Hindawi Publishing Corporation
Mathematical Problems in Engineering
Volume 2012, Article ID 786923, 24 pages
doi:10.1155/2012/786923

Research Article

A Hybrid Multiobjective Genetic Algorithm for Robust Resource-Constrained Project Scheduling with Stochastic Durations

Jian Xiong,[1,2] Ying-wu Chen,[1] Ke-wei Yang,[1,3] Qing-song Zhao,[1] and Li-ning Xing[1]

[1] *Department of Management Science and Engineering, College of Information System and Management, National University of Defense Technology, Hunan, Changsha 410073, China*
[2] *School of Engineering and Information Technology, University of New South Wales at the Australian Defence Force Academy, Canberra ACT 2600, Australia*
[3] *Department of Computer Science, The University of York, York YO10 5GH, UK*

Correspondence should be addressed to Jian Xiong, xiongjian1984@hotmail.com

Received 22 September 2011; Revised 23 November 2011; Accepted 7 December 2011

Academic Editor: Sri Sridharan

We study resource-constrained project scheduling problems with perturbation on activity durations. With the consideration of robustness and stability of a schedule, we model the problem as a multiobjective optimization problem. Three objectives—makespan minimization, robustness maximization, and stability maximization—are simultaneously considered. We propose a hybrid multiobjective evolutionary algorithm (H-MOEA) to solve this problem. In the process of the H-MOEA, the heuristic information is extracted periodically from the obtained nondominated solutions, and a local search procedure based on the accumulated information is incorporated. The results obtained from the computational study show that the proposed approach is feasible and effective for the resource-constrained project scheduling problems with stochastic durations.

1. Introduction

As for its practical importance in a wide range of real-world application areas, project scheduling problems (PSPs) have received considerable attention in the operation research field. PSPs usually address two matters: resource and time [1]. Given scarce resources, the problem is modelled as a resource-constrained project scheduling problem (RCPSP), which is a general type of PSPs that contains job-shop, flow-shop, and open-shop problems as special cases [2]. As RCPSP is an NP-hard problem [1], heuristics or metaheuristics are preferred to solve it. For details about the models and methods of the RCPSP, readers are referred to several comprehensive surveys [3–7].

It is very common that the execution of a schedule will be in the presence of uncertain factors. With the consideration of uncertainty, the problem can be modelled as stochastic RCPSP. One of the main avenues to solve the stochastic project scheduling problems is to develop robust schedules that can deal with uncertain circumstances. Robustness maximization is often taken as an objective in stochastic project scheduling problems. This methodology is referred to as *robust scheduling*. Van de Vonder et al. [8] presented concepts of *quality robustness* and *solution robustness* and addressed the issue of achieving a tradeoff between these two robustness measures.

For stochastic RCPSP, the conventional objective, makespan minimization, and the new introduced objective, robustness maximization, are in conflict with each other to a certain degree. In other words, stochastic RCPSP is inherently a multiobjective optimization problem. It is interesting to note that although stochastic project scheduling has considerable importance in practical applications, the works on solving this problem by multiobjective evolutionary approaches are rather scant [9]. Al-Fawzan and Haouari [10] proposed to measure robustness of a given schedule via the total amount of free slack of all activities. The free slack was also referred to as floating time by Abbasi et al. [11]. Chtourou and Haouari [12] followed the approach of Al-Fawzan and Haouari but assigned weights to the free slack of an activity according to the number of its successors and/or the sum of its required resources. Lambrechts et al. [13] studied the stochastic RCPSP with uncertainty resource availability. The authors developed proactive strategies to build a robust schedule that meets the project deadline and minimizes the schedule instability cost, indicated as the deviations between the planned and the actually realized activity starting times during the execution. Ballestín and Leus [14] investigated various objective functions related to timely project completion of RCPSP with stochastic activity durations. The authors developed a Greedy Randomized Adaptive Search Procedure (GRASP) to produce high-quality solutions. Ashtiani et al. [15] introduced a new class of scheduling policies for solving RCPSP with stochastic activity durations. The authors underlined the value of preprocessing in stochastic scheduling. In the process, a subset of sequencing choices at the beginning of the planning horizon was made and the rest of the scheduling decisions to future points were relegated in time.

In the present paper, we focus on the RCPSP with stochastic activity durations. The term "robustness" used in the remainder of this paper refers to *quality robustness*, which represents the schedule's ability to cope with the perturbation on activity durations. We measure the robustness as the difference between the planned makespan and the expectation of the makespan in the real execution under stochastic environment. By considering a schedule's stability, the problem is modelled as a three-objective optimization problem, where makespan minimization, robustness maximization, and stability maximization are simultaneously taken into account. A hybrid multiobjective genetic algorithm incorporating a local search procedure is proposed to solve the problem.

2. Problem Description

2.1. Deterministic Resource-Constrained Project Scheduling

RCPSP is the optimization problem to schedule the interrelated activities (tasks, operations) of a project with consideration of minimizing the makespan while given precedence constraints among activities and resource constraints are satisfied.

RCPSP can be formally described as follows: a project consists of a set of activities $N = N_0, N_1, \ldots, N_n, N_{n+1}$, where each activity has to be executed without interruption. N_0 and

N_{n+1} are dummy activities which, respectively, represent the start and end activity. The number of available resource types is K and the vector of resources availability is denoted as $R = (R_1, R_2, \ldots, R_K)$, where R_k $(k = 1, \ldots, K)$ represents the availability of each resource type k in each time period. For each activity N_i $(i = 1, \ldots, n)$, the duration is denoted as d_i, and the requested resources vector over the entire duration is represented as $r_i = (r_{i1}, \ldots, r_{iK})$. Note that for dummy activities N_0 and N_{n+1}, the durations and the required resources are both 0. An activity-on-node direct acyclic graph, denoted as $G(N, A)$, is employed to represent the precedence relations amongst the activities. The direct successors and predecessors of an activity N_i are denoted as $Succ_i$ and $Pred_i$, respectively. The activities are interrelated by two types of constraints: precedence constraint and resource constraint. The former constraint indicates that each activity should be scheduled after all of its predecessor activities are accomplished. The latter constraint ensures that for each type of resource, the amount of being occupied cannot exceed the total availability at any time point. Each activity has start and finish times, denoted as st_i and ft_i, respectively. A schedule S consists of the vector of start time of the nondummy activities, denoted as $S = (st_1, \ldots, st_n)$. The makespan of the schedule S is denoted as $M(S)$.

2.2. Resource-Constrained Project Scheduling with Stochastic Duration

However, in reality, it is quite common that execution of a schedule may be confronted with uncertain factors. Although there might be multiple uncertain factors affecting a schedule's execution, we just consider the situation that the activity durations are subject to perturbation, which is also a most common uncertainty addressed in the related literature. For example, in an airlift task scheduling, the extreme bad weather will possibly have a negative impact on the execution of the tasks.

Under the effect of perturbation, the durations of some activities will be delayed and, finally, the perturbation might result in delay to the whole schedule. The delay amount is corresponding to the magnitude of perturbation, which is depicted by the following two elements: *perturbation strength* and *perturbation period*. The former element represents the strength of a perturbation, which is denoted as St and is normalized in the interval [0, 1]. The closer St is to 1, the more strength the perturbation is and vice versa. The perturbation period is used to indicate the duration of perturbation and is determined by the occurrence time and the end time of the perturbation, denoted as OT and ET, respectively. In the presence of perturbation, actual start time and finish time of an activity are represented as st_i^u and ft_i^u, respectively. The actual duration is denoted as d_i^u and is calculated as follows:

$$d_i^u = \begin{cases} \text{Round}((1 + St)d_i) + 1, & \text{if } st_i^u \in [OT, ET] \\ & \text{or } st_i^u + d_i \in [OT, ET], \\ d_i, & \text{otherwise,} \end{cases} \tag{2.1}$$

where Round$(\cdot)$ is the function of obtaining the integer part of a real value. We assume that if any part of the execution of an activity is affected by the perturbation, then the whole activity is considered to be affected.

In this paper, we assume that perturbation strength, perturbation occurrence time and end time follow normal probability distribution, which is a most often used distribution in real-world application. In a specific realization of the perturbation, denoted as μ, the actual

makespan of a schedule S is denoted as $M^{\mu}(S)$. In the presence of perturbation, robust predictive schedules are preferred to cope with the impact on the durations. Here we indicate the robustness of a schedule under a specific realization of uncertainty, denoted as $\mathrm{Rob}^{\mu}(S)$, by the difference between the deterministic makespan before disruption, and the actual makespan after execution [13, 16]. The robustness under a specific realization of the uncertainty is formally defined as follows:

$$\mathrm{Rob}^{\mu}(S) = \frac{1}{\exp(M^{\mu}(S) - M(S)/M(S))}. \tag{2.2}$$

In the measurement of robustness under uncertainty with a probability distribution, an analytical closed form of the effective fitness function is usually not available [17]. In order to tackle this issue, we employ simulation method, Monte Carlo in concrete, to approximate the effective function of the robustness, denoted as the expectation value with a sample size N_{μ}. Then, the robustness value of a schedule in the presence of uncertainty, $\mathrm{Rob}(S)$, is calculated as follows:

$$\mathrm{Rob}(S) = E\left[\mathrm{Rob}^{\mu}(S)\right] = \frac{1}{N_{\mu}} \sum_{m=1}^{N_{\mu}} \mathrm{Rob}^{\mu_m}(S), \tag{2.3}$$

where μ_m is the mth realization of the uncertainty in the sample space.

In the robust design techniques, another important issue in need to be considered is the stability of the solutions. We employ the standard deviation to indicate the stability measure of the solutions, denoted as $Std[\mathrm{Rob}^{\mu}(S)]$ and calculated as follows:

$$Std\left[\mathrm{Rob}^{\mu}(S)\right] = \sqrt{\frac{1}{N_{\mu}} \sum_{m=1}^{N_{\mu}} \left(\mathrm{Rob}^{\mu_m}(S) - E[\mathrm{Rob}^{\mu}(S)]\right)^2}. \tag{2.4}$$

With the consideration of both magnitudes of robustness measure and stability of the solution, the problem can be modelled as an uncertainty-based multiobjective design problem [18], instead of combining these two measures by a linear weighted sum method, such as in [16]. What should be noted is that the difference between predesigned and actual makespan by itself has little relevance since a lower level of difference can be easily achieved by predesigning a schedule with a longer makespan. In [16], the authors defined the robustness of a schedule as the linear combination of the expected delay and the expected makespan via weighted sum method. Lambrechts et al. [13] just measured the robustness as a weighted deviation of the starting times. However, the weight coefficient or weight vector varies in different decision makers. A final solution is much dependent on the decision makers' preferences. Thus, we model the RCPSP with uncertain durations as a multiobjective optimization problem where the concurrently considered three objectives are makespan minimization,

robustness maximization, and stability maximization (standard deviation minimization). The optimization model is formally given as follows:

$$\begin{aligned}
\text{obj.}: &\ (1)\ \min f_1 = M(S),\\
&\ (2)\ \max f_2 = E\left[\text{Rob}^{\mu}(S)\right],\\
&\ (3)\ \min f_3 = Std\left[\text{Rob}^{\mu}(S)\right]\\
s.t.: &\ (1)\ \max\left\{st_j + d_j \mid N_j \in \text{Pred}_i\right\} \le st_i, \quad \forall i,\\
&\ (2)\ \max\left\{st_j^u + d_j^u \mid N_j \in \text{Pred}_i\right\} \le st_i^u, \quad \forall i,\\
&\ (3)\ \sum_{i=1}^{n} r_{ik}(t) \le R_k, \quad \forall t \text{ and } k = 1, 2, \ldots, K.
\end{aligned} \tag{2.5}$$

In the above optimization model, objective (1) represents the minimization of makespan, objectives (2) and (3), respectively, indicate the maximization of robustness and stability. Constraints (1) and (2) are precedence constraints which ensure the precedence feasibility in both deterministic and uncertain environment, that is, the activity N_i cannot be started until all of its predecessors are completed. Constraint (3) is a resource constraint, indicating that the total amount of the resources occupied at any time point cannot exceed the resource availability.

Within the framework of multiobjective optimization, the concept of dominance is defined as follows: an individual I_1 dominates individual I_2 if I_1 does at least as well as I_2 on all objectives, and strictly better on at least one. Thus, in this paper, we formally define the schedule dominance as follows.

Definition 2.1. A schedule S_1 dominates schedule S_2 if $M(S_1) \le M(S_2) \wedge E[\text{Rob}^{\mu}(S_1) \ge E[\text{Rob}^{\mu}(S_2)] \wedge Std[\text{Rob}^{\mu}(S_1)] \le Std[\text{Rob}^{\mu}(S_2)]$, with strict inequality holding for at least one objective measure.

The nondominance concept can be defined as follows.

Definition 2.2. A schedule S^* is an individual in the population. It is said to be a nondominated schedule if it is not dominated by any other individual in the population.

For the sake of clarity, we list the notations which are used to represent the RCPSP as follows.

R_k: the availability of kth type of resource, $k = 1, \ldots, K$.

r_{ik}: the amount of the kth type of resource requested by activity N_i, $i = 1, \ldots, n$.

d_i: the duration of activity N_i in the deterministic situation.

st_i: the start time of activity N_i in the deterministic situation.

ft_i: the finish time of activity N_i in the deterministic situation.

d_i^{μ}: the duration of activity N_i in the presence of perturbation.

st_i^μ: the start time of activity N_i in the presence of perturbation.

ft_i^μ: the finish time of activity N_i in the presence of perturbation.

S: the schedule, $S = (st_1, \ldots, st_n)$.

$M(S)$: the makespan of schedule S in the deterministic situation.

$M^\mu(S)$: the makespan of schedule S under a specific realization of uncertainty μ.

St: the perturbation strength.

OT: the occurrence time of the perturbation.

ET: the end time of the perturbation.

3. Methodology

As the three objectives are in conflict with each other, a multiobjective optimization approach is required to solve the problem. To do this, we employ one of the classic multiobjective evolutionary algorithms, named NSGA-II [19], as the basic optimizer for the problem. In the procedure of NSGA-II, the parent population and the offspring are combined and sorted in order to generate the next-generation population. The combined population is classified into different ranks of nondomination via a nondominated sorting mechanism. In order to maintain the diversity among nondominated solutions, a crowding-distance assignment mechanism is used in the selection of solutions in the same nondomination rank. However, the chromosome and genetic operators are redesigned for the RCPSP.

3.1. Chromosome Representation and Population Initialization

3.1.1. Chromosome Representation

In the RCPSP, a solution must contain the information about the sequence of activity execution. We employ the activity sequence proposed by Hartmann [20] to represent an individual, I, denoted as follows:

$$I = (j_0, j_1, \ldots, j_n, j_{n+1}). \tag{3.1}$$

The activity sequence is a precedence feasible permutation of the set of activities and the element at each index is the ID of the activity to be executed.

In order to generate a schedule, the Serial Schedule Generation Scheme (SSGS) is employed to perform this transformation from chromosome representation. SSGS consists of n stages for a network of n nondummy nodes, and in each stage one activity from the activity sequence is selected and executed at its earliest possible time, considering precedence and resource constraints [21]. Algorithm 1 shows the pseudocode of the procedure of SSGS.

3.1.2. Population Initialization

In the operator of population initialization, the individuals are generated serially via a two-step procedure [20, 21]: the first step is identifying the eligible activities, then one of the them is randomly selected in the second step. At a position in the activity list, an activity is said to

$i = 1$
while $i \leq n$ **do**
  $st_i = 0, \quad st'_i = 0, \quad ft^{\max} = 0$
  Obtain the maximum finish time of the predecessors: $ft^{\max} = \max\{ft_j \mid N_j \in \text{Pred}_i\}$
  Determine the earliest time point, st'_i, which satisfying the resource constraints and $st'_i \geq ft^{\max}$.
  Set $st_i = st'_i$.
end while

Algorithm 1: Pseudocode of the Serial Schedule Generation Scheme (SSGS).

be eligible if all of its predecessors have been scheduled. Formally, let $P(i)$ be the position of activity N_i in the activity list, if we look at the position q $(q = 0, 1, \ldots, n+1)$, an activity N_{q^*} is said to be an eligible activity at the position q if $P(j) < q$, for all $j, N_j \in \text{Pred}_{q^*}$. The set which consists of all eligible activities at position q is denoted as EA_q.

3.2. Genetic Operators

The crossover and the mutation operators are crucial for the performance of the genetic algorithms. The crossover operator combines two parents to generate two children. Similar with the situation in nature, crossover ensures that the children maintain some good characteristics of their parents. We employ a two-point position-based crossover operator [22, 23] in this research. For each crossover operator, two integer random numbers, r_1 and r_2, $r_1 < r_2$, are selected from the interval $[1, n]$. Usually, the two-point position-based crossover for scheduling problem can be divided into two versions [22]: in the first version, the activities outside the two selected points are inherited from one parent to the child, while in the other version, the set of activities between two randomly selected points is inherited from one parent to the child. In this research, we employ the first version of crossover operator. Let $P1$ and $P2$ be the two selected parents, $CH1$ and $CH2$ be the two new generated children. It is clear that the children consist of three parts separated by the two crossover points (r_1 and r_2). For example, $CH1$ is constructed as $(j_1^{CH1}, \ldots, j_{r_1}^{CH1}, j_{r_1+1}^{CH1}, \ldots, j_{r_2}^{CH1}, j_{r_2+1}^{CH1}, \ldots, j_n^{CH1})$, where the first part $(j_1^{CH1}, \ldots, j_{r_1}^{CH1})$ and the third part $(j_{r_2+1}^{CH1}, \ldots, j_n^{CH1})$ are inherited from the parent $P1$ and the second part $(j_{r_1+1}^{CH1}, \ldots, j_{r_2}^{CH1})$ is taken from the parent $P2$. What should be noted is that in order to maintain the precedence feasibility of the new generated children, in the inheritance of the second and the third parts, all elements that already exist in the previous parts of children should be eliminated. The construction of $CH2$ is the same with $CH1$. In [21], the authors proved this crossover operator can create precedence feasible offspring. Figure 1 shows an example of the crossover operation. Suppose a project has 10 nondummy activities, the parents $P1$ and $P2$ are $(1, 2, 3, 4, 5, 6, 7, 8, 9, 10)$ and $(1, 4, 5, 2, 3, 7, 6, 9, 8, 10)$, respectively. The crossover points are $r_1 = 4$ and $r_2 = 7$. According to the two-point crossover operation, the new generated children are $CH1 = (1, 2, 3, 4, 5, 7, 6, 8, 9, 10)$ and $CH1 = (1, 4, 5, 2, 3, 6, 7, 9, 8, 10)$.

Mutation is another important operator which is used to strengthen the algorithm's ability of exploring the unexplored area of the search space. We use the mutation operator presented by [21]. The mutation operator can be described as follows. An random integer

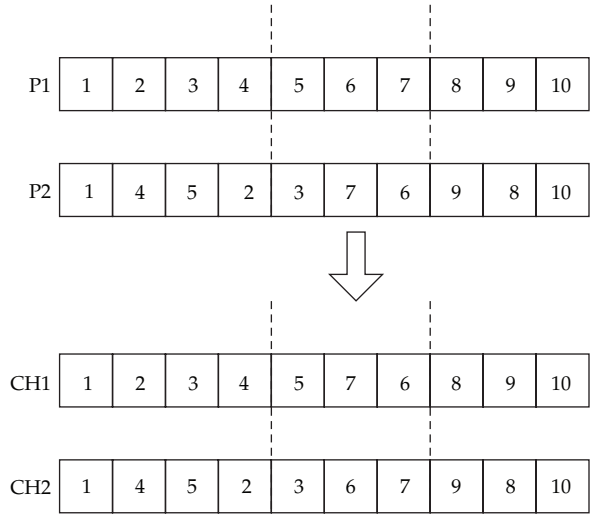

Figure 1: An example of crossover operation.

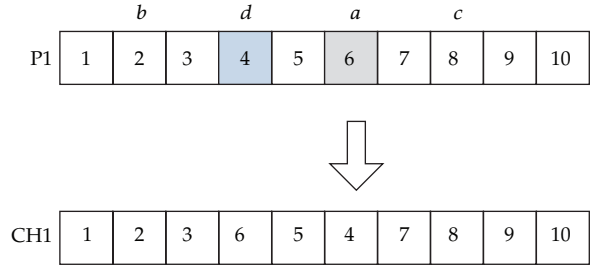

Figure 2: An example of mutation operation.

number, a, is generated from interval $[1, n]$. Let activity j'_b be the last predecessor of activity j'_a and let activity j'_c be the first successor of j'_a in the activity list. Another random integer number, d, is generated from the interval $[b + 1, c - 1]$. If $d < a$, the activity list is replaced by $(j'_0, \ldots, j'_{d-1}, j'_a, j'_d, \ldots, j'_{a-1}, j'_{a+1}, \ldots, j'_{n+1})$. If $d > a$, it is replaced by $(j'_0, \ldots, j'_{a-1}, j'_{a+1}, \ldots, j'_{d-1}, j'_a, j'_{d+1}, \ldots, j'_{n+1})$. Figure 2 shows an example of mutation operator.

3.3. *Hybrid Multiobjective Evolutionary Algorithm for RCPSP*

3.3.1. Framework of the H-MOEA

As a nature-inspired metaheuristic, multiobjective evolutionary algorithms (MOEAs) have been used successfully in the past to solve the multiobjective problems, especially the complex optimization problem. To date, there are several representative MOEAs. For the brief view history of the MOEAs and the mechanism of each MOEA, readers are referred to the survey [24]. In order to speed up the convergency of the optimization algorithm, local search procedure is usually hybridized with the MOEAs. Ishibuchi and Murata [25, 26] were among the first to implement such a hybridization. [27] improved the performance of multiobjective

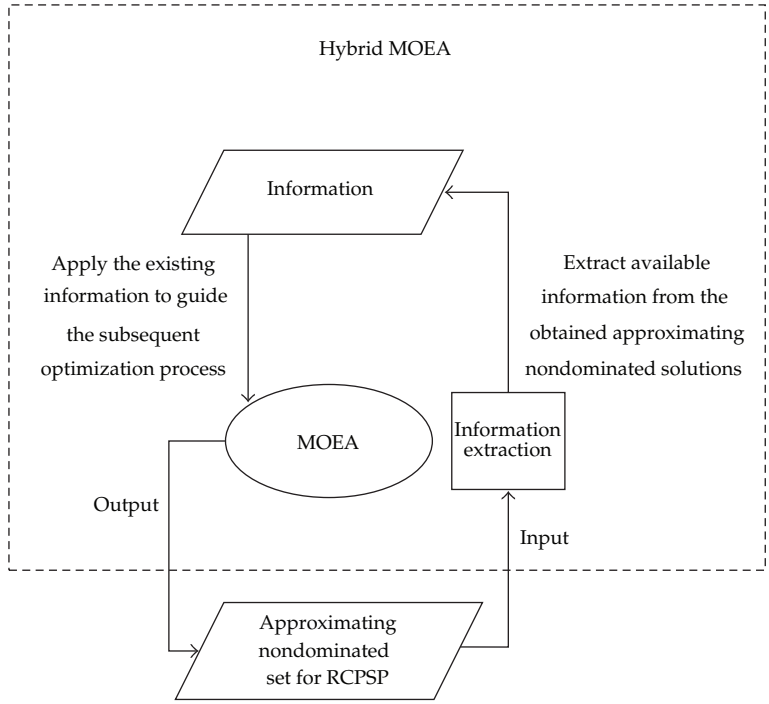

Figure 3: The conceptual framework of the hybrid multiobjective evolutionary algorithm for RCPSP.

genetic local search, and a comparative experiment study was implemented on the 0/1 knapsack problem [28].

In order to deal with the stochastic RCPSPs more effectively, a hybrid multiobjective evolutionary algorithm (H-MOEA) is proposed. In the natural-inspired metaheuristics, the individuals in the population contain a rich information about the solution. In several single-objective combinatorial optimization problems, experience shows that this heuristic information is useful to the convergence performance of the algorithms [29–32]. Different from single-objective optimization, the useful information can be obtained from the nondominated set, instead of from a single solution. In the procedure of H-MOEA, the information contained in the nondominated set is extracted periodically and utilized with a Pareto dominance relation based local search [33]. Figure 3 shows the conceptual framework of the proposed H-MOEA. From the figure, it can be seen that H-MOEA consists of two main functional modules: a search module (MOEA) and a knowledge module. The first module is used to search through the vast solution space and identify the nondominated solutions, while the knowledge module takes charge of obtaining useful information throughout the ongoing optimization process and applying this knowledge to guide the subsequent search process.

3.3.2. Heuristic Information

In this paper, we focus on the extraction and utilization of *Position Priority Information* which indicates the useful knowledge about the position of an activity in the schedule. The information used to establish an appropriate schedule priority for different activities is stored in a matrix PPI with size $n \times n$, where n is the number of nondummy activities. The element in

Table 1: Example of 6 schedules with 6 nondummy activities.

	I_1	I_2	I_3	I_4	I_5	I_6
Schedule 1	1	2	3	4	5	6
Schedule 2	1	3	2	4	6	5
Schedule 3	2	1	3	4	5	6
Schedule 4	2	3	1	5	4	6
Schedule 5	1	2	4	3	6	5
Schedule 6	3	1	2	6	4	5

Table 2: Example of the *Position Priority Information* matrix.

	I_1	I_2	I_3	I_4	I_5	I_6
Activity 1	3	2	1	0	0	0
Activity 2	2	2	2	0	0	0
Activity 3	1	2	2	1	0	0
Activity 4	0	0	1	3	2	0
Activity 5	0	0	0	1	2	3
Activity 6	0	0	0	1	2	3

the matrix $PPI(i, j)$ denotes the appearance times of an activity N_i at the position j. Tables 1 and 2 show an example of the construction of the priority information matrix. Table 1 shows 6 different schedules of a project with 6 nondummy activities. If we look at the first position I_1, for the 6 different schedules, the scheduled activities at this position are 1, 1, 2, 2, 1, 3, respectively. In other words, for the 6 different activities, the appearance times at the first position are 3, 2, 1, 0, 0 and 0, respectively. Then, the first column of the matrix PPI is accordingly constructed (see Table 2).

3.3.3. Local Search with the Heuristic Information

In H-MOEA, the feasible schedule (or part of it) can be constructed based on the obtained heuristic information. Then the two-step individual generation procedure [20, 21] can be modified as follows: the first step is identifying the eligible activities, then one of them is selected according to the *Position Priority Information*. In the second step of the modified individual generation procedure, the heuristic information stored in the matrix is transformed into the selection probability and Roulette Selection is employed to identify the next activity to be scheduled. For example, we suppose the Eligible Activity Set in the second position $EA_2 = (N_3, N_4)$, from the matrix PPI shown in Table 2, and we can see that $PPI(3, 3)$ and $PPI(4, 3)$ are 2 and 1, respectively. Then the selection probability for activities N_3 and N_4 are 2/(2+1) and 1/(2+1), respectively. We denote this Modified Individual Generation procedure as MIG.

In single-objective project scheduling problems, experience showed that good candidate schedules are usually found "fairly close" to other good schedules [34–36]. Encouraged by this experience, we employ a local search procedure around the neighborhood of the schedule in the multiobjective version of the PSPs. Let Ind_i be an individual in the population, $\text{Ind}_i^{\text{new}}$ the individual after the local search. If $\text{Ind}_i^{\text{new}}$ dominates Ind_i, then replace the Ind_i with $\text{Ind}_i^{\text{new}}$ in the population. Similar with the crossover and the mutation operators,

```
i = 1,   p = 0,   m = 0
while i ≤ population size do
  p = randomly generated value between the interval [0, 1]
  if p ≤ p_ls then
    m = randomly generated value between the interval [1,n].
    Copy the [1, m] part of the individual Ind_i to the individual Ind_i^new.
    Generate the [m + 1, n] part of the individual Ind_i^new by the operator
    MIG.
    Evaluate the individual Ind_i^new.
    if Ind_i^new dominates Ind_i then
      Replace Ind_i with Ind_i^new.
    end if
  end if
  i = i + 1
end while
```

Algorithm 2: Pseudocode of the local search procedure.

the local search is conducted with a probability rate, denoted as p_{ls}. The procedure of the local search is illustrated in Algorithm 2. In each generation, the local search is implemented on each individual with the predefined local search probability. For each local search procedure, just one neighbor is generated for the individual. However, the local search procedure employed in the current research is the simplest form. We are aware of that several issues are important in the hybridization of MOEAs and local search, such as balance between genetic search and local search [37, 38] size of examined neighborhood and choice of solutions to which local search is applied [39]. The main focus of this paper is extracting and utilizing the knowledge in RCPSP, and the above-mentioned issues and comparative study with respect to RCPSP will be addressed in our future study to improve the effectiveness of knowledge utilization and the performance of H-MOEA.

4. Case Study

4.1. Hypothetic Problem Description

In this section, we first studied a hypothetical problem to illustrate the proposed approach. Suppose that a project consists of 20 nondummy activities and there are 4 types of resources with availability $R = (48,\ 53,\ 42,\ 50)$. The precedence relation amongst activities is depicted as in Figure 4. In the network, the nodes N_0 and N_{21} are dummy nodes which, respectively, indicate the start node and the end node. The parameters about durations and required resources for each activity are listed in Table 3.

4.2. Parameter Settings

We used a population size of 40. Crossover and mutation probabilities were 0.98 and 0.1, respectively. The evolution was terminated after 200 generations. Please note that the crossover probability is defined for each individual. In the calculation of robustness, the sample size was set as 50. For the proposed H-MOEA, the rate of local search was set as 0.45. In all

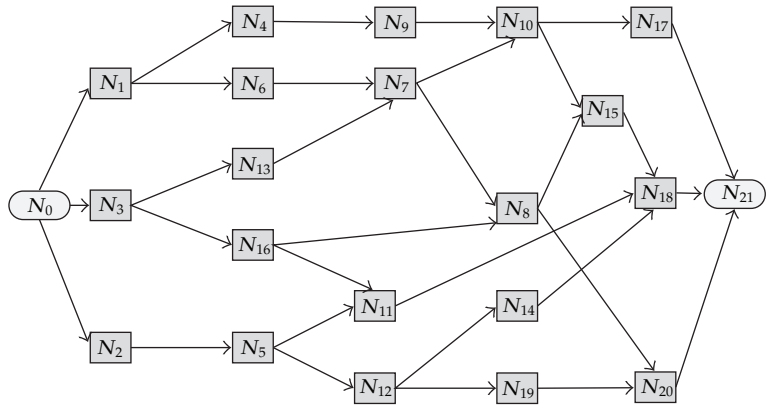

Figure 4: The precedence relation of tasks.

Table 3: Durations and required resources of the nondummy activities of the hypothetic problem.

N_i	d_i	r_i
N_1	17	(19, 20, 10, 19)
N_2	30	(13, 7, 9, 13)
N_3	23	(14, 7, 7, 20)
N_4	21	(19, 20, 20, 20)
N_5	29	(13, 15, 5, 20)
N_6	26	(14, 19, 6, 16)
N_7	29	(18, 12, 18, 6)
N_8	22	(11, 18, 10, 5)
N_9	21	(12, 16, 7, 18)
N_{10}	24	(6, 7, 19, 16)
N_{11}	26	(13, 9, 6, 15)
N_{12}	22	(18, 8, 9, 11)
N_{13}	25	(8, 13, 16, 12)
N_{14}	19	(16, 13, 6, 20)
N_{15}	24	(10, 15, 6, 18)
N_{16}	25	(17, 20, 19, 7)
N_{17}	20	(14, 22, 13, 12)
N_{18}	18	(13, 17, 20, 10)
N_{19}	27	(21, 22, 19, 17)
N_{20}	20	(16, 24, 17, 14)

of the experiments, we assumed that the perturbation interval covered the whole execution process of the schedules, that is, all of the activities were affected by the perturbation. For the hope of eliminating the effect of random generator, each algorithm was repeated 30 times with different random seeds.

4.3. Experiment Results

We studied the scenarios under uncertainties with different perturbation strengths 0.15, 0.25, and 0.35. The obtained nondominated solutions in three-dimensional space are depicted as

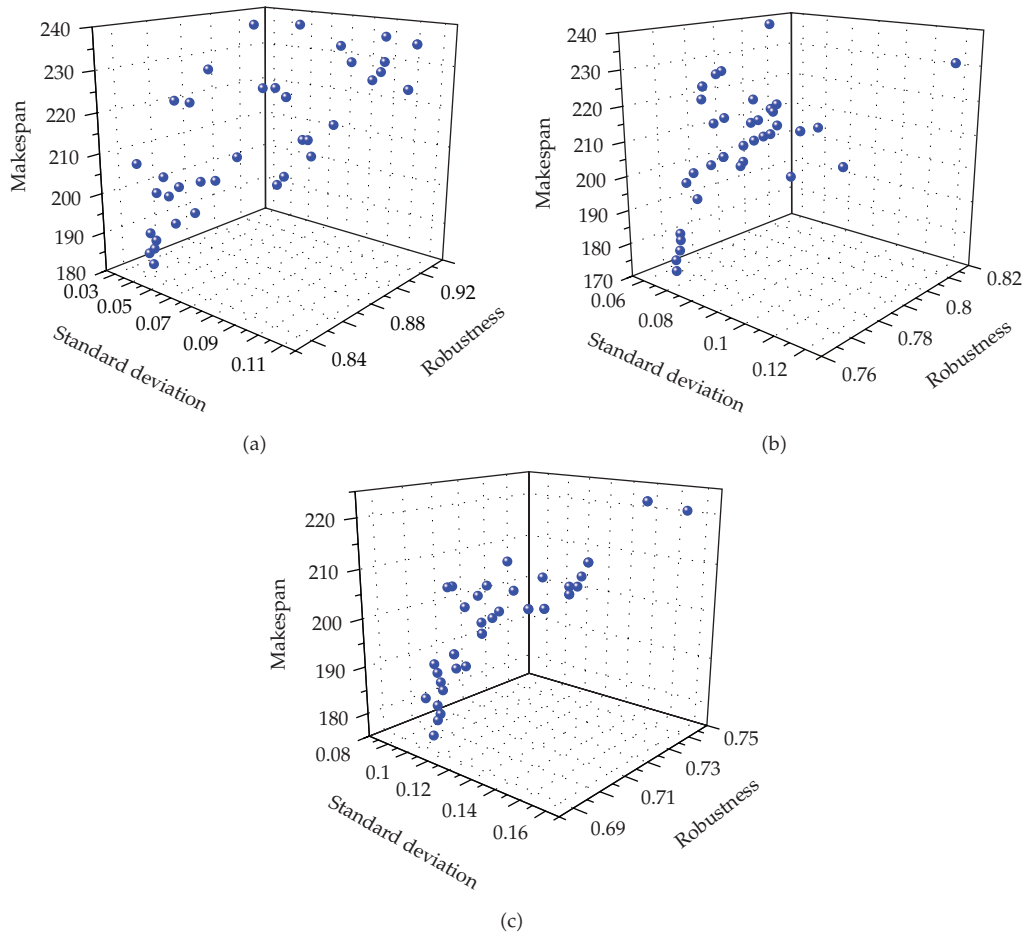

Figure 5: The obtained nondominated solutions in the three-dimensional space with different perturbation strengths: (a) perturbation strength 0.15; (b) perturbation strength 0.25; (c) perturbation strength 0.35.

in Figure 5, where the distribution of the solutions under perturbation strengths 0.15, 0.25, and 0.35 is shown in Figures 5(a), 5(b), and 5(c), respectively. The well-distributed solutions shown in these figures indicate that the proposed three objectives, named makespan, robustness, and stability, are in conflict with each other.

In order to visualize the relation between the makespan and the robustness, we projected the nondominated solutions in the robustness-makespan space (see Figure 6). Figures 6(a), 6(b), and 6(c), respectively, show the projection of the obtained results under perturbation strengths 0.15, 0.25, and 0.35. It can be seen from these figures that some dominated solution were included in the robustness-makespan space. This is because these solution provided a better performance on stability. By looking at the measure of robustness, we can see that, by the increase of the perturbation strength, the robustness value decreased (from Figure 6(a) to Figure 6(c)) as expected.

Another important issue of RCPSP is genotype-phenotype analysis. We investigated two nondominated solutions obtained under the perturbation 0.25. Solution A, $I_A = (0, 1, 6, 3, 13, 2, 7, 4, 16, 5, 9, 10, 12, 19, 8, 14, 15, 20, 17, 11, 18, 21)$, with objective values *makespan* = 210,

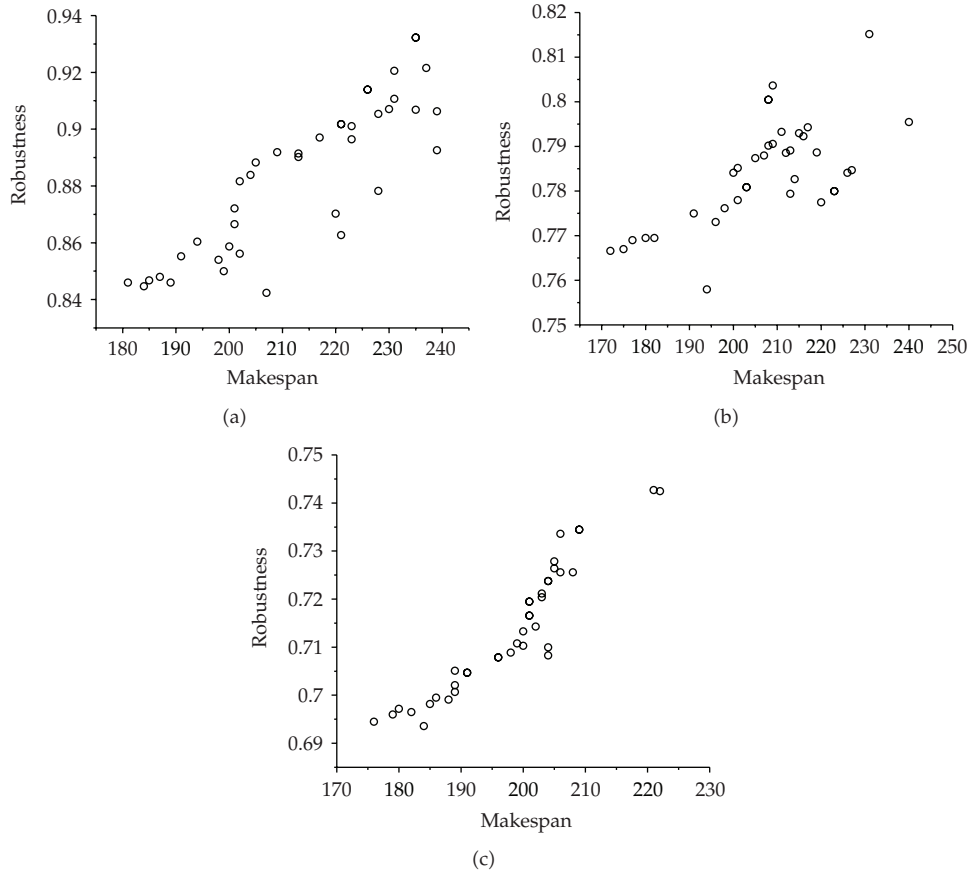

Figure 6: The projection of the nondominated solutions in the robustness-makespan space with different perturbation strengths: (a) perturbation strength 0.15; (b) perturbation strength 0.25; (c) perturbation strength 0.35.

robustness = 0.8951, and *standard deviation* = 0.0859; solution B, $I_B = (0, 3, 1, 6, 4, 16, 13, 2, 5, 7, 9, 12, 8, 14, 11, 10, 19, 17, 15, 18, 20, 21)$, with objective values *makespan= 210*, *robustness* = 0.8444, and *standard deviation* = 0.0342. One can see that although these two solutions had the same makespans, solution A had a better measure on robustness than solution B. Figures 7 and 8, respectively, show the genotype-phenotype mapping of solutions A and B. It can be seen from the figures that for both schedules, since there were many available resources, more than one precedence-feasible activities were allowed to be executed concurrently.

Let us recall that we indicated the robustness of a schedule as the difference between the deterministic makespan before disruption, and the actual makespan after execution. In some literature, such as [10], a surrogate measure—total amount of *free slacks*—was employed to indicate the robustness of a schedule. It seems that a schedule with higher measure of *free slack* sum would cope with the perturbation better. Actually, this situation cannot be held in some cases. For example, by looking at the obtained solutions A and B, it can be seen from Figure 7 that the total amount of the free slacks of schedule A is 4 + 4 + 6 = 14 (denoted as blue blocks in the figure), while the value for schedule B is 3 + 2 + 6 + 10 = 21 (see Figure 8). However, in real execution in the presence of perturbation, solution A, compared to solution

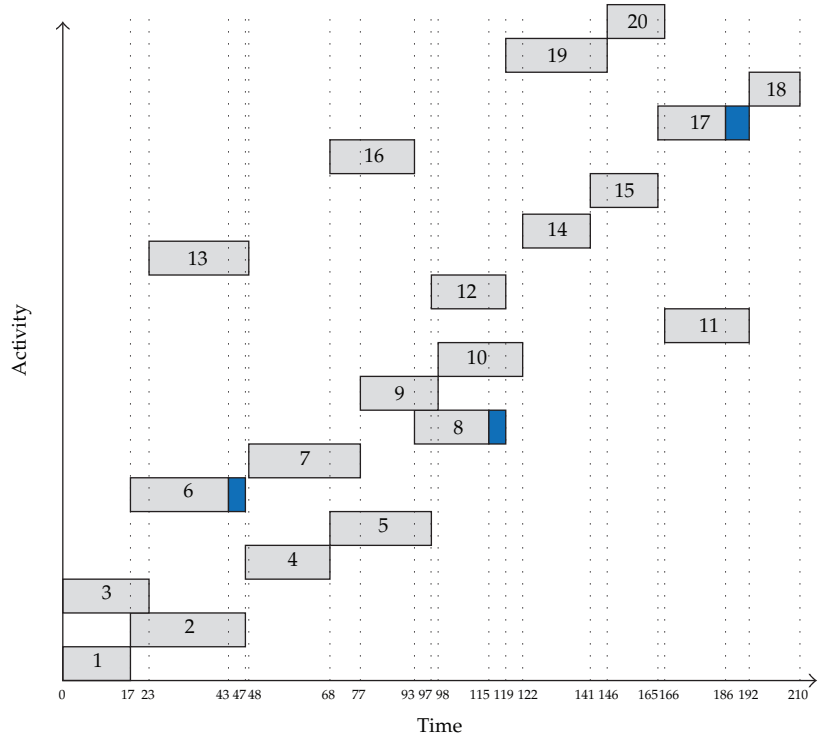

Figure 7: Snapshots of obtained schedule A, with objective values *makespan* = 210, *robustness* = 0.8951, and *standard deviation* = 0.0859.

B, had a better performance of absorbing the uncertainty in the environment, thus, had a lower measure of difference with the planned execution in deterministic environment.

4.4. Performance Analysis

In this section, we evaluated the performance of the proposed H-MOEA. Here we use MOEA to represent the normal MOEA. We employ the measure of *set cover* [40], denoted as SC, to implement the comparison among different approaches. Let A and B be the two obtained nondominated sets, $SC(A, B)$ is defined as the rate of solutions in B which are dominated by the solutions in A. It is clear that a higher value of $SC(A, B)$ indicates a better performance of the set A. Note, however, $SC(A, B)$ is not necessarily equal to $SC(B, A)$.

4.4.1. Test Problem Instances

Besides the hypothetical problem described in the above section (denoted as P0), we tested our approach on the benchmark problems generated with *Progen*, which is a problem generator developed by Kolisch et al. [41]. The *network complexity* (NC) and *resource factor* (RF) are two important parameters to *Progen*. The former parameter indicates the average number of immediate successors of an activity. The latter parameter, ranged between 0 and 1, is used to control the percentage of resource types required by an activity. In the generation of

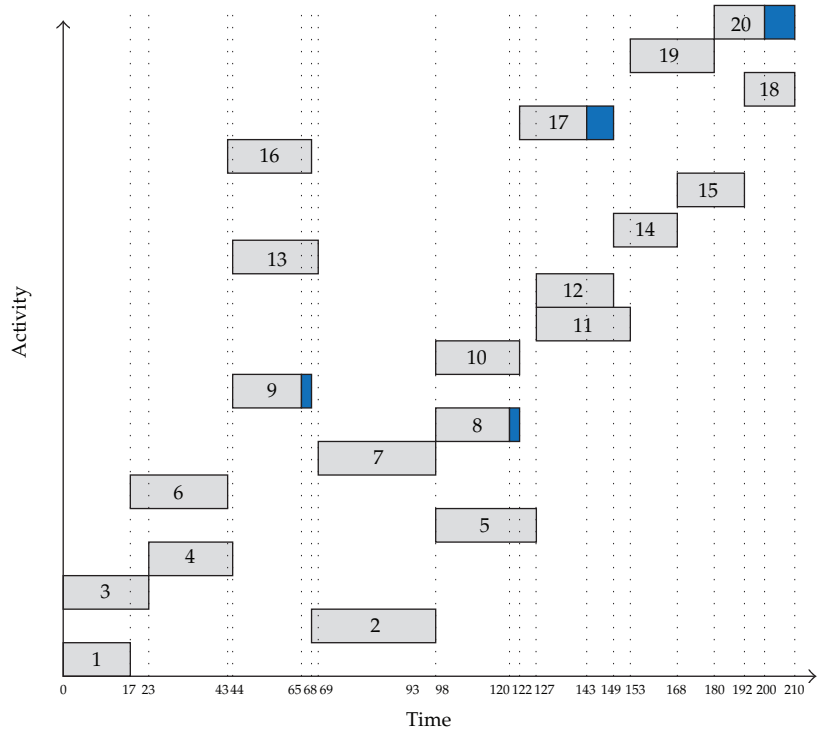

Figure 8: Snapshots of obtained schedule A, with objective values *makespan* = *210*, *robustness* = 0.8444, and *standard deviation* = 0.0342.

Table 4: Parameter settings of the 20 generated instances of benchmark.

Index	N	NC	K	RF	Index	N	NC	K	RF
P1	20	1.8	4	0.75	P11	30	2.5	2	0.75
P2	20	1.8	4	1	P12	30	2.5	2	1
P3	20	2.1	4	0.75	P13	60	1.8	4	0.75
P4	20	2.1	4	1	P14	60	1.8	4	1
P5	20	2.5	2	0.75	P15	60	2.1	4	0.75
P6	20	2.5	2	1	P16	60	2.1	4	1
P7	30	1.8	4	0.75	P17	60	2.5	2	0.75
P8	30	1.8	4	1	P18	60	2.5	2	1
P9	30	2.1	4	0.75	P19	120	1.8	4	0.75
P10	30	2.1	4	1	P20	120	1.8	4	1

the problem instances, we assumed that the durations, the resource availability, and the required resources for each activity were generated between the predefined intervals, satisfying the resource constraint that the maximum of the required resource cannot exceed the resource availability. Table 4 shows the parameter settings of the 20 generated instances.

4.4.2. Performance Comparison under the Same Number of Generations

The algorithm parameters remained the same as in the above section. We ran the algorithm with all of the generated instances under three different perturbation strengths (0.15, 0.25,

Table 5: The mean value and the variance of the *set covers* $SC(\text{H-MOEA}, \text{MOEA})(SC_1)$ and $SC(\text{MOEA}, \text{H-MOEA})(SC_2)$ within the scenarios of perturbation strengths 0.15, 0.25, and 0.35, under the same number of generations.

	St = 0.15		St = 0.25		St = 0.35	
	SC_1	SC_2	SC_1	SC_2	SC_1	SC_2
P0	**0.3158** ± 0.0631	0.1158 ± 0.0151	**0.4375** ± 0.0675	0.0983 ± 0.0093	**0.4550** ± 0.0967	0.0883 ± 0.0076
P1	**0.2492** ± 0.0487	0.1508 ± 0.0176	**0.3900** ± 0.0931	0.1908 ± 0.0367	**0.2942** ± 0.0659	0.1542 ± 0.0270
P2	**0.2933** ± 0.0401	0.1792 ± 0.0245	**0.3167** ± 0.0647	0.1683 ± 0.0327	**0.3483** ± 0.0564	0.2058 ± 0.0427
P3	**0.4383** ± 0.0487	0.2000 ± 0.0200	**0.4667** ± 0.0901	0.0825 ± 0.0124	**0.2992** ± 0.0354	0.1200 ± 0.0150
P4	**0.2942** ± 0.0530	0.1350 ± 0.0245	**0.3400** ± 0.0600	0.1542 ± 0.0221	**0.4517** ± 0.0974	0.1400 ± 0.0264
P5	**0.2892** ± 0.0574	0.1250 ± 0.0145	**0.4283** ± 0.0817	0.1608 ± 0.0274	**0.4267** ± 0.0820	0.1758 ± 0.0416
P6	**0.4700** ± 0.0406	0.1008 ± 0.0145	**0.5475** ± 0.0759	0.1458 ± 0.0256	**0.6250** ± 0.0870	0.0558 ± 0.0068
P7	**0.3417** ± 0.0690	0.1992 ± 0.0461	**0.4250** ± 0.0762	0.1958 ± 0.0579	**0.5083** ± 0.0917	0.1550 ± 0.0479
P8	**0.3517** ± 0.0552	0.2583 ± 0.0653	**0.3542** ± 0.0645	0.2800 ± 0.0534	**0.3608** ± 0.0694	0.2667 ± 0.0302
P9	**0.3858** ± 0.0747	0.2317 ± 0.0354	0.2600 ± 0.0290	0.2592 ± 0.0259	**0.3864** ± 0.0743	0.2218 ± 0.0189
P10	**0.4625** ± 0.0653	0.1442 ± 0.0396	**0.4458** ± 0.0777	0.1733 ± 0.0345	**0.4408** ± 0.0979	0.1442 ± 0.0299
P11	**0.5025** ± 0.0981	0.1650 ± 0.0432	**0.3608** ± 0.0629	0.1500 ± 0.0281	**0.4127** ± 0.0503	0.2967 ± 0.0363
P12	**0.3258** ± 0.0494	0.2258 ± 0.0446	**0.3025** ± 0.0583	0.2650 ± 0.0455	**0.3325** ± 0.0647	0.3033 ± 0.0510
P13	**0.3033** ± 0.0427	0.2367 ± 0.0415	0.2508 ± 0.0441	**0.3242** ± 0.0780	**0.3783** ± 0.0673	0.2433 ± 0.0610
P14	**0.3408** ± 0.0636	0.2567 ± 0.0456	**0.3483** ± 0.0716	0.2992 ± 0.0559	0.2908 ± 0.0562	0.2974 ± 0.0758
P15	**0.3983** ± 0.0803	0.2492 ± 0.0481	**0.3358** ± 0.0814	0.2325 ± 0.0663	**0.3175** ± 0.0612	0.2608 ± 0.0549
P16	**0.4392** ± 0.0781	0.2383 ± 0.0535	**0.3292** ± 0.0627	0.2708 ± 0.0398	**0.3667** ± 0.0927	0.3017 ± 0.0765
P17	0.3017 ± 0.0693	**0.3992** ± 0.0727	**0.3500** ± 0.0680	0.3225 ± 0.0727	**0.4075** ± 0.0974	0.2642 ± 0.0780
P18	**0.3400** ± 0.0475	0.2833 ± 0.0447	0.3508 ± 0.0834	0.3433 ± 0.0700	**0.3792** ± 0.0598	0.2683 ± 0.0440
P19	**0.3842** ± 0.0528	0.2817 ± 0.0720	**0.3983** ± 0.0912	0.2375 ± 0.0694	**0.3375** ± 0.0804	0.2533 ± 0.0740
P20	**0.3858** ± 0.0649	0.1375 ± 0.0271	**0.3508** ± 0.1001	0.2392 ± 0.0720	**0.3075** ± 0.0516	0.2358 ± 0.0592

and 0.35). First, we compared the performances of the proposed H-MOEA with normal MOEA under the same number of generations, that is, both algorithms were terminated after 200 generations. The results are reported in the Table 5, where SC_1 represents the *set cover* $SC(\text{H-MOEA, MOEA})$ and SC_2 denotes the *set cover* $SC(\text{MOEA, H-MOEA})$.

In Table 5, the mean value of the *set cover* which is better than its reverse figure is presented in bold face. From the figures, one can clearly see that the H-MOEA considerably outperformed the normal one in most cases. For the instances with 20 (P0–P6) and 30 (P7–P12) nondummy activities, H-MOEA had a significant better performance than MOEA in all scenarios with three different perturbations. For example, if we look at the experimental results of instance P1, the mean values of $SC1$ under the perturbations with strengths 0.15, 0.25, and 0.35 are 0.2492, 0.3900, and 0.2942, respectively, while the reverse figures ($SC2$ under the perturbations with strengths 0.15, 0.25, and 0.35) are only 0.1508, 0.1908, and 0.1542, respectively. The same set of figures for P6 are 0.4700, 0.5475, 0.6250 and 0.1008, 0.1458, 0.0558, respectively. For the instances with 60 nondummy activities, H-MOEA performed better than normal MOEA in most cases. However, one can see that for the instances P13 under perturbation strength 0.25 and P17 under perturbation strength 0.15, MOEA outperformed H-MOEA, with the mean values of *set cover* SC_2 0.3242 and 0.3992, respectively, and the reverse figures are 0.2508 and 0.3017. Also, for the instances P14 under perturbation strength 0.35 and P18 under perturbation strength 0.25, H-MOEA and MOEA approximately had the same

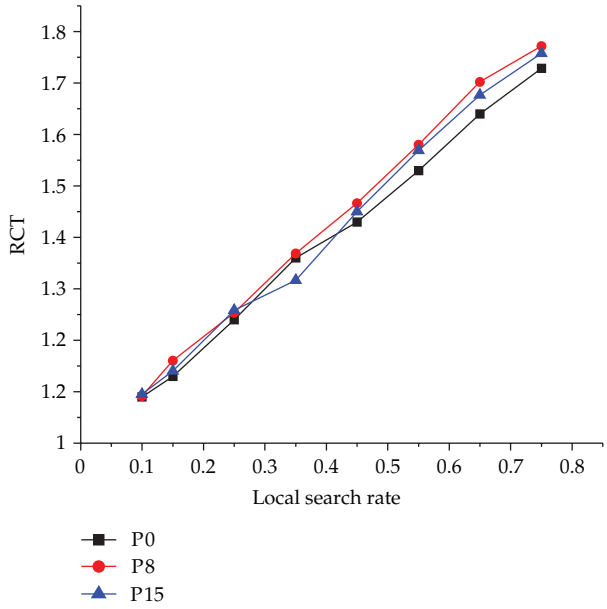

Figure 9: The obtained relative computation time (RCT) with different local search rates for P0, P8, and P15 with a perturbation strength 0.15.

performance. Finally, we investigated two instances with 120 nondummy activities. The experimental results also suggest that H-MOEA performed better than MOEA under deferent perturbation strength levels.

4.4.3. Remarks on H-MOEA

In the proposed H-MOEA, the information contained in the obtained approximating nondominated solutions were extracted and utilized periodically. The experimental results discussed above show that H-MOEA has a considerable better performance than normal MOEA. This indicates that the heuristic information accumulated in the previous runs is very useful to guide the subsequent search process. Since we employed a local search procedure in the process of MOEA, it was unavoidable that the computation time of the proposed H-MOEA increased to a certain degree. In other words, the improvement of convergence performance was with the sacrifice of computation time. Thus, it is an important issue to investigate the relation between the local search rate and the performance of H-MOEA. Here, we employ two measures, relative set cover *RSC* and relative computation time *RCT*, to indicate the performance of the proposed H-MOEA. *RSC* and *RCT* are calculated as follows:

$$RSC = \frac{SC(\text{H-MOEA}, \text{MOEA})}{SC(\text{MOEA}, \text{H-MOEA})}, \tag{4.1}$$

$$RCT = \frac{CT_{\text{H-MOEA}}}{CT_{\text{MOEA}}}, \tag{4.2}$$

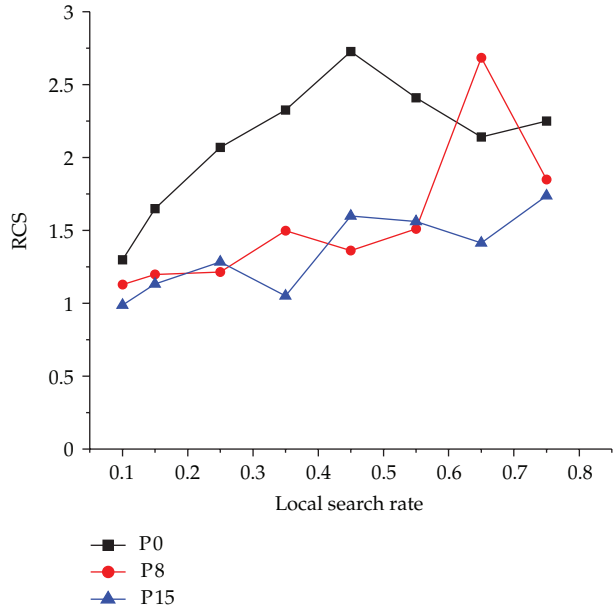

Figure 10: The obtained relative set cover (RSC) with different local search rates for P0, P8, and P15 with a perturbation strength 0.15.

where $CT_{\text{H-MOEA}}$ denotes the computation time of the proposed H-MOEA and CT_{MOEA} represents the computation time of the normal MOEA.

We tested the effect of local search rate p_{ls} on the proposed H-MOEA by varying the local search rate from 0.1 to 0.75 on the problems P0 (20 nondummy activities), P8 (30 nondummy activities), and P15 (60 nondummy activities) under perturbation strength 0.15. Figure 9 shows the values of relative computation time *RCT* with variation of local search rate p_{ls}. The values of relative set cover *RSC* with variation of p_{ls} are reported in Figure 10. All the values in the figures are the mean values of the obtained results after 30 repeated runs. It can be clearly seen, from Figure 9, that, by the increase of local search rate, more computation time was needed to perform the H-MOEA.

The definition of relative set cover *RSC* in (4.1) shows that a value greater than 1 indicates a better performance for H-MOEA. By looking at Figure 10, it can be seen that even a lower local search rate (0.1) could yield a better performance for the instances with 20 nondummy activities (P0) and 30 nondummy activities (P8), with *RSC* values about 1.30 and 1.13, respectively. It is interesting to notice that the values of relative set cover *RSC* did not improve linearly with the increase of local search rate (see Figure 10). In some cases, for example, for instance with 20 nondummy activities (P0), the highest value of *RSC* did not correspond to the highest local search rate. One of the reasons for this is that a higher local search rate might limit the exploration ability of the algorithm in the search space. Thus, with another consideration, computation time, the local search rate should not be too high. In this research, we commonly consider that a local search rate between 0.3 and 0.5 is suitable to the proposed algorithm and the problem domain. In real application on other complex problems, a tradeoff between convergence performance and computational time is possibly needed.

Table 6: The mean value and the variance of the *set covers* $SC(\text{H-MOEA}, \text{MOEA})(SC_1)$ and $SC(\text{MOEA}, \text{H-MOEA})(SC_2)$ within the scenarios of perturbation strengths 0.15, 0.25, and 0.35, under the same examined solutions.

	St = 0.15		St = 0.25		St = 0.35	
	SC_1	SC_2	SC_1	SC_2	SC_1	SC_2
P0	**0.3692** ± 0.0939	0.0983 ± 0.0083	**0.4425** ± 0.0900	0.1108 ± 0.0147	**0.4117** ± 0.0756	0.1017 ± 0.0136
P1	**0.3275** ± 0.0559	0.1267 ± 0.0239	**0.3992** ± 0.0966	0.1425 ± 0.0242	**0.2800** ± 0.0536	0.1842 ± 0.0313
P2	**0.2792** ± 0.0380	0.1458 ± 0.0170	**0.3025** ± 0.0491	0.1633 ± 0.0219	**0.3233** ± 0.0476	0.2242 ± 0.0446
P3	**0.3392** ± 0.0414	0.2383 ± 0.0287	**0.4067** ± 0.0883	0.1392 ± 0.0253	**0.3392** ± 0.0726	0.1175 ± 0.0207
P4	**0.2833** ± 0.0469	0.1883 ± 0.0423	**0.2775** ± 0.0335	0.1925 ± 0.0317	**0.4492** ± 0.1074	0.2408 ± 0.0659
P5	**0.3100** ± 0.0659	0.1492 ± 0.0205	**0.4158** ± 0.0748	0.1908 ± 0.0307	**0.4150** ± 0.0542	0.1583 ± 0.0273
P6	**0.4433** ± 0.0582	0.1292 ± 0.0293	**0.5600** ± 0.1226	0.2292 ± 0.0635	**0.4800** ± 0.0732	0.1175 ± 0.0188
P7	**0.3417** ± 0.0690	0.1992 ± 0.0461	**0.4250** ± 0.0762	0.1958 ± 0.0579	**0.5083** ± 0.0917	0.1550 ± 0.0479
P8	0.3075 ± 0.0526	0.2942 ± 0.0487	0.3208 ± 0.0777	0.3333 ± 0.0774	0.3075 ± 0.0579	0.3117 ± 0.0492
P9	**0.2808** ± 0.0428	0.2617 ± 0.0322	**0.3433** ± 0.0576	0.2408 ± 0.0335	**0.3627** ± 0.0348	0.2769 ± 0.0198
P10	**0.3117** ± 0.0741	0.2233 ± 0.0299	**0.3758** ± 0.0501	0.1775 ± 0.0228	**0.3483** ± 0.0585	0.2092 ± 0.0564
P11	**0.5037** ± 0.1087	0.1650 ± 0.0462	**0.3642** ± 0.0524	0.1558 ± 0.0297	**0.3128** ± 0.0265	0.2364 ± 0.0127
P12	0.2883 ± 0.0597	0.2983 ± 0.0609	**0.3167** ± 0.0619	0.2700 ± 0.0447	0.3542 ± 0.0540	0.3500 ± 0.0780
P13	0.2458 ± 0.0523	**0.3183** ± 0.0746	0.2475 ± 0.0348	**0.3458** ± 0.0845	**0.3733** ± 0.0895	0.2883 ± 0.0681
P14	**0.3642** ± 0.0791	0.2525 ± 0.0534	0.3175 ± 0.0826	0.3242 ± 0.0632	**0.3726** ± 0.0635	0.3214 ± 0.0468
P15	**0.3558** ± 0.0946	0.3033 ± 0.0619	0.3162 ± 0.0486	0.3210 ± 0.0331	**0.4058** ± 0.0596	0.3374 ± 0.0288
P16	**0.3225** ± 0.0503	0.2783 ± 0.0455	**0.3433** ± 0.0880	0.2792 ± 0.0629	0.2700 ± 0.0480	**0.3750** ± 0.0840
P17	0.2717 ± 0.0574	**0.3667** ± 0.0661	**0.3292** ± 0.0575	0.3033 ± 0.0488	0.2833 ± 0.0654	**0.3767** ± 0.0676
P18	**0.3483** ± 0.0707	0.2417 ± 0.0522	0.3150 ± 0.0674	**0.3925** ± 0.0687	**0.3842** ± 0.0694	0.2258 ± 0.0473
P19	0.3458 ± 0.0727	0.3575 ± 0.0790	**0.3650** ± 0.0779	0.3142 ± 0.0572	0.3150 ± 0.0609	0.3125 ± 0.0863
P20	**0.3254** ± 0.0649	0.2384 ± 0.0341	**0.2704** ± 0.0801	0.2137 ± 0.0520	**0.2674** ± 0.0613	0.2059 ± 0.0462

4.4.4. Performance Comparison under the Same Computational Effort

As indicated above, due to local search at each generation, much more solutions are examined in proposed H-MOEA using more computation time. In order to further investigate the performance of proposed H-MOEA, we, respectively, compared the performances of H-MOEA and MOEA under the same number of examined solutions and computational time. To do this, we executed the algorithms as follows: at first, we ran the MOEA with 200 generations and recorded the number of examined solutions in MOEA and its computational time. Then, the proposed H-MOEA was executed with the following two termination conditions: (1) the number of examined solutions was achieved; (2) the computational time was achieved.

Table 6 shows the computational results obtained under the same examined solutions, while the results under the same computation time are reported in Table 7. After careful examination of the data, it could be found that the proposed H-MOEA outperformed normal MOEA on small size instances, while the performance of H-MOEA decreased on relative large scale cases. By looking at the instances P_0–P_{12}, the data suggests H-MOEA performed better than normal MOEA in most instances, under both the same examined solutions (see Table 6) and the same computation time (see Table 7). For relative large scale instances (P_{13}–P_{20}), shown as in Table 6, H-MOEA still outperformed MOEA in most cases under the same

Table 7: The mean value and the variance of the *set covers* SC(H-MOEA, MOEA) (SC_1) and SC(MOEA, H-MOEA)(SC_2) within the scenarios of perturbation strengths 0.15, 0.25, and 0.35, under the same computational time.

	St = 0.15		St = 0.25		St = 0.35	
	SC_1	SC_2	SC_1	SC_2	SC_1	SC_2
P0	**0.2800** ± 0.0551	0.1367 ± 0.0095	**0.3800** ± 0.0428	0.1083 ± 0.0082	**0.4775** ± 0.0733	0.0892 ± 0.0117
P1	**0.3025** ± 0.0684	0.1633 ± 0.0288	**0.3858** ± 0.0845	0.0917 ± 0.0115	0.2250 ± 0.0407	**0.2517** ± 0.0365
P2	0.2075 ± 0.0128	**0.2200** ± 0.0163	**0.2850** ± 0.0462	0.1833 ± 0.0213	**0.3233** ± 0.0424	0.2217 ± 0.0280
P3	**0.4133** ± 0.0786	0.2025 ± 0.0285	**0.3850** ± 0.0682	0.1067 ± 0.0130	**0.3108** ± 0.0427	0.1233 ± 0.0069
P4	**0.2617** ± 0.0501	0.1917 ± 0.0290	**0.3183** ± 0.0471	0.1842 ± 0.0305	**0.3575** ± 0.0616	0.2092 ± 0.0606
P5	**0.2125** ± 0.0161	0.1525 ± 0.0186	**0.3650** ± 0.0691	0.1942 ± 0.0406	**0.3833** ± 0.0867	0.1975 ± 0.0425
P6	**0.4475** ± 0.0746	0.1283 ± 0.0362	**0.5933** ± 0.0746	0.1475 ± 0.0323	**0.4800** ± 0.0744	0.1417 ± 0.0282
P7	**0.3417** ± 0.0690	0.1992 ± 0.0461	**0.4250** ± 0.0762	0.1958 ± 0.0579	**0.5083** ± 0.0917	0.1550 ± 0.0479
P8	0.2983 ± 0.0645	0.2808 ± 0.0599	**0.4275** ± 0.0694	0.2208 ± 0.0425	0.3133 ± 0.0445	0.3250 ± 0.0488
P9	0.2883 ± 0.0486	0.2967 ± 0.0468	**0.2833** ± 0.0415	0.2308 ± 0.0179	**0.3015** ± 0.0312	0.2453 ± 0.0203
P10	**0.3524** ± 0.0452	0.29432 ± 0.0298	**0.3453** ± 0.0674	0.30343 ± 0.0446	**0.4018** ± 0.0906	0.3472 ± 0.0439
P11	**0.4725** ± 0.1044	0.2175 ± 0.0618	**0.4450** ± 0.0840	0.1717 ± 0.0308	**0.4531** ± 0.0368	0.2741 ± 0.0153
P12	**0.3008** ± 0.0489	0.2617 ± 0.0555	0.3000 ± 0.0372	0.3025 ± 0.0575	0.3383 ± 0.0674	0.3433 ± 0.0684
P13	0.2158 ± 0.0472	**0.4250** ± 0.0594	0.2267 ± 0.0400	**0.3825** ± 0.0651	0.3117 ± 0.0579	0.3158 ± 0.0816
P14	**0.3967** ± 0.0742	0.2583 ± 0.0466	0.3458 ± 0.0699	0.3513 ± 0.0992	**0.3369** ± 0.0264	0.2867 ± 0.0224
P15	**0.3350** ± 0.0488	0.2750 ± 0.0459	0.3452 ± 0.0431	0.3501 ± 0.0523	0.3940 ± 0.0511	0.4012 ± 0.0634
P16	0.3417 ± 0.0865	0.3400 ± 0.0971	**0.3233** ± 0.0525	0.2692 ± 0.0623	0.3267 ± 0.0858	0.3350 ± 0.0696
P17	0.2875 ± 0.0656	0.2904 ± 0.0557	0.2575 ± 0.0650	0.2467 ± 0.0655	0.3183 ± 0.0588	0.3258 ± 0.0854
P18	0.3342 ± 0.0610	0.3450 ± 0.0775	0.2575 ± 0.0638	0.2592 ± 0.0608	0.2908 ± 0.0574	0.3000 ± 0.0497
P19	0.2883 ± 0.0451	**0.3267** ± 0.0745	0.2683 ± 0.0712	**0.3458** ± 0.0786	0.2567 ± 0.0557	**0.4400** ± 0.0887
P20	0.2978 ± 0.0447	0.3004 ± 0.0631	**0.2503** ± 0.0601	0.2294 ± 0.0523	0.3772 ± 0.0719	0.3685 ± 0.0793

examined solutions. While under the same computation time, the performances of H-MOEA and MOEA were comparable on instances P_{13}–P_{20}, suggested as in Table 7. The main reason of performance decrease of H-MOEA on large scale instances could be considered as limitation on global search ability of the algorithm. As indicated in [37], the number of generations is decreased when the available computation time is limited. As a result, the exploration in search space is not fully utilized.

5. Conclusion

In this paper, we study a resource-constrained project scheduling problem in the presence of perturbation on activity durations. *Robust scheduling* is employed as the methodology to solve this problem. We measure the robustness of a schedule as the expectation of the difference between the actual makespan after execution and the planned makespan. The stability of the schedule under the perturbation is indicated by the statistical measure of standard deviation. Then, the problem is modelled as a multiobjective optimization problem, where makespan minimization, robustness maximization, and stability maximization (standard deviation minimization) are considered simultaneously.

We employ multiobjective evolutionary algorithm (MOEA) to obtain the Pareto optimal solutions of the problem. The normal MOEA is improved by incorporating a knowledge-based local search procedure. We call the improved MOEA as Hybrid MOEA (H-MOEA). In the process of the proposed H-MOEA, two additional procedures are integrated: *information extraction* and *information utilization*. In the former procedure, heuristic information is obtained periodically from the obtained approximating nondominated individuals. In the latter procedure, the obtained information is utilized in the local search to improve the individuals in the population. Experimental study shows that the proposed approach is feasible and effective for the resource-constrained project scheduling problem with stochastic durations and that the proposed H-MOEA performs better than normal MOEA. In future work, we plan to employ the hybrid multiobjective evolutionary algorithm to investigate the project scheduling problem under dynamic environment. It is also worthwhile to focus on the design of more efficient hybrid MOEAs for solving large-scale problems in future work.

Acknowledgments

This research was supported in part by the China Scholarship Council and National Natural Science Foundation of China under the Contract nos. 70971131, 70901074, 71001104, and 71101150. The authors would like to thank the anonymous reviewers for their invaluable comments and constructive suggestions to improve this paper. Also, the authors wish to thank the editor-in-chief and editor(s) for their diligent working on this paper. The first author would like to thank Dr. Hugo Fouchal at the Department of Computer Science, University of Nantes, France, for his suggestions to revise this paper.

References

[1] R. H. Moehring, "Minimizing costs of resource requirement in project networks subject to a fixed completion," *Operations Research*, vol. 32, no. 1, pp. 89–120, 1984.

[2] D. Merkle, M. Middendorf, and H. Schmeck, "Ant colony optimization for resource-constrained project scheduling," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 4, pp. 333–346, 2002.

[3] W. Herroelen, B. De Reyck, and E. Demeulemeester, "Resource-constrained project scheduling: a survey of recent developments," *Computers and Operations Research*, vol. 25, no. 4, pp. 279–302, 1998.

[4] P. Brucker, A. Drexl, R. Möhring, K. Neumann, and E. Pesch, "Resource-constrained project scheduling: notation, classification, models, and methods," *European Journal of Operational Research*, vol. 112, no. 1, pp. 3–41, 1999.

[5] R. Kolisch and R. Padman, "An integrated survey of deterministic project scheduling," *Omega*, vol. 29, no. 3, pp. 249–272, 2001.

[6] R. Kolisch and S. Hartmann, "Experimental investigation of heuristics for resource-constrained project scheduling: an update," *European Journal of Operational Research*, vol. 174, no. 1, pp. 23–37, 2006.

[7] S. Hartmann and D. Briskorn, "A survey of variants and extensions of the resource-constrained project scheduling problem," *European Journal of Operational Research*, vol. 207, no. 1, pp. 1–14, 2010.

[8] S. Van De Vonder, E. Demeulemeester, W. Herroelen, and R. Leus, "The use of buffers in project management: the trade-off between stability and makespan," *International Journal of Production Economics*, vol. 97, no. 2, pp. 227–240, 2005.

[9] W. Herroelen and R. Leus, "Project scheduling under uncertainty: survey and research potentials," *European Journal of Operational Research*, vol. 165, no. 2, pp. 289–306, 2005.

[10] M. A. Al-Fawzan and M. Haouari, "A bi-objective model for robust resource-constrained project scheduling," *International Journal of Production Economics*, vol. 96, no. 2, pp. 175–187, 2005.

[11] B. Abbasi, S. Shadrokh, and J. Arkat, "Bi-objective resource-constrained project scheduling with robustness and makespan criteria," *Applied Mathematics and Computation*, vol. 180, no. 1, pp. 146–152, 2006.

[12] H. Chtourou and M. Haouari, "A two-stage-priority-rule-based algorithm for robust resource-constrained project scheduling," *Computers and Industrial Engineering*, vol. 55, no. 1, pp. 183–194, 2008.
[13] O. Lambrechts, E. Demeulemeester, and W. Herroelen, "A tabu search procedure for developing robust predictive project schedules," *International Journal of Production Economics*, vol. 111, no. 2, pp. 493–508, 2008.
[14] F. Ballestín and R. Leus, "Resource-constrained project scheduling for timely project completion with stochastic activity durations," *Production and Operations Management*, vol. 18, no. 4, pp. 459–474, 2009.
[15] B. Ashtiani, R. Leus, and M. B. Aryanezhad, "New competitive results for the stochastic resource-constrained project scheduling problem: exploring the benefits of pre-processing," *Journal of Scheduling*, pp. 1–15, 2009.
[16] V. Jorge Leon, S. David Wu, and R. H. Storer, "Robustness measures and robust scheduling for job shops," *IIE Transactions*, vol. 26, no. 5, pp. 32–43, 1994.
[17] Y. Jin and J. Branke, "Evolutionary optimization in uncertain environments—a survey," *IEEE Transactions on Evolutionary Computation*, vol. 9, no. 3, pp. 303–317, 2005.
[18] D. S. Lee, L. F. Gonzalez, J. Périaux, and K. Srinivas, "Efficient hybrid-game strategies coupled to evolutionary algorithms for robust multidisciplinary design optimization in aerospace engineering," *IEEE Transactions on Evolutionary Computation*, vol. 15, no. 2, pp. 133–150, 2011.
[19] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 2, pp. 182–197, 2002.
[20] S. Hartmann, "A competitive genetic algorithm for resource-constrained project scheduling," *Naval Research Logistics*, vol. 45, no. 7, pp. 733–750, 1998.
[21] S. Shadrokh and F. Kianfar, "A genetic algorithm for resource investment project scheduling problem, tardiness permitted with penalty," *European Journal of Operational Research*, vol. 181, no. 1, pp. 86–101, 2007.
[22] T. Murata, H. Ishibuchi, and H. Tanaka, "Genetic algorithms for flowshop scheduling problems," *Computers and Industrial Engineering*, vol. 30, no. 4, pp. 1061–1071, 1996.
[23] T. Murata, H. Ishibuchi, and H. Tanaka, "Multi-objective genetic algorithm and its applications to flowshop scheduling," *Computers and Industrial Engineering*, vol. 30, no. 4, pp. 957–968, 1996.
[24] C. A. Coello Coello, "Evolutionary multi-objective optimization: a historical view of the field," *IEEE Computational Intelligence Magazine*, vol. 1, no. 1, pp. 28–36, 2006.
[25] H. Ishibuchi and T. Murata, "Multi-objective genetic local search algorithm," in *Proceedings of the IEEE International Conference on Evolutionary Computation*, pp. 119–124, May 1996.
[26] H. Ishibuchi and T. Murata, "A multi-objective genetic local search algorithm and its application to flowshop scheduling," *IEEE Transactions on Systems, Man and Cybernetics Part C*, vol. 28, no. 3, pp. 392–403, 1998.
[27] A. Jaszkiewicz, "Genetic local search for multi-objective combinatorial optimization," *European Journal of Operational Research*, vol. 137, no. 1, pp. 50–71, 2002.
[28] A. Jaszkiewicz, "On the performance of multiple-objective genetic local search on the 0/1 knapsack problem-A comparative experiment," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 4, pp. 402–412, 2002.
[29] L. N. Xing, Y. W. Chen, K. W. Yang, F. Hou, X. S. Shen, and H. P. Cai, "A hybrid approach combining an improved genetic algorithm and optimization strategies for the asymmetric traveling salesman problem," *Engineering Applications of Artificial Intelligence*, vol. 21, no. 8, pp. 1370–1380, 2008.
[30] L. Xing, P. Rohlfshagen, Y. Chen, and X. Yao, "An evolutionary approach to the multidepot capacitated Arc routing problem," *IEEE Transactions on Evolutionary Computation*, vol. 14, no. 3, Article ID 5352249, pp. 356–374, 2010.
[31] L. N. Xing, Y. W. Chen, P. Wang, Q. S. Zhao, and J. Xiong, "A knowledge-based ant colony optimization for flexible job shop scheduling problems," *Applied Soft Computing Journal*, vol. 10, no. 3, pp. 888–896, 2010.
[32] L. N. Xing, P. Rohlfshagen, Y. W. Chen, and X. Yao, "A hybrid ant colony optimization algorithm for the extended capacitated arc routing problem," *IEEE Transactions on Systems, Man, and Cybernetics Part B*, vol. 41, no. 4, pp. 1110–1123, 2011.
[33] J. D. Knowles and D. W. Corne, "M-PAES: a memetic algorithm for multiobjective optimization," in *Proceedings of the Congress on Evolutionary Computation (CEC '00)*, pp. 325–332, July 2000.
[34] V. Valls, S. Quintanilla, and F. Ballestín, "Resource-constrained project scheduling: a critical activity reordering heuristic," *European Journal of Operational Research*, vol. 149, no. 2, pp. 282–301, 2003.
[35] V. Valls, F. Ballestín, and S. Quintanilla, "A population-based approach to the resource-constrained project scheduling problem," *Annals of Operations Research*, vol. 131, pp. 305–324, 2004.

[36] V. Valls, F. Ballestín, and S. Quintanilla, "A hybrid genetic algorithm for the resource-constrained project scheduling problem," *European Journal of Operational Research*, vol. 185, no. 2, pp. 495–508, 2008.

[37] H. Ishibuchi, T. Yoshida, and T. Murata, "Balance between genetic search and local search in memetic algorithms for multiobjective permutation flowshop scheduling," *IEEE Transactions on Evolutionary Computation*, vol. 7, no. 2, pp. 204–223, 2003.

[38] J.-Y. Lin and Y.-P. Chen, "Analysis on the collaboration between global search and local search in memetic computation," *IEEE Transactions on Evolutionary Computation*, vol. 15, no. 5, pp. 608–623, 2011.

[39] H. Ishibuchi, Y. Hitotsuyanagi, Y. Wakamatsu, and Y. Nojima, "How to choose solutions for local search in multiobjective combinatorial memetic algorithms," in *Proceedings of the 11th International Conference on Parallel Problem Solving from Nature (PPSN '10)*, pp. 516–525, 2010.

[40] K. Deb, *Multi-Objective Optimization Using Evolutionary Algorithms*, John Wiley & Sons, New York, NY, USA, 2001.

[41] R. Kolisch, A. Sprecher, and A. Drexl, "Characterization and generation of a general class of resource-constrained project scheduling problem," *Management Science*, vol. 41, no. 10, pp. 1693–1703, 1995.