RAIRO-Oper. Res. 58 (2024) 4589–4605
https://doi.org/10.1051/ro/2024173

RAIRO Operations Research
www.rairo-ro.org

A JOB SCHEDULING AND REJECTION PROBLEM CONSIDERING SELF-CONTAINED AND CROSS-FUNCTIONAL JOBS

Yu-Chuan Chen[1] and Jen-Ya Wang[2,*]

Abstract. In today's large projects and complex assembly lines, a single multi-skilled worker often needs to complete jobs requiring multiple areas of expertise. Even if a worker possesses all necessary skills, their proficiency can vary. This variability makes it challenging to assess a job's cost-performance ratio before assignment. Larger problem sizes often involve many such jobs that need scheduling or rejection. Clearly, the processing times and workers in the presented problem are more complex than traditional scheduling problems with single-valued processing times and single-functional machines. Two important observations serve as the motivation. First, traditional genetic algorithms with fixed-length chromosomes may not effectively handle the complexity of self-contained and cross-functional jobs and multi-skilled workers. Second, traditional genetic algorithms cannot guarantee a certain level of solution quality. Motivated by these observations, a novel genetic algorithm is developed. This algorithm can quickly search the solution space using an outbreeding technique. Additionally, an upper bound is provided to ensure solution quality. Experimental results demonstrate that the proposed genetic algorithm is superior to others through comprehensive comparisons.

Mathematics Subject Classification. 68W50, 68M20, 68W25, 90B40.

Received April 10, 2024. Accepted August 30, 2024.

1. Introduction

Today, a self-contained job that draws upon multiple areas of expertise can be referred to as a multidisciplinary or cross-functional job. For example, in [42, 43], the completion of a video game requires multiple types of expertise such as storyboarding, 3D figure modeling, scene design, sound effects, programming, testing, and so on. Another example is a seru assembly line [29, 49]. A multi-skilled worker needs to formulate, modify, disassemble, or reformulate multiple jobs one by one, working alone. Due to contract confidentiality clauses or agile development requirements, such a cross-functional job cannot be split into multiple pieces and assigned to multiple workers, maintaining its atomicity.

However, these workers cannot always complete all incoming jobs before a deadline, so some must be rejected. Thevenin *et al.* [45] considered a single-machine scheduling problem, aiming to balance the tardiness penalty and the rejection cost. Mosheiov *et al.* [31] considered earliness and tardiness of jobs on multiple machines. Moreover, job rejection is also a way to prevent heavy tardiness penalties. Since the capability of each machine

Keywords. Job scheduling, parallel machine scheduling, maximization problem, genetic algorithm, biodiversity.

[1] Department of Intelligent Technology and Application, Hungkuang University, Taichung, Taiwan, R.O.C.

[2] Department of Information Management, National Taichung University of Science and Technology, No. 129, Section 3, Sanmin Road, North District, Taichung 404336, Taiwan, R.O.C.

*Corresponding author: jywang@nutc.edu.tw

and the due date of each job are known, it is evident that not all incoming orders can be accepted. Consequently, some trade-offs must be made in advance, leading to job rejection.

In general, such a time-consuming scheduling problem requires effective metaheuristics. Note that a total of 50 jobs results in 50! ($\cong 10^{64}$) possible schedules, making the pursuit of optimality infeasible in the real world. For instance, the scheduling problems in [31,45] are NP-hard and their proposed metaheuristics can generate nearly optimal solutions for their real-world problems. Due to the property of job rejection, both scheduling problems are also partition problems. If effective metaheuristics are employed to reject jobs with low cost performance (CP) ratios as early as possible, the original problem will be partitioned into two parts: accepted and rejected jobs. Consequently, it is sufficient to schedule only the remaining accepted jobs, thereby downsizing a large problem instance.

The scheduling problem presented here differs from the original 0–1 knapsack problem [12], making it challenging to divide jobs into two categories: accepted and rejected. Although both are partition problems, certain differences deter researchers from applying traditional metaheuristic algorithms to this problem. For the typical 0–1 knapsack problem, the value and weight of each object are known and decisionmakers can easily determine which ones have high CP ratios. However, in this problem, the CP ratios of a job vary from worker to worker. Consider the following example: Bob is a baker and Carl is a Chinese chef. Their time consumption ratios for bakery and Chinese food are $r_{\mathrm{B}} = (1, 3)$ and $r_{\mathrm{C}} = (2, 1)$, respectively. There are one unit job of bakery and one unit job of Chinese food. The bakery job yields a profit of \$2, while the Chinese food job yields \$3. If the bakery job is assigned to Bob, a profit of \$2 is earned within one time unit; if assigned to Carl, the same profit is earned within two time units. Similarly, if the Chinese food job is assigned to Carl, a profit of \$3 is earned within one time unit; if assigned to Bob, the same profit is earned within three time units. Clearly, determining the CP ratios in advance for an undetermined job is challenging, especially for interdisciplinary or cross-functional jobs. Therefore, new metaheuristics are essential for effective job evaluation and rejection.

By developing more effective metaheuristic algorithms, unnecessary computations can be minimized. Traditional genetic algorithms often evolve with fixed crossover and mutation rates or unchanged operations. However, in the presented problem, jobs with low CP ratios should be pruned early; *i.e.*, rejected jobs should be excluded from the evolution process whenever possible. Thus, this study aims to develop metaheuristics that adapt by rejecting jobs during the evolution process.

This study introduces several contributions to the field of job scheduling. The primary novelty lies in the development of a genetic algorithm that dynamically adjusts the lengths of its chromosomes during the evolution process, specifically designed to handle complex jobs requiring multiple areas of expertise. The algorithm's efficiency and solution quality are enhanced by its ability to discard unnecessary genes and utilize an outbreeding technique. Additionally, the study provides a mathematical upper bound to benchmark each algorithm's performance. Despite these advancements, the study has certain limitations. For instance, the proposed algorithm requires precise estimation of each worker's skill proficiency, and it may not account for unforeseen factors that can impact processing times, such as variations in worker performance. Overall, this study offers a significant improvement over traditional genetic algorithms and provides valuable insights into the scheduling of multi-skilled workers in human–machine integrated industries.

The rest of this paper is organized as follows: Section 2 discusses related studies. Section 3 defines the scheduling problem. Section 4 examines the properties of the problem and proposes an upper bound to evaluate the performance of metaheuristic algorithms. Section 5 introduces a genetic algorithm designed to generate high-quality schedules for large problem instances. Computational experiments are conducted in Section 6. Section 7 includes a small case study. Finally, Section 8 concludes this study.

2. Related work

This section first examines the current state of human–machine integrated industries. Then, it reviews the development of multi-machine scheduling. Additionally, it explores how current metaheuristic algorithms achieve job rejection.

2.1. Human–machine integrated industries

Human–machine integrated industries, particularly those requiring multi-skilled workers, are increasingly prevalent today. In such industries, workers are expected to possess a diverse range of skills and abilities, often spanning multiple disciplines. For example, in the game industry [42], professionals are adept at game planning, scriptwriting, directing, graphic design, sound effects, programming, and testing. This interdisciplinary requirement ensures that a game developer can contribute to various aspects of game development and enhance its efficiency. Similarly, in the manufacturing industry, workers need to be skilled in operating various machines, using different tools, and carrying out basic maintenance. Studies [10,15] showed that workers are required to be adept at robotics control, quality inspection, or routine maintenance. That is, they must be highly collaborative and multifaceted. Another example is the agricultural industry. As highlighted by Esenam [14], a farmer might need to be proficient in satellite, drone, IoT, and robotics. These skills are essential for increasing efficiency, precision, and profit. These examples demonstrate the increasing need for multifaceted workers in industries where the integration of human skills and technological advancements is critical.

Assigning a cross-functional job in its entirety to a single worker, rather than dividing it among several workers, can be advantageous for several reasons. First, this strategy preserves trade secrets and confidential business information by centralizing the exposure of sensitive information to a single individual instead of a team. Second, it retains efficiency in job scheduling, as it eliminates the need for coordination among multiple workers. Third, a single worker possessing multiple skills ensures consistent quality. Fourth, assigning a job to one worker reduces administrative and operational overheads by simplifying payroll, training, and supervisory requirements. Fifth, this strategy offers greater flexibility and adaptability, as a multi-skilled worker can adapt to various aspects and make real-time adjustments. Sixth, a multi-skilled worker has a holistic understanding of a cross-functional job, enhancing decision-making and problem-solving capabilities. Seventh, having superior competencies can also increase workers' motivation, contributing to improved performance. Such a worker understands how various components fit together, allowing for more integrated approaches.

In summary, while splitting a job among multiple workers may be beneficial in some industries, there are significant advantages to assigning a cross-functional job to a single multi-skilled worker in the industries discussed above, particularly in terms of efficiency, quality control, cost management, and flexibility. However, research on scheduling such versatile jobs requiring multiple skills is seldom addressed. This gap underscores the need for further research on job scheduling in human–machine integrated industries, specifically focusing on effectively matching self-contained and cross-functional jobs with multi-skilled workers.

2.2. Multi-machine scheduling

Past research on multi-machine scheduling has largely focused on simple or monodisciplinary jobs on parallel identical machines. Cheng *et al.* [11] proposed a genetic algorithm for minimizing maximum tardiness on identical machines, solving 30-job and 5-machine problem instances near-optimally. Chaudhry *et al.* [8] addressed total tardiness minimization by allocating workers to identical machines, scheduling only six jobs with 10 equally capable workers distributed among three machines, indicating that more workers per machine suggest higher performance. Arik *et al.* [3] aimed to minimize total tardiness on identical machines, with their genetic algorithm solving 400-job and 20-machine instances. They proposed a simple bound to evaluate their metaheuristic algorithm, serving as an indicator of solution quality. Chen *et al.* [9] scheduled serving times for multiple cooks with identical abilities, using the built-in genetic algorithm in MATLAB 2023a to schedule 2 cooks handling 8 meals, with each solution taking approximately 25 s. Even with identical capabilities, these multi-machine scheduling problems are often NP-hard, necessitating efficient metaheuristic algorithms.

On the other hand, heterogeneous machines are seldom discussed in the field of multi-machine scheduling. Liu [27] developed a genetic algorithm to minimize total tardiness on parallel machines with different capabilities. Due to the NP-hardness of the scheduling problem, this genetic algorithm was only able to generate an approximate schedule for 120 jobs within 10 min. Moreover, no error bound was provided for these large problem instances, *i.e.*, unassured solution quality. Li *et al.* [26] also minimized the total tardiness on non-identical

machines. In this study, the proposed genetic algorithm was able to solve 850-job and five-machine problem instances. Again, their jobs were simple or monodisciplinary. On the other hand, in references [42, 43], each worker is characterized by multiple skills, and each job is categorized as belonging to one of several different types. While workers in these studies are multi-skilled and can handle a variety of job types, each job itself remains specialized, being either monodisciplinary or single-functional. Boukedroun *et al.* [6] developed a custom genetic algorithm to address a job-shop scheduling problem involving 15 machines processing 30 jobs. Each machine is specialized and can only handle the parts it is designed for.

In summary, scheduling cross-functional jobs is seldom addressed in the field of job scheduling, and there are few algorithms that effectively schedule heterogeneous workers and machines to process such jobs. Consequently, there is a need for further research into metaheuristic algorithms capable of scheduling multi-skilled workers and cross-functional jobs.

2.3. Tardiness and rejection

Minimizing total tardiness is a crucial strategy for accepting all jobs as a form of loss avoidance. First, adjusting schedules to accommodate late jobs can lead to better resource allocation and higher productivity [36]. Second, rejecting jobs might lead to dissatisfaction and damage an organization's market standing [32]. Third, completing tardy jobs, even with some delay, may be more profitable than rejecting them [35]. Fourth, it demonstrates a commitment to clients' needs, strengthens client relationships, and encourages repeat business [34]. Fifth, the experiences gained can be used to improve the efficiency and effectiveness of future scheduling algorithms in the long term [13]. Therefore, this approach serves as a distinguishing merit, setting organizations apart from competitors, and represents a strategy that balances resource optimization, reputation management, profitability, and fostering client relationships. Additional research on tardiness minimization can be found in [5, 20, 21].

Rejecting jobs destined to be delayed is another strategy for preventing loss and avoiding penalties. First, rejecting jobs that are likely to cause tardiness can avoid penalties associated with delays [38]. Second, this strategy enhances reliability and allows for smoother operation [44]. Third, it helps maintain a high standard of quality [22]. Fourth, preemptive rejection of jobs sets clear expectations with clients and prevents dissatisfaction [4]. Fifth, rejecting late jobs can reduce the risk of overcommitment and poor performance [28]. Sixth, it contributes to operational stability by ensuring that the workload aligns with capacity, promoting long-term sustainability [40, 41]. In summary, rejecting tardy jobs is a proactive strategy for better resource allocation, fewer penalties, higher quality, and improved sustainability. For more details on job rejection, please refer to [1, 16, 30].

Although many studies have focused on job rejection, it is regrettable that hardly any research has explored metaheuristic algorithms that enhance efficiency by eliminating unnecessary encodings. Typically, these studies treat the problem as if the original size is fully actionable, which can result in exploring many unnecessary areas in the solution space. For instance, consider a situation where only 25 out of a nominal 50 jobs are actually feasible due to constraints. Proceeding as if all 50 jobs must be managed not only explores many unnecessary areas but also leads to inefficient use of computational resources. By effectively reducing the nominal problem size from 50 to 25, the performance of scheduling algorithms can be significantly enhanced. This adjustment in acknowledging the actual manageable load is crucial for optimizing both the solution space and algorithm efficiency.

2.4. Metaheuristic algorithm

In the field of job scheduling, particularly regarding total tardiness minimization, metaheuristic algorithms provide several significant advantages. First, these algorithms are designed to explore large and complex solution spaces, efficiently utilizing accumulated search experiences [39]. Second, they are highly adaptable and customizable for various types of scheduling problems, including linear programming, non-linear programming, constraint programming, and combinatorial optimization problems [2]. Third, many job scheduling problems,

especially those involving total tardiness minimization, are NP-hard, and metaheuristic algorithms are effective in providing feasible solutions within a reasonable time frame [25]. Fourth, metaheuristic algorithms are scalable, capable of handling large-scale problem instances while maintaining their effectiveness [19]. Fifth, they can be easily combined with other optimization techniques, such as Bayesian methods, to enhance their performance in solving complex scheduling problems [50]. In summary, metaheuristic algorithms bring a combination of adaptability, efficiency, and ease of application to total tardiness minimization in job scheduling, making them a valuable tool in the field.

Metaheuristic algorithms also have certain limitations, despite their benefits in total tardiness minimization. First, the quality of their solutions varies with different trials of a single problem instance. Consistency in achieving high-quality solutions is not always assured [24]. Second, they rely heavily on the appropriate choice and tuning of parameters [17]. Finding the right balance of parameters is challenging and time-consuming. Metaheuristic algorithms require a significant level of expertise and understanding of the problem domain, which can be a barrier for practitioners without this specialized knowledge. Third, they sometimes converge to local minima rather than global minima, particularly in complex solution spaces [33]. Fourth, metaheuristic algorithms can become computationally intensive and time-consuming when scaling to very large-scale problem instances or extremely constrained environments [7]. Fifth, due to the complex and stochastic nature of metaheuristic algorithms, conducting thorough theoretical analyses of their abilities and shortcomings is challenging [18]. Although metaheuristic algorithms are a powerful tool for total tardiness minimization, their application comes with challenges related to solution quality, parameter dependency, computational intensity, and the need for specialized knowledge, among others.

Past metaheuristic algorithms have shown limitations in three key areas. First, they lack mathematical error bounds to guarantee solution quality. Second, they often rely on local search strategies that focus narrowly on local minima, missing other potential solutions. Third, their encoding methods are fixed, lacking the flexibility to adapt to changes, which hinders their effectiveness in dynamic environments. These shortcomings highlight the need for an improved metaheuristic algorithm that incorporates error bounds for reliability and offers more adaptive and comprehensive solution strategies.

3. Problem definition

A job partition and scheduling problem for maximizing profit is defined as follows. For a given time period, there are n non-preemptive jobs to be assigned to m multi-skilled workers; each job requires three types of expertise, and each job has a due date d_j and a profit o_j. Each worker can process one job at a time and each job can be assigned to only one worker. Let p_{jk} be the default processing time of job j in expertise k, and r_{ik} be the time consumption rate of worker i in expertise k, for $i = 1, 2, \ldots, m$, $j = 1, 2, \ldots, n$, and $k = 1, 2, 3$. If job j is assigned to worker i, the actual processing time is denoted by $\sum_{k=1}^{3} r_{ik} p_{jk}$. If job j is selected to form a schedule π, let $C_j(\pi)$ be the completion time of job j and $T_j(\pi)$ be the tardiness of job j. No tardiness is allowed in this problem. That is, some jobs may be rejected due to capacity limitations. Under the above assumptions and constraints, the objective function $f(\pi)$ is defined as

$$\begin{aligned} &\text{Max} \sum_{j=1}^{n} x_j o_j, \\ &\text{s.t.} \\ &\quad x_j = 1 \qquad \text{if job } j \text{ is assigned}, \\ &\quad x_j = 0 \qquad \text{otherwise}, \\ &\quad T_j(\pi) = 0 \qquad \text{if } x_j = 1. \end{aligned}$$

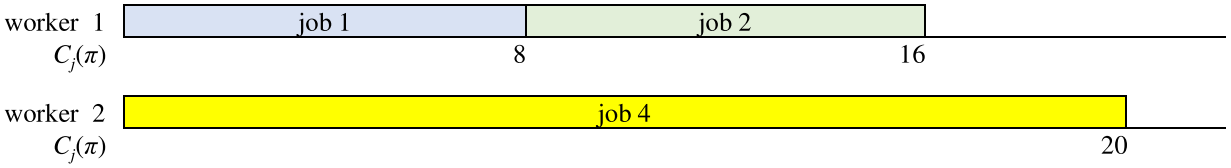
Figure 1 illustrates a problem instance. There are two workers, *i.e.*, $m = 2$, and five jobs, *i.e.*, $n = 5$. Each job's properties are listed in Figure 1a. On the other hand, each worker's abilities are listed in Figure 1b. Let

job \ expertise	$k=1$	$k=2$	$k=3$	due date	profit
$j=1$	$p_{jk}=4$	$p_{jk}=0$	$p_{jk}=4$	$d_j=8$	$o_j=5$
$j=2$	$p_{jk}=2$	$p_{jk}=2$	$p_{jk}=4$	$d_j=24$	$o_j=6$
$j=3$	$p_{jk}=8$	$p_{jk}=2$	$p_{jk}=4$	$d_j=20$	$o_j=2$
$j=4$	$p_{jk}=0$	$p_{jk}=4$	$p_{jk}=8$	$d_j=20$	$o_j=9$
$j=5$	$p_{jk}=6$	$p_{jk}=4$	$p_{jk}=0$	$d_j=16$	$o_j=3$

(a)

worker \ expertise	$k=1$	$k=2$	$k=3$
$i=1$	$r_{jk}=1$	$r_{jk}=1$	$r_{jk}=1$
$i=2$	$r_{jk}=2$	$r_{jk}=3$	$r_{jk}=1$

(b)



(c)

FIGURE 1. A problem instance of the presented problem. (a) A problem instance of five jobs. (b) A problem instance of two workers. (c) A schedule $\pi = (1, 2, 6, 4, 7, 3, 5)$.

$\pi = (1, 2, 6, 4, 7, 3, 5)$ be a schedule, where the numbers greater than n, *i.e.*, 6 and 7, indicate separators. That is, jobs 1 and 2 are assigned to worker 1, job 4 is assigned to worker 2, and jobs 3 and 5 are rejected. Note that worker 1 is good at every type of expertise, while worker 2 excels in processing jobs requiring expertise 3 but is mediocre in processing jobs requiring expertise 2. For example, the processing time of job 4 is $20 (= 2 \times 0 + 3 \times 4 + 1 \times 8)$. In schedule π, jobs 1, 2, and 4 have no tardiness. Therefore, schedule π can earn profit of $20 (= 5 + 6 + 9)$.

The following features distinguish this presented problem from past research. First, while most heterogeneous machine scheduling problems consider monodisciplinary jobs only, this problem involves multifaceted workers processing cross-functional jobs. Second, due to considerable problem sizes in the real world, exact algorithms become infeasible, especially in terms of time complexity. Third, to ensure solution quality, there is a need for error bounds to assess the solution quality of approximate algorithms. Consequently, there is a need for a new metaheuristic algorithm tailored to this job scheduling and rejection problem.

4. PROPERTIES OF THE PROBLEM

The presented scheduling problem differs from other scheduling problems in the following five characteristics. First, a job requires several types of expertise; for example, a video game comprises figures, scenes, sounds, programming, etc. This means that the processing time is not a fixed number and depends on which worker is assigned. For instance, learning effects can be applied to the jobs in [23, 46] because each job requires only a single type of expertise. However, the rules regarding learning or forgetting cannot be directly applied to the jobs in the presented problem.

Second, the processing speed of a worker varies with different jobs. In conventional homogeneous or heterogeneous machine scheduling problems, the processing speed of a machine is usually fixed. An example of

non-identical parallel machine scheduling is provided in [47], where each asphalt milling machine has a different processing speed and environmental cost. Even so, the workloads can be easily balanced between these heterogeneous machines; *i.e.*, a more capable machine processes more jobs. However, in the presented problem, a chef may be proficient at making Chinese food but not at baking, so the merits of a worker cannot be easily judged.

Third, the presented problem is different from the original 0–1 knapsack problem [12]. It is evident that this problem is NP-hard. Even if the problem degenerates into a single worker with a single type of expertise, with jobs requiring this single type of expertise and a uniform due date, it remains an NP-hard problem, similar to the original 0–1 knapsack problem [12]. Furthermore, developing an upper bound by merely sorting the CP ratios of jobs is not feasible. For instance, in the original 0–1 knapsack problem, jobs can be sorted in descending order of value/weight, and the rear ones with low CP ratios can be abandoned. However, in this presented problem, jobs cannot be simply ranked based on a single type of expertise. A more detailed exploration of the properties of these jobs is essential.

Fourth, developing an upper bound is more complicated for the presented problem. In some traditional lower bound studies, such as [42,48], each created a virtual machine to replace parallel machines/workers and aimed to develop a tight lower bound. However, when estimating an upper bound for the presented problem, there is no equivalent machine for all workers. This study explains this phenomenon with the following counterexample. Baker Bob and chef Carl deal with bakery and Chinese food in time consumption ratios of $r_{\mathrm{B}} = (1, 2)$ and $r_{\mathrm{C}} = (2, 1)$ respectively. There are one unit of bakery job J_1 and 1 unit of Chinese food job J_2. If they do what they are best at, only one unit of time is spent overall. On the other hand, if they are regarded as a virtual machine, J_1 will take 2/3 units of time; *i.e.*, Bob completes 2/3 of the job and Carl 1/3 during the time period. Subsequently, J_2 will take 2/3 units of time; *i.e.*, Bob completes 1/3 of the job and Carl 2/3. That is, the total processing time of this virtual machine is 4/3 ($= 2/3 + 2/3$). Evidently, this estimated magnitude (*i.e.*, 4/3) is much greater than the real one (*i.e.*, 1). The reason is that everyone does more or less what he is not good at in the virtual environment. Consequently, when developing an upper bound for the presented problem, the processing time of all workers cannot be estimated by treating them as a single virtual machine. Hence, a more effective upper bound is required.

According to the above observations, an upper bound to evaluate the performance of metaheuristic algorithms is proposed. First, the maximal time interval of job scheduling and each worker's CP ratio for each type of expertise are defined. Namely, there are $m \times n$ CP ratios of u_{ji} (see Def. 1). The pseudocode of the proposed upper bound for this profit maximization problem is shown in Figure 2.

Definition 1. Let $D = \max\{d_j | j = 1, 2, \ldots, n\}$ denote the largest due date and u_{ji} be the profit if worker i processes job j within a unit time for $i = 1, 2, \ldots, m$ and $j = 1, 2, \ldots, n$.

Figure 2 shows the pseudocode of the proposed upper bound. The main idea behind the upper bound is preemption and best fitting. Step 1 sorts the CP ratios according to their earned profit per unit time. That is, we have $m \times n$ opportunities to test if a job can be assigned to its best fitting worker. In Steps 2–4, some parameters are initialized. Then a CP ratio is popped; note that the first u_{ji} means the most profitable match of job and worker; *i.e.*, worker i earns the highest profit during a unit time if he processes job j. Step 7 checks the processing time of job j. If it is not allocated and worker i is available during this time period, let worker i process the whole job and the profit is accumulated (Steps 8 and 9). Otherwise, worker i processes as much of job j as he can, and the remaining part is left for a later opportunity (Steps 10–12). Then, repeat the allocation until all the $m \times n$ opportunities are checked (Step 13). Finally, the accumulated profit is returned in Step 14.

Lemma 1. *The returned value $\bar{o}$ is an upper bound.*

Proof. This property can be proved by contradiction. Suppose that there are two schedules; *i.e.*, a sequence $\bar{\pi}$ leads to the upper bound $\bar{o}$ (in a preemptive way), and another optimal schedule π^* leads to o^* such that $o^* > \bar{o}$. Sort the job slices in $\bar{\pi}$ into queue $\bar{Q}$ in descending order of CP ratio; sort jobs in π^* into queue Q^* in descending order. Now the two queues must differ from each other at some position, say the kth position at

Algorithm *UB*
INPUT
m is the number of workers;
n is the number of jobs;
r_{ik} is the time consumption rate of worker i in expertise k;
p_{jk} is the default processing time of job j in expertise k;
o_j is the profit of job of j;
OUTPUT
$\bar{o}$ is the upper bound;

---

1) Sort u_{ji}'s into a queue Q in descending order;
2) **For** $i = 1$ **to** m do set Time[i] = 0;
3) **For** $j = 1$ **to** n do set Allocated[j] = False;
4) Set $\bar{o} = 0$;
5) **Repeat** Steps 6–13;
6) Set $u_{ji} = \text{Pop}(Q)$;
7) Set $t = p_{j1}r_{i1} + p_{j2}r_{i2} + p_{j3}r_{i3}$;
8) **If** (not Allocated[j])and(Time[i] $+t \le D$) **then** do Step 9;
9) Set Time[i] = Time[i] + t, $\bar{o} = \bar{o} + o_j$, and Allocated[j] = True;
10) **Else if** (not Allocated[j])and(Time[i] $< D$) **then** do Steps 11–12;
11) Set $\varepsilon = (D$ - Time[i]), Time[i] = D, and $\bar{o} = \bar{o} + \varepsilon o_j$;
12) Set $o_j = o_j(1\text{-}\varepsilon)$, $p_{j1} = p_{j1}(1\text{-}\varepsilon)$, $p_{j2} = p_{j2}(1\text{-}\varepsilon)$, and $p_{j3} = p_{j3}(1\text{-}\varepsilon)$;
13) **Until** queue Q is empty;
14) Return $\bar{o}$.

FIGURE 2. The proposed upper bound.

$u_{j'i'}$. That is, their u_{ji}'s at the first $k-1$ positions are the same. Since $o^* > \bar{o}$, worker i' processes job j' and earns less profit in $\bar{Q}$ than Q^*. There are two reasons: Either most of job j' is completed earlier or worker i' is not always available during $u_{j'i'}$ in $\bar{Q}$. According to the procedure in Algorithm UB, worker i' and job j' are the best fitting pair that can make the most profit during $u_{j'i'}$. This implies that only a small portion of job j' is left at position k in $\bar{Q}$ such that worker i' cannot process as large partition of job j' as he does in Q^*. However, the u_{ji}'s of both queues at the first $k-1$ positions are the same, meaning that the remaining part of job j' is the same at position k in both queues. That is, there is no such larger portion of job j' that can be chosen by worker i' during $u_{j'i'}$ in Q^*. This leads to a contradiction. Therefore, the original assumption is false, and the lemma is proved. □

Lemma 1 proves the correctness of the proposed upper bound. Since the upper bound is implemented in a preemptive way, it ensures that each time slice earns the greatest possible profit. For an optimal schedule π^*, the scheduling unit is the job, rather than the time slice. Clearly, the profit achieved by π^* will not exceed the profit $\bar{o}$ obtained by Algorithm UB.

Fifth, under certain conditions, the maximum profit can be achieved. Consider three virtual workers, labeled 1, 2, and 3. Each virtual worker k, possessing only expertise k, has a time consumption rate defined as r_{ik} for $k = 1$, 2, and 3. When these three workers are collectively considered, the cost performance (CP) ratios of jobs become deterministic from their perspective, allowing the jobs to be sorted in descending order of CP ratio. The following two lemmas demonstrate the correctness of the obtained maximum profit.

Lemma 2. *For a given problem instance, the overall time consumption rate of the three virtual workers is equivalent to that of the m real workers.*

Proof. For expertise 1, the amount of job completed by worker i in a unit time is $1/r_{i1}$. The total amount completed by the m workers in this unit time is $\sum_{i=1}^{m} 1/r_{i1}$. Assign the total amount to virtual worker 1 and his completion time will be $(\sum_{i=1}^{m} 1/r_{i1}) \times 1/(\sum_{i=1}^{m} 1/r_{ik}) = 1$. Therefore, the m real workers can process the same amount of jobs in expertise 1 as virtual worker 1. That is, the m real workers can process the same amount of jobs in expertise 1 as virtual worker 1. Similarly, the rule holds for expertise 2 and 3. Therefore, the overall time consumption rate of the three virtual workers is equivalent to that of the m real workers. This completes the proof. □

The following definition aims to sort all jobs in descending order of CP ratio. With a deterministic job sequence, the three virtual workers can continuously and concurrently process the jobs they deem most valuable. Subsequently, the lemma demonstrates that the obtained profit is the highest that these real workers can achieve.

Definition 2. Let $P_{jk} = p_{jk}/(\sum_{i=1}^{m} 1/r_{ik})$ be the processing time of job j in expertise k and processed by virtual worker k for $k = 1, 2, 3$.

Lemma 3. *For a given problem instance, let the n jobs be sorted in descending order of* $o_j/(P_{j1} + P_{j2} + P_{j3})$. *If there exists a number Z such that the first Z jobs with* $\sum_{j=1}^{Z} p_{(j)1} = \sum_{j=1}^{Z} p_{(j)2} = \sum_{j=1}^{Z} p_{(j)3} = D$, *then the maximal profit of the problem instance is less or equal to* $\sum_{j=1}^{Z} o_{(j)}$, *where* (j) *means the job in the* j*th position of the sorted job list.*

Proof. This property can be proved by contradiction; *i.e.*, suppose there exists an optimal schedule which achieves a greater profit. Let π denote the sorted list of the first Z jobs and π^* the optimal schedule. First, there is some job j in π but not in π^*; otherwise, it means π^* contains all the jobs in π and has an additional job j'. This implies that job j' is more profitable. However, note that the first Z jobs are most valuable and are processed by the three virtual workers without any idle time during the time interval $[0, D]$. Since the three virtual workers are a substitute of the m real workers, it implies that the m real workers cannot earn more profit within the same time interval, *i.e.*, a contradiction. Second, there is some job j'' in π^* but not in π. For the three equivalent virtual workers, they process the most profitable Z jobs at full pace without any idle time. However, if π^* contains a less profitable job j'' and still achieves a greater profit, it leads to a contradiction. Therefore, the assumption of an optimal schedule achieving a greater profit is false, and the maximal profit is indeed less than or equal to $\sum_{j=1}^{Z} o_{(j)}$. This completes the proof. □

Lemma 3 reveals that the optimal profit is attainable when the accumulated volume of these valuable jobs in each expertise aligns with the combined speed of the workers in each respective expertise.

5. Genetic algorithm

In this section, a genetic algorithm is proposed based on the structure outlined by Schaller [37]. This propose algorithm differs from previous ones in two key aspects: narrowed crossover and mutation ranges, and improved biodiversity. First, given the problem's partition nature, organizing rejected jobs is unnecessary. Thus, they should be excluded from crossover and mutation operations. Second, in traditional genetic algorithms, evolution occurs within a closed system, resulting in offspring that are largely identical over time, with most genes in the same positions on each chromosome. This similarity necessitates external intervention to enhance biodiversity. This strategy is akin to introducing new bloodlines into a closed noble family to prevent inbreeding and enrich genetic diversity.

The pseudocode of the proposed genetic algorithm is shown in Figure 3. In the encoding phase (Step 1), the length of a chromosome is $n + m$, as shown in Figure 1. Each chromosome has m numbers greater than n used to divided jobs into $m + 1$ partitions. That is, the first m parts of jobs are assigned to the m workers

Algorithm *GA*(r_c, r_m, N, n)
INPUT
r_c is the crossover rate;
r_m is the mutation rate;
N is the population size;
n is the number of jobs;
OUTPUT
π^+ is a near-optimal chromosome;
---
1) **For** i = 1 **to** N do generate chromosome π_i at random;
2) Set *elapsedTime* = 0;
3) **While** (*elapsedTime* < n/10) **do** Steps 4–11;
4) **For** i = 1 **to** N do Steps 5–7;
5) Select two chromosomes π_a and π_b by a tournament operator;
6) **If** *Random*()<r_c **then** generate π_i by a uniform order-based crossover on π_a and π_b;
7) **Else** let π_i= π_a survive into the current generation;
8) **For** i = 1 **to** N do Step 9;
9) **If** *Random*()<r_m **then** modify π_i by a shift mutation operator;
10) Set $\pi^+ = \pi_i$, if some π_i achieves the maximal profit in the current generation;
11) Record the *elapsedTime*;
12) Output this near-optimal chromosome π^+.

FIGURE 3. The proposed genetic algorithm.

and the last part will be rejected. For example, the schedule $\pi = (1, 2, 6, 4, 7, 3, 5)$ in Figure 1 is encoded as $(1, 2, 6, 4, 7, 3, 5)$, exactly as shown in the figure. The fitness function for each chromosome is defined as $g(\pi) = \sum_{j=1}^{n}\{o_j|$ if job j is allocated in $\pi\} - \sum_{j=1}^{n} M \times T_j(\pi)$, where $M = \sum_{j=1}^{n} o_j$ is a predefined large number. Intuitively, each violation of tardiness does not fail a chromosome but adds a penalty to it. In this way, chromosomes will easily evolve towards higher profits. Moreover, the population size is denoted by N.

In the selection phase (Step 5), a strategy named three-tournament is adopted. Among the N chromosomes, three are randomly chosen, with the one yielding the highest profit selected as parent a. Another parent b is obtained similarly, and later, parents a and b are paired for mating. This implies that calculating the fitness of each chromosome is unnecessary.

In the crossover phase (Step 6), a uniform order-based crossover operator is adopted. The main difference between the proposed crossover operation and that of Schaller [37] is the position of the last separator, *i.e.*, the mth zero, say z. That is, crossover operations are only performed on the genes at positions 1 and z (*i.e.*, *pruning* unnecessary genes). Moreover, for every 10 generations, replace the last selected parent for crossover with a randomly generated chromosome to enhance biodiversity (*i.e.*, *outbreeding*).

In the mutation phase (Step 7), two genes are selected and swapped randomly. Suppose their positions of the two genes are x and y. Again, in the mutation operation, force that $\min(x, y) < z$, where z is the position of the mth zero in a chromosome. Finally, the nearly-optimal chromosome π^+ is returned (Step 12).

Clearly, this genetic algorithm excels in dealing with NP-hard scheduling problems with many closely spaced local optimums, necessitating safeguards against entrapment in these local optimums. It is also effective in solving similar integer permutation or combination problems, such as the 0–1 knapsack problem, where evolution is accelerated by discarding unnecessary elements.

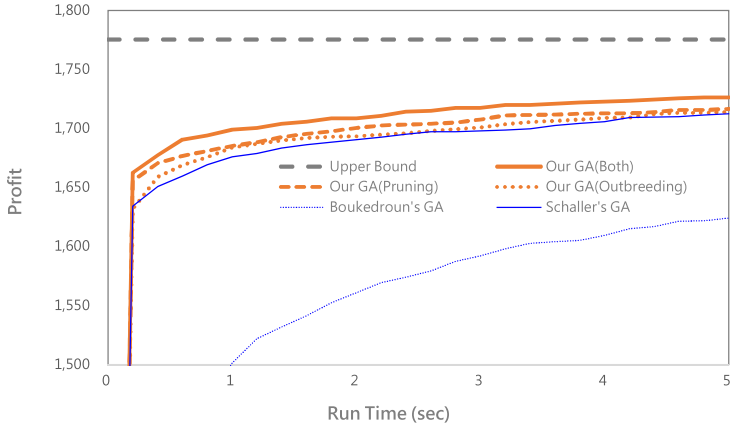


FIGURE 4. The performances of multiple genetic algorithms.

6. COMPUTATIONAL RESULTS

In the following experiments, all the related algorithms are implemented in Object Pascal and executed on an Intel Core i7 @ 3.40 GHz with 32 GB RAM in a Windows 10 environment. For both genetic algorithms, *i.e.*, GA(Ours), GA(Schaller) [37], and GA(Boukedroun) [6], let their population sizes be 100, crossover rates 08, and mutation rates 0.2. Default processing time $p_{jk} \sim DU(1, 100)$, time consumption rate $r_{ik} \sim DU(1, 3)$, profit $o_j. \sim DU(1, 100)$, and due date $d_j. \sim DU(1, 150\,n/m)$ follow several different discrete uniform distributions, respectively. Moreover, to observe the performances of the two genetic algorithms, m is set from 1 to 9, and n from 50 to 300. For each setting, 50 random trials are conducted and their earned profits are recorded, with each trial having a run time of $n/10$ s for every genetic algorithm.

6.1. Performance analysis

This subsection compares the performance of five different genetic algorithms. Figure 4 shows how these genetic algorithms evolve. First, a 50-job and three-worker problem instance is randomly generated. Then, calculate the upper bound of the instance with Algorithm UB (gray thick dashed line). Next, let the population size of each genetic algorithm be 100, *i.e.*, $N = 100$, perform each genetic algorithm 50 times, and record its performance every second. The average profits of the proposed algorithm (with both designs, *i.e.*, pruning redundant crossover and mutation ranges and improving biodiversity) and the other algorithm's average profits are represented by an orange thick solid line and a blue thick dotted line, respectively. Additionally, the average profits of each design (*i.e.*, pruning and outbreeding) are illustrated by an orange thin solid line and an orange thin dashed line, respectively. The results indicate that the proposed algorithm can converge quickly. In contrast, the other algorithm expends computational effort on rejected jobs, resulting in it becoming trapped in local optimums. For example, Boukedroun's GA is unsuitable for the presented problem for three reasons: First, it does not discard unnecessary genes in the tail part of a chromosome. Second, it lacks an outbreeding technique to enhance biodiversity. Third, it spends significant execution time on tabu-search-based local search but fails to escape local optimums. Consequently, its design is not suitable for the presented problem.

Table 1 presents a comparative analysis of two genetic algorithms across various problem sizes. For each setting, 50 random trials are conducted, each with a run time of $n/10$ s and a population size of 100 chromosomes for every genetic algorithm. Moreover, the relative deviation percentage (RDP) is defined as $((U-f)/U)\times 100\%$, where f means the profit obtained by a genetic algorithm and U stands for the upper bound obtained by Algorithm UB. The experimental results indicate that the RDP's of the proposed genetic algorithm do not exceed 18.6%. That is, these obtained schedules are comparatively closer to the optimal ones. In contrast, the other genetic algorithm demonstrates inferior solution quality, primarily because it expends excessive run time on processing rejected jobs, which hinders its ability to quickly converge to global maximums.

TABLE 1. The performances of the two genetic algorithms for different problem sizes.

n	Run time	GA (Ours)			GA (Schaller)			
		Mean profit	Max profit	RDP	Mean profit	Max profit	RDP	Upper bound
50	5	1631.97	1701	18.6%	1590.47	1661	20.7%	2004.72
100	10	2835.07	2886	16.4%	2768.80	2879	18.3%	3389.61
150	15	5709.40	5850	15.5%	5620.87	5828	16.8%	6759.44
200	20	6753.67	6865	12.4%	6592.40	6813	14.5%	7706.47
250	25	8696.17	8933	13.8%	8395.80	8632	16.8%	10 092.30
300	30	9491.97	9680	14.4%	9244.40	9601	16.6%	11 086.19

TABLE 2. The influence of m on performance for $n = 200$.

m	Run time	GA (Ours)				GA (Schaller)			
		Profit		RDP		Profit		RDP	
		Mean	Max	Mean	Max	Mean	Max	Mean	Max
1	20	1499.32	2528	8.4%	18.0%	1348.20	2473	17.7%	34.5%
3	20	3425.88	4646	17.2%	26.6%	3250.08	4664	21.4%	30.5%
5	20	4920.08	5783	20.6%	25.9%	4794.28	5693	22.6%	29.8%
7	20	6103.42	7525	22.9%	29.1%	5985.70	7168	24.4%	31.4%
9	20	7076.64	8131	23.9%	29.0%	6961.70	8010	25.1%	30.4%

Table 2 shows the impact of the number of workers on both genetic algorithms. The defaults are set as $n = 200$ and $N = 100$, and conduct each random trial within $n/10$ s. Clearly, employing just a single worker results in a high number of rejected jobs; having more workers indeed increases the profits of each genetic algorithm. Note that the proposed genetic algorithm focuses on crossover and mutation operations on the earlier jobs in a schedule, meaning that it is unnecessary to expend computational effort on the latter jobs in a schedule that are rejected and have no impact on the objective. Consequently, these results are able to more closely approximate the optimal solutions, achieving the lowest RDP of 8.4%. Conversely, when $m = 9$, almost all jobs are accepted. Under this condition, both genetic algorithms can effectively operate on the accepted 200 jobs, leading to similar solution qualities.

6.2. Sensitivity analysis

In this subsection, the number of excellent workers and the problem size are incrementally increased to demonstrate their influence on profit and solution quality. In Table 3, m' is defined as the number of excellent workers, characterized by their time consumption rates r_{ik} being uniformly 1. For the benchmark setting, where m' equals 0, the time consumption rates of five workers are randomly assigned values of 1, 2, or 3. It is observed that a greater number of excellent workers indeed leads to higher profits. However, when 200 jobs are distributed among five excellent workers, there are nearly no rejected jobs. Consequently, the proposed genetic algorithm does not have the opportunity to reduce computational effort by omitting rejected jobs.

Table 4 shows the influence of varying job numbers on the performance of both genetic algorithms. The defaults are set as $m = 5$ and $N = 100$, conducting each random trial within $n/10$ s. For smaller problem sizes (*e.g.*, $n = 100$), only a few jobs are rejected, limiting the algorithm's ability to capitalize on this aspect. As a result, both algorithms exhibit similar solution qualities under this condition. However, when the problem size increases to 300 while maintaining m at 5, this leads to a higher number of rejected jobs. Consequently, the proposed algorithm can bypass these rejected jobs, thereby achieving superior solution qualities.

TABLE 3. The influence of the number of excellent workers on performance for $n = 200$ and $m = 5$.

m'	Run time	GA (Ours)				GA (Schaller)			
		Profit		RDP		Profit		RDP	
		Mean	Max	Mean	Max	Mean	Max	Mean	Max
0	20	7168.56	9071	17.8%	24.9%	7026.48	8803	19.4%	25.0%
1	20	7709.82	9087	15.7%	20.2%	7568.38	8896	17.2%	23.7%
2	20	8162.88	9273	14.9%	19.2%	8034.60	9090	16.2%	20.5%
3	20	8567.24	9528	13.4%	17.4%	8424.84	9470	14.9%	18.6%
4	20	8859.30	9735	11.8%	16.0%	8745.26	9716	13.0%	17.1%
5	20	9052.82	9968	11.9%	15.7%	8963.36	9812	12.7%	16.5%

TABLE 4. The influence of n on performance for $m = 5$.

n	Run time	GA (Ours)				GA (Schaller)			
		Profit		RDP		Profit		RDP	
		Mean	Max	Mean	Max	Mean	Max	Mean	Max
100	10	3585.94	4096	20.0%	32.1%	3535.14	4070	21.2%	31.7%
150	15	5369.52	6146	18.7%	24.1%	5290.64	6056	19.9%	26.2%
200	20	7326.84	8508	17.2%	22.2%	7158.08	8370	19.2%	24.4%
250	25	8973.88	10 267	17.7%	21.4%	8761.40	10 026	19.7%	24.8%
300	30	10 709.44	12361	17.2%	23.7%	10 463.44	12 050	19.1%	23.9%

TABLE 5. A real problem instance for $m = 2$ and $n = 10$.

Job	Job 1	Job 2	Job 3	Job 4	Job 5	Job 6	Job 7	Job 8	Job 9	Job 10
$p_{j,1}$	1	0	8	0	1	4	6	3	6	2
$p_{j,2}$	1	9	10	2	3	4	4	7	2	4
d_j	10	10	10	10	30	40	15	90	10	10
o_j	5	20	20	10	16	35	50	55	4	20

In summary, the proposed genetic algorithm outperforms others in terms of solution quality by avoiding meaningless operations during the evolution process. For example, in a 50-job and three-worker problem instance, there might be 25 jobs that need to be rejected. The runtime saved from bypassing these jobs can be used to focus on scheduling only the accepted jobs, thereby achieving better performance.

7. REAL INSTANCE

Consider the following real problem instance from an IT company. There are two developers (*i.e.*, $m = 2$) and 10 candidate jobs (*i.e.*, $n = 10$). This company is contracted to handle the maintenance of consumables for various hospitals and clinics. Based on confidentiality agreements, each job can only be assigned to one developer and cannot be subcontracted. These jobs require two skills: web design and web programming. The processing time consumption rates for the two developers are $r_{1,1} = 0.7$, $r_{1,2} = 1.2$, $r_{2,1} = 1.1$, and $r_{2,2} = 0.8$. This means developer 1 is more proficient in web design, while developer 2 excels in web programming. Table 5 lists the general processing times, due dates, and profits for these 10 jobs.

For this instance, the proposed genetic algorithm provides a near-optimal schedule of $(10, 7, 5, 6, 8, 11, 2, 4, 12, 1, 3, 9)$, with a potential profit of 206. In this schedule, developer 1 is assigned

jobs $10, 7, 5, 6$, and 8, while developer 2 is assigned jobs 2 and 4. Due to limited manpower, the company has to abandon jobs 1, 3, and 9. It can be observed that developer 1 mostly handles high-profit jobs requiring web design, while developer 2 takes on high-profit jobs needing web programming. The jobs that are abandoned are generally those with lower profits or longer processing times.

The proposed genetic algorithm offers several advantages for job scheduling. It replaces manual scheduling, quickly matches multi-skilled developers to self-contained and cross-functional jobs, and objectively rejects low-profit, high-time jobs. However, it also has some drawbacks, such as the need for precise estimation of each developer's abilities and the inability to account for factors like negative mood affecting fixed processing time consumption rates. Overall, this study provides significant assistance in matching multi-skilled workers to self-contained and cross-functional jobs.

8. Conclusion and managerial insight

In this study, an interesting partition and scheduling problem is presented. A novel genetic algorithm is proposed to generate near-optimal schedules for the problem at hand. The proposed genetic algorithm discards unnecessary genes to improve performance, employs outbreeding to enhance biodiversity, and provides an upper bound as a benchmark to measure solution quality.

8.1. Conclusion

Based on the findings of this study, several managerial insights can be gained. First, traditional genetic algorithms with fixed-length chromosomes may not be suitable for complex job scheduling problems involving multiple expertise areas. Therefore, it is recommended to develop adaptive metaheuristic algorithms that dynamically adjust chromosomes' lengths during evolution. Second, in human–machine integrated industries, it is crucial to consider jobs and workers as a whole, evaluating each job's CP ratio in relation to the worker's specific skills. This holistic approach can enhance scheduling efficiency and optimize resource allocation. Third, focusing on high-impact jobs and adopting flexible scheduling strategies can maximize productivity and profitability.

Despite the above advantages, there are several limitations to the proposed genetic algorithm. First, it is necessary to precisely estimate each worker's skills and the specific CP ratios for each job, which may not always be feasible in real-world scenarios. That is, these applications should be confined to industries where processing times can be accurately estimated. For instance, in the renovation industry, jobs such as connecting electricity, installing distribution systems, performing interior wiring, and installing lighting fixtures have predictable processing times. Second, the model assumes that job requirements and worker skills remain constant over time, which may not account for variations in performance due to factors like fatigue or skill improvement. Third, the error bound or upper bound for this study is not tight enough. Future research may develop more efficient approximation methods, similar to the Newton-Raphson method, where the errors can approach zero in some common distributions, such as the normal distribution. Fourth, the computational complexity may still be significant for extremely large problem instances, requiring further optimization and efficiency improvements.

Compared with traditional scheduling problems, the jobs addressed here are more complex, requiring multiple types of expertise. Each job's CP ratios cannot be directly ranked without considering its corresponding worker, necessitating that jobs and workers be considered together. By shrinking the operation ranges of rejected jobs, many unnecessary operations can be avoided. This indicates that the best parameter settings, such as the mutation rate, should vary during the evolution process. For example, the mutation rate could be $r_m = 0.1 + 0.5 \times g/G$, where G is the total generations and g represents the current generation.

In summary, the proposed genetic algorithm outperforms others in terms of solution quality by avoiding meaningless operations during evolution. For instance, in a 50-job and three-worker problem instance, there might be 25 jobs that need to be rejected, allowing the saved run time to be used for scheduling the accepted jobs and achieving better performance.

8.2. Managerial insight

This study offers several contributions to the field of management science. The proposed metaheuristic algorithms can significantly improve decision-making processes and operational efficiency in complex job scheduling environments. By dynamically adjusting chromosomes' lengths during the evolution process, these algorithms can provide more effective and efficient solutions, enhancing overall resource allocation and productivity.

The environmental impact of job scheduling cannot be understated. By efficiently matching complex jobs to suitable workers based on their specific skills, resource consumption can be minimized. This includes reducing energy usage and waste because unnecessary jobs and overuse of resources are avoided. Consequently, such algorithms can contribute to more sustainable industrial practices.

From a societal standpoint, the proposed algorithms can improve worker satisfaction and productivity. By ensuring that workers are assigned jobs that match their skills and expertise, stress and dissatisfaction can be reduced. This approach also supports better performance. Moreover, the overall productivity of an industry can be enhanced, benefiting the broader community and economy.

For future research, developing metaheuristic algorithms with variable crossover and mutation rates tailored to real-world requirements is essential. These algorithms should be capable of handling large problem instances involving multi-skilled workers and cross-functional jobs, providing near-optimal schedules. Further studies may explore integrating similar genetic algorithms into various industrial applications, assessing their long-term impact on efficiency and sustainability, and enhancing the accuracy of existing error bound methods.

Acknowledgements

The authors would like to express their sincere gratitude to the anonymous reviewers for their valuable comments.

Funding

This research received funding from the National Science and Technology Council, supported under grant number NSTC 112-2410-H-025-045.

References

[1] S. Afkhami, A.H. Kashan and B. Ostadi, Effective league championship algorithm and lower bound procedure for scheduling a single batch-processing machine with non-identical job sizes and job rejection. *RAIRO:RO* **57** (2023) 1453–1479.

[2] S. Aminzadegan, M. Tamannaei and M. Rasti-Barzoki, Multi-agent supply chain scheduling problem by considering resource allocation and transportation. *Comput. Ind. Eng.* **137** (2019) 106003.

[3] O.A. Arik, M. Schutten and E. Topan, Weighted earliness/tardiness parallel machine scheduling problem with a common due date. *Expert Syst. Appl.* **187** (2022) 115916.

[4] P.S. Biçakci, I. Kara and M. Sagir, Single-machine order acceptance and scheduling problem considering setup time and release date relations. *Arab. J. Sci. Eng.* **46** (2021) 1549–1559.

[5] C. Bierwirth and J. Kuhpfahl, Extended GRASP for the job shop scheduling problem with total weighted tardiness objective. *Eur. J. Oper. Res.* **261** (2017) 835–848.

[6] M. Boukedroun, D. Duvivier, A. Ait-el-Cadi, V. Poirriez and M. Abbas, A hybrid genetic algorithm for stochastic job-shop scheduling problems. *RAIRO:RO* **57** (2023) 1617–1645.

[7] J. Branke and C.W. Pickardt, Evolutionary search for difficult problem instances to support the design of job shop dispatching rules. *Eur. J. Oper. Res.* **212** (2011) 22–32.

[8] I.A. Chaudhry and P.R. Drake, Minimizing total tardiness for the machine scheduling and worker assignment problems in identical parallel machines using genetic algorithms. *Int. J. Adv. Manuf. Technol.* **42** (2009) 581–594.

[9] Y.C. Chen and J.Y. Wang, A lower bound for minimizing waiting time in coexisting virtual and physical worlds. *IEEE Access* **12** (2024) 73470–73480.

[10] J.C. Chen, Y.Y. Chen, T.L. Chen and Y.H. Kuo, Applying two-phase adaptive genetic algorithm to solve multi-model assembly line balancing problems in TFT-LCD module process. *J. Manuf. Syst.* **52** (2019) 86–99.

[11] R.W. Cheng, M.S. Gen and T. Tozawa, Minmax earliness tardiness scheduling in identical parallel machine system using genetic algorithms. *Comput. Ind. Eng.* **29** (1995) 513–517.

[12] T.H. Cormen, C.E. Leiserson, R.L. Rivest and C. Stein, Introduction to Algorithms. MIT Press (2009).

[13] T. Eren, A note on minimizing maximum lateness in an machine scheduling problem with a learning effect. *Appl. Math. Comput.* **209** (2009) 186–190.
[14] A. Esenam, Overview of digital agriculture: making growers lives more productive. *Int. Sugar J.* **119** (2017) 466–470.
[15] C. Firth, K. Dunn, M.H. Haeusler and Y. Sun, Anthropomorphic soft robotic end-effector for use with collaborative robots in the construction industry. *Autom. Constr.* **138** (2022) 104218.
[16] X.N. Geng, X.Y. Sun, J.Y. Wang and L. Pan, Scheduling on proportionate flow shop with job rejection and common due date assignment. *Comput. Ind. Eng.* **181** (2023) 109317.
[17] F.P. Golneshini and H. Fazlollahtabar, Meta-heuristic algorithms for a clustering-based fuzzy bi-criteria hybrid flow shop scheduling problem, *Soft Comput.* **23** (2019) 12103–12122.
[18] I. González-Rodríguez, J. Puente, J.J. Palacios and C.R. Vela, Multi-objective evolutionary algorithm for solving energy-aware fuzzy job shop problems. *Soft Comput.* **24** (2020) 16291–16302.
[19] J. Grobler, A.P. Engelbrecht, S. Kok and S. Yadavalli, Metaheuristics for the multi-objective FJSP with sequence-dependent set-up times, auxiliary resources and machine down time. *Ann. Oper. Res.* **180** (2010) 165–196.
[20] J.G. Kim, S. Song and B. Jeong, Minimising total tardiness for the identical parallel machine scheduling problem with splitting jobs and sequence-dependent setup times. *Int. J. Prod. Res.* **58** (2020) 1628–1643.
[21] J. Kuhpfahl and C. Bierwirth, A study on local search neighborhoods for the job shop scheduling problem with total weighted tardiness objective. *Comput. Oper. Res.* **66** (2016) 44–57.
[22] M. Kumar and S.C. Sharma, Deadline constrained based dynamic load balancing algorithm with elasticity in cloud environment. *Comput. Electr. Eng.* **69** (2018) 395–411.
[23] W.C. Lee, A note on single-machine scheduling with general learning effect and past-sequence-dependent setup time. *Comput. Math. Appl.* **62** (2011) 2095–2100.
[24] W.C. Lee and J.Y. Wang, A three-agent scheduling problem for minimizing the flow time on two machines. *RAIRO:RO* **54** (2020) 307–323.
[25] Z. Li, R.Y. Zhong, A.V. Barenji, J.J. Liu, C.X. Yu and G.Q. Huang, Bi-objective hybrid flow shop scheduling with common due date. *Oper. Res.* **21** (2021) 1153–1178.
[26] K. Li, H. Zhang, C.B. Chu, Z.H. Jia and Y. Wang, A bi-objective evolutionary algorithm for minimizing maximum lateness and total pollution cost on non-identical parallel batch processing machines. *Comput. Ind. Eng.* **172** (2022) 108608.
[27] C.F. Liu, A hybrid genetic algorithm to minimize total tardiness for unrelated parallel machine scheduling with precedence constraints. *Math. Probl. Eng.* **2013** (2013) 537127.
[28] G. Lucarelli, B. Moseley, N.K. Thang, A. Srivastav and D. Trystram, Online non-preemptive scheduling on unrelated machines with rejections. *ACM Trans. Parallel Comput.* **8** (2021) 1–22.
[29] L. Luo, Z. Zhang and Y. Yin, Modelling and numerical analysis of seru loading problem under uncertainty. *Eur. J. Ind. Eng.* **11** (2017) 185–204.
[30] B. Mor, Single machine scheduling problems involving job-dependent step-deterioration dates and job rejection. *Oper. Res.* **23** (2023) 10.
[31] G. Mosheiov and A. Sarig, A common due-date assignment problem with job rejection on parallel uniform machines. *Int. J. Prod. Res.* **62** (2023) 2083–2092.
[32] F. Pargar, M. Zandieh, O. Kauppila and J. Kujala, The effect of worker learning on scheduling jobs in a hybrid flow shop: a bi-objective approach. *J. Syst. Sci. Syst. Eng.* **27** (2018) 265–291.
[33] S.G. Ponnambalam, V. Ramkumar and N. Jawahar, A multiobjective genetic algorithm for job shop scheduling. *Prod. Plan. Control* **12** (2001) 764–774.
[34] V. Portougal and D. Trietsch, Setting due dates in a stochastic single machine environment. *Comput. Oper. Res.* **33** (2006) 1681–1694.
[35] M. Reisi-Nafchi and G. Moslehi, Two-agent order acceptance and scheduling to maximise total revenue. *Eur. J. Ind. Eng.* **9** (2015) 664–691.
[36] A.P. Rifai, H.T. Nguyen and S.Z.M. Dawal, Multi-objective adaptive large neighborhood search for distributed reentrant permutation flow shop scheduling. *Appl. Soft Comput.* **40** (2016) 42–57.
[37] J.E. Schaller, Minimizing total tardiness for scheduling identical parallel machines with family setups. *Comput. Ind. Eng.* **72** (2014) 274–281.
[38] D. Shabtay, N. Gaspar and L. Yedidsion, A bicriteria approach to scheduling a single machine with job rejection and positional penalties. *J. Comb. Optim.* **23** (2012) 395–424.
[39] Y.R. Shiau, M.S. Tsai, W.C. Lee and T.C.E. Cheng, Two-agent two-machine flowshop scheduling with learning effects to minimize the total completion time. *Comput. Ind. Eng.* **87** (2015) 580–589.

[40] S.A. Slotnick, Order acceptance and scheduling: a taxonomy and review. *Eur. J. Oper. Res.* **212** (2011) 1–11.
[41] M. Stevenson, Y. Huang and L.C. Hendry, The development and application of an interactive end-user training tool: part of an implementation strategy for workload control. *Prod. Plan. Control* **20** (2009) 622–635.
[42] C.H. Su and J.Y. Wang, A branch-and-bound algorithm for minimizing the total tardiness of multiple developers. *Mathematics* **10** (2022) 1200.
[43] C.H. Su and J.Y. Wang, A makespan minimization problem for versatile developers in the game industry. *RAIRO:RO* **56** (2022) 3895–3913.
[44] X.Y. Tang, Y. Liu, Z. Zeng and B. Veeravalli, Service cost effective and reliability aware job scheduling algorithm on cloud computing systems. *IEEE Trans. Cloud Comput.* **11** (2023) 1461–1473.
[45] S. Thevenin, N. Zufferey and M. Widmer, Order acceptance and scheduling with earliness and tardiness penalties. *J. Heuristics* **22** (2016) 849–890.
[46] M.D. Toksari and B. Atalay, Some scheduling problems with job rejection and a learning effect. *Comput. J.* **66** (2023) 866–872.
[47] J.Y. Wang, Algorithms for minimizing resource consumption over multiple machines with a common due window. *IEEE Access* **7** (2019) 172136–172151.
[48] J.Y. Wang, A branch-and-bound algorithm for minimizing the total tardiness of a three-agent scheduling problem considering the overlap effect and environment protection. *IEEE Access* **7** (2019) 5106–5123.
[49] Y. Yin, K.E. Stecke, M. Swink and I. Kaku, Lessons from seru production on manufacturing competitively in a high cost environment. *J. Oper. Manag.* **49–51** (2017) 67–76.
[50] R. Zhang, S.J. Song and C. Wu, A dispatching rule-based hybrid genetic algorithm focusing on non-delay schedules for the job shop scheduling problem. *Int. J. Adv. Manuf. Technol.* **67** (2013) 5–17.