00*	0.03±0.01*	0.03±0.01*	0.04±0.02*	0.13±0.07*	0.03±0.01*		228	0.04±0.01*	0.06±0.02*	0.03±0.02*	0.11±0.06	0.06±0.03*	**0.00**±0.00*
	155	**0.01**±0.00*	0.03±0.01*	0.10±0.06*	0.07±0.05*	0.28±0.15	0.03±0.01*		**186***	0.06±0.02*	0.09±0.03	0.02±0.01*	0.17±0.07	0.03±0.02*	**0.00**±0.00*
	134	0.04±0.00*	0.06±0.01*	0.12±0.08*	0.07±0.03*	0.30±0.15	**0.00**±0.00*		285	0.10±0.04*	0.08±0.01*	0.03±0.01	0.07±0.04*	0.03±0.02*	**0.01**±0.00*
4×20	550	0.02±0.00*	0.11±0.04	0.10±0.05*	0.02±0.01*	**0.01**±0.00*	0.02±0.01*	8×10	**160***	**0.00**±0.00*	**0.00**±0.00*	0.03±0.01*	0.27±0.11	0.07±0.02	0.06±0.00*
	154	**0.03**±0.01*	0.09±0.03*	0.11±0.07*	0.08±0.04*	0.17±0.12*	0.04±0.02*		463	**0.00**±0.00*	**0.00**±0.00*	0.03±0.01*	0.12±0.06	0.03±0.01*	0.02±0.00*
	93	**0.04**±0.01*	0.07±0.02*	0.19±0.05*	0.20±0.15*	0.27±0.16	0.06±0.02*		192	**0.00**±0.00*	0.04±0.01*	0.04±0.02*	0.29±0.15	0.10±0.07	0.01±0.00*
	580	0.03±0.00*	0.06±0.00*	**0.02**±0.00*	**0.02**±0.00*	0.11±0.05*	**0.02**±0.01*		217	0.04±0.01*	0.07±0.02*	0.04±0.04*	0.28±0.12	0.07±0.02*	**0.01**±0.00*
	650	0.05±0.02*	0.08±0.01*	0.09±0.04*	0.04±0.02*	0.15±0.10*	**0.01**±0.00*		242	0.05±0.02*	0.03±0.01*	0.07±0.02*	0.26±0.11	0.08±0.04*	**0.01**±0.00*
6×10	205	0.02±0.00*	0.03±0.00*	0.08±0.03*	0.07±0.04*	0.05±0.03*	**0.00**±0.00*	10×6	390	0.03±0.00*	0.06±0.02*	0.03±0.01*	0.10±0.03	0.02±0.01*	**0.00**±0.00*
	183	**0.01**±0.00*	0.06±0.02*	0.04±0.02*	0.02±0.01*	0.07±0.02*	**0.01**±0.01*		264	0.04±0.01*	0.08±0.04*	0.04±0.01*	**0.01**±0.00*	0.16±0.06	**0.01**±0.00*
	557	0.02±0.00*	0.04±0.02*	0.01±0.00*	0.01±0.00*	0.01±0.00*	**0.00**±0.00*		503	0.01±0.00*	0.03±0.01*	0.02±0.01*	0.05±0.02*	0.13±0.07	**0.00**±0.00*
	304	**0.00**±0.00*	0.03±0.00*	0.06±0.02*	0.07±0.02	0.02±0.01*	0.02±0.00*		330	**0.02**±0.00*	0.04±0.02*	**0.02**±0.00*	0.09±0.04	0.18±0.09*	**0.02**±0.00*
	327	0.01±0.00*	0.04±0.01*	0.04±0.02*	0.13±0.06	0.04±0.01*	**0.00**±0.00*		224	**0.02**±0.00*	0.06±0.01*	0.03±0.02*	0.08±0.04*	0.13±0.05	0.03±0.00*
6×15	405	0.04±0.01*	0.09±0.03*	0.09±0.05*	0.16±0.09	0.12±0.07	**0.00**±0.00*	10×8	608	0.03±0.00*	0.11±0.03*	0.02±0.01*	**0.01**±0.00*	0.09±0.02*	**0.01**±0.00*
	192	0.06±0.02*	0.10±0.05	0.14±0.10	0.17±0.12	0.09±0.05*	**0.05**±0.02*		721	0.03±0.00*	0.07±0.03*	0.02±0.01*	0.11±0.04	0.03±0.02*	**0.00**±0.00*
	168	0.02±0.00*	0.04±0.01*	0.07±0.01*	0.18±0.09*	0.10±0.04*	**0.00**±0.00*		710	**0.01**±0.00*	0.08±0.03*	0.02±0.01*	0.09±0.04*	0.02±0.01*	**0.01**±0.00*
	379	0.04±0.01*	0.22±0.08	0.02±0.01*	0.17±0.08	0.08±0.01	**0.01**±0.00*		671	0.04±0.01*	0.03±0.00	**0.02**±0.01*	0.14±0.02	**0.02**±0.01*	0.04±0.00*
	213	0.06±0.02*	0.11±0.04	0.10±0.07*	0.10±0.05*	0.07±0.02*	**0.01**±0.00*		1082	0.06±0.01*	0.08±0.03*	**0.01**±0.01*	0.29±0.11	0.14±0.06	**0.01**±0.00*
6×20	329	0.07±0.02	0.17±0.06*	0.14±0.08	0.33±0.14	0.24±0.16	**0.04**±0.03*	10×10	830	0.06±0.01*	0.07±0.00*	0.03±0.01*	0.10±0.03	**0.02**±0.01*	0.03±0.00*
	594	0.07±0.03*	0.20±0.08	0.10±0.04*	0.24±0.09*	0.12±0.04*	**0.02**±0.01*		776	0.04±0.00*	0.13±0.04	**0.03**±0.00*	0.14±0.03	0.04±0.02*	**0.03**±0.00*
	126	0.08±0.02	0.09±0.02*	0.19±0.11	0.21±0.11	0.32±0.16	**0.07**±0.03*		608	0.04±0.00*	0.09±0.02*	0.05±0.02	0.18±0.08	0.05±0.03*	**0.01**±0.01*
	229	0.13±0.05	0.19±0.09	0.12±0.03	0.30±0.10	0.31±0.11	**0.04**±0.03*		879	0.06±0.01*	0.14±0.05	0.03±0.02*	0.27±0.08	0.04±0.02*	**0.01**±0.00*
	293	0.06±0.02*	0.15±0.02	0.18±0.05	0.21±0.07	0.18±0.04*	**0.04**±0.01*		609	0.07±0.02*	0.12±0.03*	0.04±0.02*	0.23±0.05	0.03±0.01*	**0.02**±0.00*

TABLE III
PERFORMANCE COMPARISON ON LARGE-SCALE INSTANCES. RESULTS IN BOLD DENOTE THAT THE ALGORITHM ACHIEVES THE LOWEST AD_2 AMONG ALL THE COMPARED ALGORITHMS. RESULTS MARKED WITH $*$ REPRESENTS THAT THE ALGORITHM OBTAINS AT LEAST ONCE THE SOLUTIONS IN BR WITHIN TEN EXECUTIONS

Best	STS	ACS	SHC	CGA	M_k-EDA$_2$	UMDA	BDEDA	Best	STS	ACS	SHC	CGA	M_k-EDA$_2$	UMDA	BDEDA
50×5								**50×10**							
1954	2376	0.06*	0.12	0.05*	0.11	0.01*	**0.00***	2304	3308	0.09*	0.27	0.08*	0.24	0.11	**0.01***
969	1447	0.18	0.13	0.27	0.53	0.04*	**0.01***	3022	4074	0.17	0.25	0.14	0.26	**0.04***	**0.04***
2000	2743	0.08*	0.19	0.09	0.17	0.17	**0.04***	1974	2508	0.13	0.31	0.15	0.36	**0.07**	**0.07***
2570	3278	0.11	0.22	0.13	0.13	0.08*	**0.06***	2948	4196	0.19*	0.06	0.12	0.42	**0.03***	**0.03***
1727	2493	0.10	0.27	0.15	0.20	0.18	**0.05***	1837	2694	0.12	0.29	0.09	0.24	0.14	**0.05***
100×5								**100×10**							
2676	3864	0.19	0.23	0.27	0.77	0.10	**0.06***	4410	5935	0.25	0.31	0.21	0.58	0.08	**0.04***
4190	4903	0.17	0.35	0.15	0.39	0.10	**0.07***	4762	6739	0.18	0.30	0.17	0.81	0.14	**0.05***
4435	4607	0.09	0.26	0.25	0.47	0.09*	**0.04***	2042	4832	0.23	0.40	0.15	0.41	0.08*	**0.04***
3234	4077	0.25	0.12	0.16	0.13	0.08*	**0.06***	2570	4183	0.19	0.28	0.19	0.33	0.08*	**0.02***
2723	3873	0.41	0.18	0.13	0.20	0.06*	**0.04***	3776	4983	0.38	0.47	0.17	0.36	0.06*	**0.02***
150×5								**150×10**							
6943	8904	0.20	0.25	0.17	0.52	0.05*	**0.01***	7820	12834	0.24	0.23	0.24	0.47	0.12	**0.08***
4976	8493	0.16	0.37	0.22	0.73	0.14	**0.06***	6837	9673	0.09*	0.28	0.18	0.36	0.06*	**0.03***
7631	9473	0.19	0.27	0.18	0.46	0.09	**0.03***	9238	14967	0.17	0.45	0.27	0.63	0.14	**0.05***
7148	8735	0.23	0.33	0.29	0.42	0.13	**0.06***	7236	9864	0.21	0.17	0.12	0.42	0.12	**0.05***
5976	7983	0.08*	0.36	0.14	0.38	**0.07***	**0.07***	7390	10365	0.27	0.24	0.23	0.40	0.07*	**0.02***

to the parameter AD_1 employed in the experiment, BDEDA achieved most of the lowest AD_1s, which suggests that it can be a strong alternative for solving small-scale instances.

E. Performance Comparison on Large-Scale Instances

In order to demonstrate that BDEDA still maintains its advantages over the compared approaches on medium- and large-scale instances, the following experiment is conducted. The instances employed in this subsection were with the following configurations: 1) 50 × 5; 2) 50 × 10; 3) 100 × 5; 4) 100 × 10; 5) 150 × 5; and 6) 150 × 10. For each configuration, five instances were generated. All approaches except the exhaustive algorithm are compared in this experimentation. For each instance, all EA algorithms were executed for ten times. The parameter recorded in the results is

$$AD_2 = \left(\sum_{i=1}^{10} \frac{A_i - \text{best}}{\text{best}} \right) \Big/ 10 \tag{20}$$

where best represents the best solution obtained during all the executions for each instance. The results are shown in Table III. The results in bold denotes the algorithm that achieves the lowest AD_2. The results marked with $*$ represent that the corresponding algorithm obtains at least once the results in the BR (best region) within ten executions, where BR = [best, $(1 + 5\%) \times$ best].

As can be seen from Table III, BDEDA achieved all the lowest AD_2s and at least once the solution in BR for all the instances in each configuration. STS, SHC, and M_k-EDA$_2$ had the least satisfying performance, since they achieved not even once the lowest AD_2 or the solutions in the best regions. UMDA obtained 15 solutions in BR region and four lowest AD_2s. There is no obvious difference between ACS and CGA. CGA obtained two solutions in BR within ten repetitions and ACS 6. Both ACS and CGA achieved no lowest AD_2s.

In the conclusion, as to the parameters kept track of in the experiments, for solving either small- or large-scale problems, BDEDA is capable of presenting the best performance. For the optimization of the problem under investigation, BDEDA has the following advantages.

1) The convergence speed of BDEDA is competitive, especially for the problems with high search spaces.
2) BDEDA has a good ability to discover the promising regions near the optimal solutions for small- and large-scale problems.
3) The solutions gained by BDEDA are more stable than the compared approaches.

V. CONCLUSION

In this paper, a scheduling problem with controllable processing times subject to a common deadline is analyzed and tackled. The discrete model is implemented for the indication of the one-to-one mapping relations between the resource allocations and processing times. We show that a simplified version of the problem can be polynomially reduced to a partition problem, which demonstrates its NP-completeness. In order to solve the problem, we present an EDA. In the EDA approach, each component is associated with an approximation of the Boltzmann distribution. In the learning procedure, we provided two approximation methods of different complexities and accuracies. On the basis of the results of the conducted experiment, a strategy for how to improve the performance of BDEDA is presented.

In order to evaluate the performance of BDEDA several comparative approaches are implemented. The experiment showed that BDEDA had competitive convergence speed and stability. Among all the compared approaches, BDEDA obtained the best performance for both the small- and large-scale problems.

However, it should be mentioned that there is still room for improvement. The estimation methods in the learning procedure for small- and large-cases are different. Although ANR has a reasonable computation cost, its computation time takes up most of the execution time in each iteration. Despite the

fact that GM seems to maintain a tradeoff between accuracy and time complexity, it has relatively low optimal probability value and convergence ratio, which may greatly affect the convergence speed and solution quality under extreme conditions. Future research work could focus on the presentation of more adaptive approximation methods for the estimation of the distribution.

In this paper, we have assumed independence between the components, however, it is strongly believed that by taking advantage of the interrelation between them (for example, the dependencies between center points) it may bring a faster convergence speed or lead to the improvement of the ability to find optimal regions.

With the aim of improving the performance of BDEDA, hybrid approaches can be also developed based on the characteristics of BDEDA. Moreover, it is considered to be an interesting issue to test BDEDA on other problems with integer-based solutions.

APPENDIX

Parameter Settings for Benchmarks

The benchmark employed in all the experiments are generated randomly. All the parameters are sampled from uniform distributions and the relevant parameters settings are presented as follows.

1) *The Weight Parameter w_i:* w_i is sampled from a uniform distribution from integer set 10–20.
2) *The Processing Time:* For each job i, p_{i1} is generated from a uniform distribution from 15 to 20. To generate $p_{ij}, j = 2, \ldots, k$, firstly a gap set $\{g_1, g_2, \ldots, g_{k-1}\}$ is sampled from a unform distribution from 0 to p_{i1}/k independently and then the processing times are calculated in a forward way as $p_{ij} = p_{i[j-1]} - g_{j-1}, j = 2 \ldots, n$.
3) *The Deadline Parameter d:* d is sampled from a uniform distribution from $\sum_{i=1}^{n} p_{ik}$ to $\sum_{i=1}^{n} p_{i1}$ to guarantee feasible solutions for each benchmark.

REFERENCES

[1] S. Henningsson, K. Hyde, A. Smith, and M. Campbell, "The value of resource efficiency in the food industry: A waste minimisation project in East Anglia, U.K.," *J. Clean. Prod.*, vol. 12, no. 5, pp. 505–512, 2004.

[2] C. Lang-Koetz, N. Pastewski, and H. Rohn, "Identifiying new technologies, products and strategies for resource efficiency," *Chem. Eng. Technol.*, vol. 33, no. 4, pp. 559–566, 2010.

[3] C. I. Pardo Martínez, "Energy use and energy efficiency development in the German and Colombian textile industries," *Energy Sustain. Develop.*, vol. 14, no. 2, pp. 94–103, 2010.

[4] M. Neelis *et al*., "Energy efficiency developments in the Dutch energy-intensive manufacturing industry, 1980–2003," *Energy Policy*, vol. 35, no. 12, pp. 6112–6131, 2007.

[5] A. Janiak, "One-machine scheduling with allocation of continuously-divisible resource and with no precedence constraints," *Kybernetika*, vol. 23, no. 4, pp. 289–293, 1987.

[6] D. Biskup and H. Jahnke, "Common due date assignment for scheduling on a single machine with jointly reducible processing times," *Int. J. Prod. Econ.*, vol. 69, no. 3, pp. 317–322, 2001.

[7] N. V. Shakhlevich and V. A. Strusevich, "Pre-emptive scheduling problems with controllable processing times," *J. Sched.*, vol. 8, no. 3, pp. 233–253, 2005.

[8] R. Vickson, "Choosing the job sequence and processing times to minimize total processing plus flow cost on a single machine," *Oper. Res.*, vol. 28, no. 5, pp. 1155–1167, 1980.

[9] R. Vickson, "Two single machine sequencing problems involving controllable job processing times," *AIIE Trans.*, vol. 12, no. 3, pp. 258–262, 1980.

[10] E. Nowicki and S. Zdrzałka, "A survey of results for sequencing problems with controllable processing times," *Discrete Appl. Math.*, vol. 26, no. 2, pp. 271–287, 1990.

[11] D. Shabtay and G. Steiner, "A survey of scheduling with controllable processing times," *Discrete Appl. Math.*, vol. 155, no. 13, pp. 1643–1666, 2007.

[12] Z.-L. Chen, Q. Lu, and G. Tang, "Single machine scheduling with discretely controllable processing times," *Oper. Res. Lett.*, vol. 21, no. 2, pp. 69–76, 1997.

[13] A. Janiak and M. Y. Kovalyov, "Single machine scheduling subject to deadlines and resource dependent processing times," *Eur. J. Oper. Res.*, vol. 94, no. 2, pp. 284–291, 1996.

[14] C. D. Ng, T. E. Cheng, and M. Y. Kovalyov, "Single machine batch scheduling with jointly compressible setup and processing times," *Eur. J. Oper. Res.*, vol. 153, no. 1, pp. 211–219, 2004.

[15] T. E. Cheng and M. Y. Kovalyov, "Single machine batch scheduling with deadlines and resource dependent processing times," *Oper. Res. Lett.*, vol. 17, no. 5, pp. 243–249, 1995.

[16] C. D. Ng, T. E. Cheng, A. Janiak, and M. Y. Kovalyov, "Group scheduling with controllable setup and processing times: Minimizing total weighted completion time," *Ann. Oper. Res.*, vol. 133, nos. 1–4, pp. 163–174, 2005.

[17] B. Alidaee and A. Ahmadian, "Two parallel machine sequencing problems involving controllable job processing times," *Eur. J. Oper. Res.*, vol. 70, no. 3, pp. 335–341, 1993.

[18] Z.-L. Chen, "Simultaneous job scheduling and resource allocation on parallel machines," *Ann. Oper. Res.*, vol. 129, nos. 1–4, pp. 135–153, 2004.

[19] M. Karimi-Nasab and S. F. Ghomi, "Multi-objective production scheduling with controllable processing times and sequence-dependent setups for deteriorating items," *Int. J. Prod. Res.*, vol. 50, no. 24, pp. 7378–7400, 2012.

[20] T. Cheng, A. Janiak, and M. Y. Kovalyov, "Single machine batch scheduling with resource dependent setup and processing times," *Eur. J. Oper. Res.*, vol. 135, no. 1, pp. 177–183, 2001.

[21] D. Shabtay, "Single and two-resource allocation algorithms for minimizing the maximal lateness in a single machine," *Comput. Oper. Res.*, vol. 31, no. 8, pp. 1303–1315, 2004.

[22] D. Shabtay and M. Kaspi, "Parallel machine scheduling with a convex resource consumption function," *Eur. J. Oper. Res.*, vol. 173, no. 1, pp. 92–107, 2006.

[23] D. Shabtay and G. Steiner, "Single machine batch scheduling to minimize total completion time and resource consumption costs," *J. Sched.*, vol. 10, nos. 4–5, pp. 255–261, 2007.

[24] L. Yedidsion, D. Shabtay, and M. Kaspi, "A bicriteria approach to minimize maximal lateness and resource consumption for scheduling a single machine," *J. Sched.*, vol. 10, no. 6, pp. 341–352, 2007.

[25] L. Yedidsion, D. Shabtay, E. Korach, and M. Kaspi, "A bicriteria approach to minimize number of tardy jobs and resource consumption in scheduling a single machine," *Int. J. Prod. Econ.*, vol. 119, no. 2, pp. 298–307, 2009.

[26] R. G. Michael and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*. San Francisco, CA, USA: W. H. Freeman, 1979.

[27] P. Larrañaga and J. A. Lozano, *Estimation of Distribution Algorithms: A New Tool for Evolutionary Computation*, vol. 2. Boston, MA, USA: Kluwer, 2002.

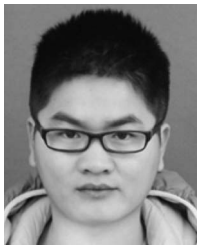
[28] J. A. Lozano, *Towards a New Evolutionary Computation: Advances on Estimation of Distribution Algorithms*, vol. 192. Berlin, Germany: Springer, 2006.

[29] P. Larrañaga, R. Etxeberria, J. A. Lozano, and J. M. Peña, "Combinatorial optimization by learning and simulation of Bayesian networks," in *Proc. 16th Conf. Uncertainty Artif. Intell.*, San Francisco, CA, USA, 2000, pp. 343–352.

[30] R. Santana, P. Larrañaga, and J. A. Lozano, "Protein folding in 2-dimensional lattices with estimation of distribution algorithms," in *Biological and Medical Data Analysis*. Berlin, Germany: Springer, 2004, pp. 388–398.

[31] R. Santana, P. Larrañaga, and J. A. Lozano, "Protein folding in simplified models with estimation of distribution algorithms," *IEEE Trans. Evol. Comput.*, vol. 12, no. 4, pp. 418–438, Aug. 2008.

[32] J. Ceberio, E. Irurozki, A. Mendiburu, and J. A. Lozano, "A distance-based ranking model estimation of distribution algorithm for the flow-shop scheduling problem," *IEEE Trans. Evol. Comput.*, vol. 18, no. 2, pp. 286–300, Apr. 2014.
[33] E. S. Hou, N. Ansari, and H. Ren, "A genetic algorithm for multiprocessor scheduling," *IEEE Trans. Parallel Distrib. Syst.*, vol. 5, no. 2, pp. 113–120, Feb. 1994.
[34] J. C. Bean, "Genetic algorithms and random keys for sequencing and optimization," *ORSA J. Comput.*, vol. 6, no. 2, pp. 154–160, 1994.
[35] C. R. Reeves, "A genetic algorithm for flowshop sequencing," *Comput. Oper. Res.*, vol. 22, no. 1, pp. 5–13, 1995.
[36] S. Ahire, G. Greenwood, A. Gupta, and M. Terwilliger, "Workforce-constrained preventive maintenance scheduling using evolution strategies," *Decis. Sci.*, vol. 31, no. 4, pp. 833–859, 2000.
[37] J. C. Tay and N. B. Ho, "Evolving dispatching rules using genetic programming for solving multi-objective flexible job-shop problems," *Comput. Ind. Eng.*, vol. 54, no. 3, pp. 453–473, 2008.
[38] L. Wang and D.-Z. Zheng, "A modified evolutionary programming for flow shop scheduling," *Int. J. Adv. Manuf. Technol.*, vol. 22, nos. 7–8, pp. 522–527, 2003.
[39] N. Sinha, R. Chakrabarti, and P. Chattopadhyay, "Fast evolutionary programming techniques for short-term hydrothermal scheduling," *Electr. Power Syst. Res.*, vol. 66, no. 2, pp. 97–103, 2003.
[40] B.-B. Li and L. Wang, "A hybrid quantum-inspired genetic algorithm for multiobjective flow shop scheduling," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 37, no. 3, pp. 576–591, Jun. 2007.
[41] H. Mühlenbein, J. Bendisch, and H.-M. Voigt, "From recombination of genes to the estimation of distributions II. Continuous parameters," in *Parallel Problem Solving From Nature—PPSN IV*. Berlin, Germany: Springer, 1996, pp. 188–197.
[42] P. A. Bosman *et al.*, "Linkage information processing in distribution estimation algorithms," in *Proc. Genet. Evol. Comput. Conf. (GECCO)*, Orlando, FL, USA, 1999, pp. 60–67.
[43] H. Mühlenbein, "The equation for response to selection and its use for prediction," *Evol. Comput.*, vol. 5, no. 3, pp. 303–346, 1997.
[44] S. Baluja, "Population-based incremental learning: A method for integrating genetic search based function optimization and competitive learning," Dept. Comput. Sci., Carnegie Mellon Univ., Pittsburgh, PA, USA, Tech. Rep. CMU-CS-94-163, 1994.
[45] G. R. Harik, F. G. Lobo, and D. E. Goldberg, "The compact genetic algorithm," *IEEE Trans. Evol. Comput.*, vol. 3, no. 4, pp. 287–297, Nov. 1999.
[46] J. S. De Bonet *et al.*, "MIMIC: Finding optima by estimating probability densities," in *Advances in Neural Information Processing Systems*. Cambridge, MA, USA: MIT Press, 1997, pp. 424–430.
[47] R. Santana *et al.*, "MATEDA-2.0: Estimation of distribution algorithms in MATLAB," *J. Stat. Softw.*, vol. 35, no. 7, pp. 1–30, 2010.
[48] S. Baluja and S. Davies, "Combining multiple optimization runs with optimal dependency trees," Dept. Comput. Sci., Carnegie Mellon Univ., Pittsburgh, PA, USA, Tech. Rep. CMU-CS-97-157, 1997.
[49] G. Harik, "Linkage learning via probabilistic modeling in the ECGA," *Urbana*, vol. 51, no. 61, p. 801, 1999.
[50] R. Etxeberria and P. Larranaga, "Global optimization using Bayesian networks," in *Proc. 2nd Symp. Artif. Intell. (CIMAF)*, Havana, Cuba, 1999, pp. 332–339.
[51] H. Mühlenbein and T. Mahnig, "Evolutionary optimization and the estimation of search distributions with applications to graph bipartitioning," *Int. J. Approx. Reasoning*, vol. 31, no. 3, pp. 157–192, 2002.
[52] H. Mühlenbein, T. Mahnig, and F. Ais, "Evolutionary algorithms and the Boltzmann distribution," in *Foundations of Genetic Algorithms 7*. San Mateo, CA, USA, 2003, pp. 525–556.
[53] H. Mühlenbein and T. Mahnig, "Mathematical analysis of evolutionary algorithms," in *Essays and Surveys in Metaheuristics*. Boston, MA, USA: Springer, 2002, pp. 525–556.
[54] J. M. Borwein and P. B. Borwein, *Pi and the AGM: A Study in the Analytic Number Theory and Computational Complexity*. New York, NY, USA: Wiley, 1987.
[55] F. Glover, "Tabu search—Part I," *ORSA J. computing*, vol. 1, no. 3, pp. 190–206, 1989.
[56] F. Glover, "Tabu search—Part II," *ORSA J. Comput.*, vol. 2, no. 1, pp. 4–32, 1990.
[57] S. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*, vol. 25. Eaglewood Cliffs, NJ, USA: Prentice-Hall, 1995.
[58] S. Shakya and J. McCall, "Optimization by estimation of distribution with DEUM framework based on Markov random fields," *Int. J. Autom. Comput.*, vol. 4, no. 3, pp. 262–272, 2007.
[59] M. Dorigo and L. M. Gambardella, "Ant colony system: A cooperative learning approach to the traveling salesman problem," *IEEE Trans. Evol. Comput.*, vol. 1, no. 1, pp. 53–66, Apr. 1997.
[60] M. Dorigo and C. Blum, "Ant colony optimization theory: A survey," *Theor. Comput. Sci.*, vol. 344, no. 2, pp. 243–278, 2005.
[61] D. E. Goldberg, *Genetic Algorithms in Search Optimization and Machine Learning*. Reading, MA, USA: Addison-Wesley, 1989.
[62] J. H. Holland, "Outline for a logical theory of adaptive systems," *J. ACM (JACM)*, vol. 9, no. 3, pp. 297–314, 1962.
[63] J. H. Holland, *Adaptation in Natural and Artificial Systems*. Cambridge, MA, USA: MIT Press, 1992.
[64] J. H. Holland, *Adaptation in Natural and Artificial Systems: An Introductory Analysis With Applications to Biology, Control, and Artificial Intelligence.* Ann Arbor, MI, USA: Univ. Michigan Press, 1975.
[65] E. T. Jaynes, "Information theory and statistical mechanics," *Phys. Rev.*, vol. 106, no. 4, p. 620, 1957.
[66] T. M. Cover and J. A. Thomas, *Elements of Information Theory*. New York, NY, USA: Wiley, 1991.



Xinle Liang received the B.Eng.Mgt. degree from Beijing Institute of Technology, Beijing, China, in 2011. He is currently working toward the Ph.D. degree from the School of Computer Science and Technology, University of Science and Technology of China, Hefei, China.

His research interests include single and multiobjective evolutionary algorithms, planning and scheduling problems, and optimization applications in service computing.

Huaping Chen received the Ph.D. degree from University of Science and Technology of China, Hefei, China, 1997.

He is a Professor with the School of Computer Science and Technology, University of Science and Technology of China. His research interests include evolutionary algorithms, nonlinear programming, project scheduling, and resource scheduling problems.

Jose A. Lozano (M'05) received the B.S. degrees in mathematics and computer science and the Ph.D. degree from University of the Basque Country, Leioa, Spain, in 1991, 1992, and 1998, respectively.

Since 2008 he has been a Full Professor with the Department of Computer Science and Artificial Intelligence, University of the Basque Country, where he leads the Intelligent System Group. His research interests include evolutionary computation, machine learning, data mining, pattern analysis, probabilistic graphical models, and bioinformatics. He has edited three books and has published over 60 refereed journal papers.

Prof. Lozano is an Associate Editor of IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION and an Editorial Board Member of *Evolutionary Computation Journal*, *Soft Computing*, and three other journals.