Date of publication xxxx 00, 0000, date of current version xxxx 00, 0000.

*Digital Object Identifier 10.1109/ACCESS.2017.DOI*

# Weighted Multi-Skill Resource Constrained Project Scheduling: A Greedy and Parallel Scheduling Approach

**SAEED AKBAR[1], MUHAMMAD ZUBAIR [2], RIZWAN KHAN [1], UBAID UL AKBAR [3], RAHMAT ULLAH[4,*], and ZHONGLONG ZHENG[1,*]**

[1]School of Computer Science and Technology, Zhejiang Normal University, Jinhua 321004, China
[2]Department of Software Engineering, Lahore Garrison University, Pakistan
[3]Department of Computer Science, City University of Science and Information Technology, Pakistan
[4]School of Computer Science and Electronic Engineering, University of Essex, UK

*Corresponding author: Zhonglong Zheng (e-mail: zhonglong@zjnu.edu.cn)

**ABSTRACT** This study addresses the Weighted Multi-Skill Resource Constrained Project Scheduling Problem (W-MSRCSPSP) with the aim of minimizing software project makespan. Unlike previous works, our investigation regards heterogeneous resources characterized by varying skill proficiency levels. Another major problem with existing methodologies is the potential underutilization of human resources due to varying task durations. This work introduces an innovative scheduling approach known as the Greedy and Parallel Scheduling (GPS) algorithm to handle the said issues. GPS focuses on assigning the most suitable resources available to project activities at each scheduling point. The fundamental goal of our proposed approach is to reduce resource wastage while efficiently allocating surplus resources, if any, to project tasks, ultimately leading to a decrease in the makespan. To empirically evaluate the efficacy of the GPS algorithm, we conduct a comparative analysis against the Parallel Scheduling Scheme (PSS). The advantage of our proposed approach lies in its ability to optimize the utilization of available resources, resulting in accelerated project completion. Results from extensive simulations substantiate this claim, demonstrating that the GPS scheme outperforms the PSS approach in minimizing project duration.

**INDEX TERMS** Greedy and Parallel Scheduling, Heterogeneous skill proficiency, Parallel Scheduling Scheme, Project scheduling, Resource assignment, Weighted multi-skilled resources.

## I. INTRODUCTION

PROJECT scheduling is an integral step in software project management that involves assigning resources over time to execute project tasks. The timely completion of a project is intricately linked to the decisions made in resource scheduling [1], [2]. An ideal project schedule achieves project completion within the stipulated timeframe while minimizing project costs, all the while maintaining the quality of the final deliverable. Project scheduling serves as the conduit that transforms a project plan into a sequence of project activities, optimizing the project's total duration, all within the confines of pre-existing constraints such as human resource and the task precedence relationships [3], [4]. The Resource-Constrained Project Scheduling Problem (RCPSP) seeks to optimize activity start and end times to achieve the earliest project completion, considering both prerequisite relationships and resource constraints [5]–[7]. Notably, human resources, often scarce with specific skill sets, pose a significant challenge when assigned to tasks adhering to precedence relationships. This challenge characterizes a more inclusive variant of RCPSP known as the Multi-Skill Resource-Constrained Project Scheduling Problem (MSRCPSP) [3], [8], [9].

Originating from Blazewicz et al.'s work [10], RCPSP remains a complex and enduring research domain, giving rise to various models accommodating diverse combinations of resources, activities, and objectives [11], [12]. Driven by the rapid growth of the software industry, MSRCPSP has gained substantial significance, particularly in the realm of medium and large-scale projects. In MSRCPSP, each resource possesses one or more distinct skills enabling them to handle various project tasks [13]. Task execution requires collaboration among resources with specific proficiencies. Notably, these renewable human resources can be allocated to multiple activities, provided there is no temporal overlap in their execution. The precise allocation of resources based on their skills is of paramount importance within the MSRCPSP domain [14].

Project scheduling problems are multifaceted, with constraints spanning deadlines, task precedence, resource availability, and skill requirements. Ultimately, the primary objective remains the efficient arrangement of all project activities to minimize the project's makespan while adhering to task precedence constraints [13], [15]. Numerous project strategies have been outlined within the literature with a primary emphasis on the minimization of the project makespan. Typically, project scheduling techniques are categorized into three distinct classes, as evidenced by the existing body of research. First among these classes are the exact methods [16], which harness integer programming models to ascertain the optimal solution. However, a prevailing limitation of these approaches arises when addressing problems characterized by substantial dimensions, as they tend to be computationally intensive.

To circumvent the challenges associated with exact methods, the research community introduces heuristic algorithms to expedite solution identification within reasonable time frames. Among these, parallel and serial schedule generation schemes have emerged as predominant heuristic approaches, demonstrating efficacy in practical project scheduling contexts. Nonetheless, it is noteworthy that these heuristic techniques do not universally ensure the attainment of optimal solutions. In response to this limitation, meta-heuristic algorithms have emerged as a pivotal advancement in the field, enhancing upon the capabilities of heuristic methods. A salient feature of meta-heuristic algorithms is their capacity to provide optimal solutions within a reasonable amount of time [16].

Literature offers numerous scheduling algorithms to address the MSRCPSPs, employing diverse exact methods [17], [18], heuristics [19]–[22], and meta-heuristic algorithms [5], [23]–[33]. These approaches take into consideration multiple constraints inherent to project scheduling when assigning human resources to project tasks, such as the number of project tasks, the precedence relationship among tasks, the number of resources, and the number of skills. Nonetheless, a prevalent limitation observed in these existing scheduling techniques lies in the overlook of skill proficiency heterogeneity among resources, assuming uniform proficiency levels for specific skills required in performing project tasks. Furthermore, the prevailing strategies may yield under-utilization of human resources due to variations in the duration of project tasks.

To address the aforementioned issues, this work offers a novel Greedy and Parallel Scheduling (GPS) technique. The present study focuses on the Weighted Multi-Skill Resource Constrained Project Scheduling Problem (WMSRCPSP) while explicitly considering the presence of heterogeneous resources, with the primary goal of minimizing the project makespan. The GPS approach first allocates human resources to project tasks in a greedy manner, employing a best-fit strategy to ensure optimal resource utilization. Also, it adeptly assigns resources to groups of parallel tasks, maximizing resource utilization. It accomplishes this by assigning the available, unoccupied resources—freed up as a result of recently completed tasks—to already running or subsequent project tasks in a manner that collectively minimizes the overall makespan. Each resource in this context possesses one or more skills, with varying efficiency levels for each skill. Its inherent greediness ensures that a substantial proportion of project resources remain actively engaged, thereby enhancing overall resource utilization. Additionally, this approach facilitates effective project monitoring and the desirable allocation of highly skilled resources to pertinent project activities by the project manager.

The principal contributions of this research can be summarized as follows:

- Introducing a novel Greedy and Parallel Scheduling (GPS) algorithm. GPS uses a best-fit strategy – it prioritizes efficiency by initially assigning the best available resources to the shortest tasks to minimize waiting times and optimize resource utilization. Moreover, the GPS also performs parallel execution of tasks by creating parallel executable task groups based on their precedence relationships in order to avoid resource wastage and minimize the overall makespan.
- Considering heterogeneous resources for performing project tasks. Incorporating a realistic weighting scheme that assigns resource weights for each skill based on their respective proficiency levels. Distinct weights are allocated for each skill mastered by individual resources. Additionally, the GPS reassigns free resources arising from the completion of preceding activities to minimize resource wastage during the resource allocation process.
- Conducting extensive simulations considering 6 different factors such as the number of skills, number of resources, and the graph complexity. Moreover, we conduct a comprehensive analysis of the simulation results to evaluate the effectiveness of the GPS strategy compared to the Parallel Scheduling Scheme (PSS) heuristic. Extensive experimental results affirm the superior performance of the proposed strategy in terms of minimizing the project makespan.

The subsequent sections of this paper are organized as follows. Section II presents the existing work related to project scheduling in the literature. Section III delineates the problem formulation. Section IV provides a brief discussion of the proposed framework for project scheduling and resource staffing. Finally, Sections **??** and VI present the simulation results and conclusion of this work, respectively.

## II. RELATED WORK

While task scheduling is vital in various domains beyond project management [34], [35], its settings and objectives differ from those in project management. Given the unique context and goals of MSRCPSP, specialized approaches and solutions are necessary. Thus, this study distinctly addresses the complexities of MSRCPSP within the project management context. Extensive literature exists that tries to unravel the complexities inherent to MSRCPSP. In contemporary academic discourse, a notable surge of interest among researchers centers on project scheduling, particularly with a focus on minimizing the project makespan.

In the context of MSRCPSP, Mirnezami et al. [17] propose a precise method for the Multi-mode Resource-Constrained Project Scheduling Problem (MRCPSP) with multiple skills, allowing activity preemption. They devise a novel Multi-Objective Mixed-Integer Linear Programming (MILP) model to handle uncertainty in non-renewable resources. The model minimizes project makespan, resource costs, and project risk. Shorter project durations increase resource consumption and costs, while higher resource usage tends to reduce project risk. Similarly, [36] proposes a MILP model for MSRCPSP, addressing overlapping activities and rework in a fuzzy environment. The model aims to optimize project duration, cost, and quality, handling uncertainty through fuzzy mathematical programming. However, the computational complexity of the said models is very high.

In addition to exact methods, literature suggests several heuristics-based approaches. For instance, in [21], the authors address the challenge of efficiently scheduling multiple projects with limited resources in high-end equipment development, employing Variable Neighborhood Search (VNS) for effective problem-solving within a reasonable timeframe. Their approach introduces a comprehensive scheduling model, considering multi-mode and multi-skill factors and emphasizing optimal allocation of research and development personnel and task sequencing. However, the assumption of a constant employee ability coefficient neglects potential skill improvement.

Similarly, Ref. [19] treats project scheduling as a multi-objective optimization problem and introduces a novel method to address the "new employee addition" dynamic event, utilizing domain knowledge for robust scheduling. The heuristic specifically handles scenarios involving mid-project additions of new employees, considering budget and time constraints. It also addresses the challenge of assigning one resource to multiple projects concurrently in a multi-project setting. Utilizing domain knowledge for population initialization, the approach aims to generate high-quality solutions. However, the approach overlooks certain dynamic events, such as considering task slack time and handling software requirements cancellation.

In [20], a mixed-integer goal programming model and a three-phase greedy heuristic method are introduced for the multi-project scheduling and multi-skilled employee assignment problem. The method addresses constraints like minimum task load and limits on tasks per week per employee, employing a local and tabu search algorithm. Tested on real data, the approach rapidly yields effective solutions for realistically sized instances. However, it lacks consideration for precedence relationships between projects or tasks. In a related context, Akbar et al. [4] propose a scheduling heuristic for MSRCPSP that accounts for the soft skills of human resources. This approach examines factors such as available resources, tasks per project, and precedence constraints, evaluating their impact on the overall project makespan.

A significant literature uses meta-heuristic approaches to study the MSRCPSP. For instance, Chen et al. [33] propose Hyper-Heuristic Filtered Genetic Programming (HH-FGP) to improve the efficiency of solving the Stochastic Resource-Constrained Multi-Project Scheduling Problem with Non-identical Parallel Machines (SRCMPSP-NPI). The framework refines Genetic Programming (GP) by simultaneously filtering both the attribute set and depth range of gene expression trees, offering a unique hyper-heuristic perspective for project scheduling in stochastic environments. Ma et al. [29] explore a proactive project scheduling problem with flexible resources possessing multiple skills, allowing skill changes at discrete intervals. They introduce a tabu search algorithm to generate a feasible skill allocation plan, aiming to maximize schedule robustness. However, the proposed method does not address the costs associated with skill switching.

Snauwaert et al. [9] study the impact of factors such as skill proficiency, workforce size, and multi-skilled resources. They find that beyond a certain skill level, additional skills may increase costs without improving makespan. Workforce size has a smaller impact than skill mastery, with the number of skills, not resources, primarily influencing makespan. The paper's limitation is its deterministic focus, neglecting the stochastic dimension of the MSRCPSP. In [23], Wang et al. address the MSRPSP with uncertain resource availability, introducing the GA-PR algorithm. It dynamically assigns skills, mitigating shortages by reallocating from idle resources, aiming to minimize additional costs and project makespan. However, the study considers homogeneous resources. Li et al. [24] introduce MODJaya for MSRCPSP, minimizing makespan and cost concurrently. MODJaya outperforms NSGA-II and NTGA in CPU time, with MOEA/D as the fastest, handling instances efficiently. The paper assumes single-resource task assignments, neglecting real-world collaboration scenarios involving multiple resources.

Li et al. [26] propose a two-stage priority rule-based heuristic algorithm enhanced with a genetic algorithm (GA) for the Software Project Scheduling Problem with Multi-tasking (SPSPM), addressing employees handling multiple tasks with varying skills and duration influenced by skill characteristics. However, the approach needs to incorporate more realistic factors in software project management, such as uncertainty, resource leveling, and preemption. Tian [30] introduces an innovative strategy for the multi-skill resource-constrained project scheduling problem, considering skill switches impacting time and cost. An Evolution Strategy framework with multi-objective optimization (MOES) addresses the influence of skill transitions on project duration and cost, guided by an improved resource-leveling operator for new individuals with reduced duration. However, the paper overlooks varying resource proficiencies, resulting in a limitation where task processing time is not dynamically adjusted when assigning different resources.

A meta-heuristic form of Genetic Algorithm (GA) [37], [38] has been deliberated for addressing staffing issues in project scheduling. This approach, guided by the GA principles, enhances quality by up to 17% when compared to cost-based strategies, primarily by prioritizing the assignment of tasks to the most experienced resources. Furthermore, sallam et al. [39] investigate knowledge-based Evolutionary Algorithms (EAs) to intelligently assign the most effective set of resources to project activities. These EA algorithms contribute to the early-stage development of an optimal schedule, benefiting project management endeavors. Dang et al. [27] present M-PSO, a novel particle swarm optimization technique. To enhance exploration, the algorithm incorporates a migration method to escape local optima and broaden the search space. However, the aforementioned strategies work well only for large-scale projects.

In [24], a multi-objective discrete Jaya algorithm is presented for minimizing makespan and cost in mixed-integer linear programming problems. The algorithm employs encoding, decoding techniques, and resource assignment criteria to enhance solution diversity and resource utilization. In [23], a dynamic optimization model integrates MSRCPSP with uncertainty in resource availability, utilizing a Branch-and-Bound algorithm and a hybrid Benders decomposition algorithm to minimize makespan and total cost for multi-skilled resources. It considers uncertainty in resource availability, aiming for a robust schedule to minimize schedule instability costs. Nonetheless, the model assumes homogeneous skill levels and does not account for the costs associated with switching resources' skills.

To address the intricacies posed by MSRCPSPs, studies in [40], [41] offer multi-objective algorithms with the main objective of minimizing project makespan and costs. These proposed solutions exhibit optimal performance across a spectrum of problem sizes, spanning small to significant scales. Notably, a breadth-first search (BFS) algorithm has been employed to determine the optimal mode selection for each resource concerning the execution of specific project tasks. Nevertheless, it is essential to underscore that these BFS-based algorithms do not factor in skill assessment based on resource capabilities. The prevailing corpus of literature collectively underscores the central aim of scheduling algorithms to minimize project makespan. However, the assignment of resource weights based on resource proficiency levels is a facet largely overlooked in the majority of previous works.

**TABLE 1.** Symbols and Notations

| Notation | Definition |
|---|---|
| GPS | Greedy and Parallel Scheduling |
| PSS | Parallel Scheduling Scheme |
| TPG | Task Precedence Graph |
| MSRCPSP | Multi-Skill Resource Constrained Project Scheduling Problem |
| $T = \{t_0, t_1, \ldots, t_n, t_{n+1}\}$ | Set of project tasks |
| $S = \{s_1, s_2, \ldots, s_l\}$ | Set of skills |
| $R = \{r_1, r_2, \ldots, r_k\}$ | Set of all resources |
| $T_{\shortparallel}$ | Set of parallel executable tasks |
| $V$ | Set of vertices in TPG |
| $E$ | Set of edges in TPG |
| $e_{t_i}$ | Edge of $t_i$ |
| $S_{t_i}$ | Set of skills required to perform $t_i$ |
| $S^{r_k}$ | Set of skills mastered by $r_k$ |
| $S_{wsf}$ | Set of skills sorted as worst skills first |
| $eff_{t_i}$ | Total effort required to perform $t_i$ |
| $eff^*_{t_i s_l}$ | Effort done against $s_l$ of $t_i$ |
| $d_{t_i}$ | Duration of $t_i$ |
| $C_{t_i}$ | Time already spent on $t_i$ |
| $C_{T_{\shortparallel}}$ | Sum of time spent on tasks in $T_{\shortparallel}$ |
| $R^*_{t_i}$ | Resources assigned to $t_i$ |
| $R^*$ | Set of assigned resources |
| $R^{**}_{t_i}$ | Set of surplus resources assigned to $t_i$ |
| $LF_{t_i s_l}$ | Latest finishing $s_l$ in $t_i$ |
| $R_{avail}$ | Set of available resources |
| $w_{r_k s_l}$ | Weight or proficiency of $r_k$ against $s_l$ |
| $\sum_{w_{r_k}}^{\|R\|} r_k t_i s_l$ | Sum of proficiency values of resources $r_k$ assigned to $t_i$ against $s_l$ |
| $B_{t_i}$ | Starting time of $t_i$ |
| $F_{t_i}$ | Finishing time of $t_i$ |
| $t_{ef}$ | Earliest finishing task |
| $t_{lf}$ | Latest finishing task |
| $F_{t_i s_l}$ | Finishing time of $t_i$ with respect to $s_l$ |
| $LV(e_{t_i})$ | Function returning left vertex of $e_{t_i}$ |
| $RV(e_{t_i})$ | Function returning right vertex of $e_{t_i}$ |
| $TS$ | Task skills matrix |
| $t_i s_j = 1$ | Skill $s_j$ is required to perform $t_i$ |
| $RS$ | Resource skills Matrix |
| $r_k s_j = [0, 1]$ | $r_k$ has $s_j$ with weight [0,1] |

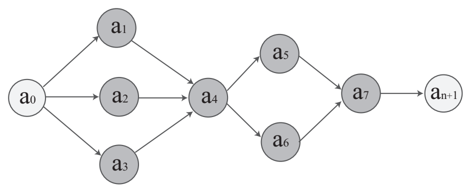

FIGURE 1. Directed-acyclic graph showing task precedence relationship

## III. PROBLEM FORMULATION

Consider a software project $T$ that consists of $n$ tasks i-e $T = \{t_0, t_1, t_2, ....., t_i, t_j, \dots, t_n, t_{n+1}\}$ where $t_i$ denotes the $i^{th}$ task. While $t_0$ and $t_{n+1}$ are dummy tasks representing the start and end of the project, respectively. The tuple defines each project task $(eff_{t_i}, d_{t_i})$ where $eff_{t_i}$ is the effort required to complete $t_i$ and $d_{t_i}$ represents the deadline for $t_i$. The task precedence graph $(TPG)$ shows the precedence constraints among the project tasks as depicted in Fig. 1. A $TPG$ is defined by $TPG = (V, E)$ where $V$ is the set of tasks, i.e. $V = \{t_0, t_1, t_2, ....., t_i, t_j, \dots, t_n, t_{n+1}\}$ and $|V| = n + 2$ while $E$ is the set of directed edges used to represent the precedence constraints among the tasks. The set $E$ can be defined as: $E = \{\ (a, b) \mid a, b \in T \wedge (a \to b)\ \}$, where $a \to b$ defines the dependency of $b$ on $a$; task $a$ must be executed ahead of $b$. Each project task $t_i$ requires several resources having specific skills. These skills are defined by the set $S = \{s_1, s_2, ....., s_l\}$. A task-skill matrix $(TS)$ is used to represent skills required to perform each task, such as:

$$TS = \begin{bmatrix} t_1s_1 & t_1s_2 & \dots & t_1s_m \\ t_2s_1 & t_2s_2 & \dots & t_2s_m \\ \vdots & \vdots & \vdots & \vdots \\ t_ns_1 & t_ns_2 & \dots & t_ns_m \end{bmatrix}$$

where $t_is_j = 1$ means that $i^{th}$ skill is required to perform the $j^{th}$ task. To perform the project tasks, we have a set of resources defined by $R = \{r_1, r_2, \dots, r_k\}$ while $r_k$ denotes the total number of resources used in the project. In this set of resources, each resource masters one or more skills. The proficiency level of each resource is defined by the resource-skill $(RS)$ matrix as given below:

$$RS = \begin{bmatrix} r_1s_1 & r_1s_2 & \dots & r_1s_m \\ r_2s_1 & r_2s_2 & \dots & r_2s_m \\ \vdots & \vdots & \vdots & \vdots \\ r_ks_1 & r_ks_2 & \dots & r_ks_m \end{bmatrix}$$

where $r_ksj = [0, 1]$ means that resource $k$ possesses the skill $j$ with a proficiency score $[0, 1]$.

In this scheduling problem, each resource $r_k \in R$ has one or several skills denoted by $S^{r_k}$. Similarly, each project task $t_i \in T$ requires a set of skills to be executed and denoted by $S_{t_i}$. While $R_{t_is_l}$ is the number of resources needed to perform a skill $s_l$ required by task $t_i$. Considering this setup, the scheduling problem is defined as follows: Given a project defined by the set T and resources defined by the set $R$, we must assign the project tasks to resources in such a way that the overall completion time of the project is minimized while ensuring that no precedence constraint is violated. Non-negative decision variables check the resources and skills demanded to perform their corresponding activities. This way, through decision variables assignment of resources to project tasks is checked with respective skill capability. In the case of W-MSRCPSP, we have considered the following decision variables.

$$w_{r_ks_lt_i} = \begin{cases} 1, & if\ r_k\ with\ s_l\ is\ assigned\ to\ t_i \\ 0, & otherwise \end{cases}$$

while $r_k \in R, t_i \in T, s_l \in S_{t_i} \cap S^{r_k}$

Based on the above decision variable, the W-MSRCPSP can be formulated as follows. The basic notation for the model is provided in Table 2, Appendix. The objective function is defined as:

$$minimize\ \ (B_{t_{n+1}}) \tag{1}$$

subject to:

$$B_{t_j} \geq B_{t_i} + d_{t_i},\ \ t_i, t_j \in T \wedge (t_i, t_j) \in E \tag{2}$$

$$\sum_{r_k \in R_{s_l}} w_{r_ks_lt_i} = R_{t_is_l}, t_i \in T, s_l \in S_{t_i} \tag{3}$$

$$\sum_{s_l \in S^{r_k} \cap S_{t_i}} w_{r_ks_lt_i} \leq 1, r_k \in R, t_i \in T \tag{4}$$

$$\sum_{s_l=1}^{|S|} t_is_l > 0, \forall t_i \in T \wedge s_l \in S_{t_i} \tag{5}$$

$$\sum_{r_k=1}^{|R|} t_ir_k > 0, \forall t_i \in T \wedge r_k \in R^{r_k} \tag{6}$$

$$B_{t_i} \geq 0, d_{t_i} \geq 0, \forall t_i \in T \tag{7}$$

$$R_{t_is_l} \subseteq \bigcup_{t_ir_k > 0} S_{t_i}, \forall t_i \in T \cap s_l \in S_{t_i} \tag{8}$$

$$F_{t_i} = B_{t_i} + d_{t_i}, \forall t_i \in T \tag{9}$$

The goal of the proposed strategy is to minimize the overall project duration, as depicted in Equation (1). Here, $B_{t_{n+1}}$ defines the starting time of the task $(n + 1)$. Since, the task $(n + 1)$ marks the end of the project and is a dummy node, $B_{t_{n+1}}$ denotes the end of the project. The objective function defined in Equation (1) is subject to 8 constraints depicted in Equations (2-11). The constraint in equation (2) defines the fulfillment of the precedence constraints – the predecessors $t_i$ of a selected task $t_j$ must be completed before starting $t_j$.

According to constraint (3), a resource $r_k$ can be assigned to a task $t_i$ with required skill $s_l$ if and only if resource $r_i$ possesses the skill $s_l$. Similarly, constraint (4) states that a resource can be assigned to only one task at a time and can perform at most one skill for any project task.

Equation (5) delineates that at least one skill is required to perform each task. Similarly, equation (6) specifies that at least one resource must perform each project task. Constraint (7) depicts the starting time $B_{t_i}$ and duration $d_{t_i}$ of any project task $t_i$ cannot be negative. According to (8), resources allocated to the task $t_i$ denoted by $R_{t_i s_l}$ must have the skills required to perform $t_i$ in order to execute the task. Equation (9) states that the finishing time of any task $t_i$ must be equal to the sum of its starting time $B_{t_i}$ and duration $d_{t_i}$.

## IV. PROPOSED GPS ALGORITHM

In this section, we introduce GPS, an innovative hybrid scheduling approach that combines best-fit and parallel scheduling strategies. The primary objective is to effectively allocate optimal resources to project tasks at each scheduling point. The proposed approach functions through a series of iterations, the total number of which aligns with the overall task count in the project. Tasks capable of simultaneous execution are grouped into clusters or sets based on their precedence relationships.

The algorithm commences by addressing the initial cluster of parallel executable tasks, allocating resources to the tasks within that cluster. Task prioritization occurs through arranging tasks in ascending order of their estimated execution times, ensuring precedence for tasks with shorter duration to minimize the waiting time of project tasks. Following this, the proposed methodology assigns resources to the chosen task based on its specific skill requirements using a best-fit strategy. The proposed approach allocates available resources based on their weights to project tasks against each skill required by the project tasks. Using a best-fit strategy, the proposed approach takes a resource with the highest weight and assigns it greedily to the earliest finishing task first if the required skill matches the skills of the selected resource. Allocation continues until the task is completed within its stipulated duration.

Following resource assignment to the earliest finishing task, the next earliest finishing task among the unscheduled ones is selected within the current group, and resources are allocated from the available resource pool. This process ensures the allocation of all available resources to unscheduled tasks in a given group. In contrast to existing strategies, if any surplus resources are available, they are assigned to the latest finishing task among the scheduled tasks to minimize the overall makespan. Upon task completion, the assigned resources are reallocated to another task within the same group. When all tasks in a group are completed, the subsequent set of parallel executable tasks is selected, and the same procedure is repeated to execute the tasks in the selected group.

### A. TASK PRECEDENCE GRAPH

The proposed scheduling technique generates a task precedence graph (TPG), where several tasks are linked through precedence relations. A project graph is defined by $TPG = (V, E)$ where $V$ is the set of tasks, i.e. $V = \{t_1, t_2, ....., t_i, t_j, \dots, t_n\}$ and $|V| = n$ while $E$ is the set of edges used to represent the precedence constraints among the tasks. The set $E$ can be defined as $E = \{(a, b) | a, b \in P \wedge (a \rightarrow b)\}$, where $a \rightarrow b$ means that task $a$ must be executed before $b$.

### B. TASK SUBSETS

In the context of TPG, as mentioned earlier, certain project tasks can be executed concurrently if they do not violate their precedence relationships. The proposed GPS design involves partitioning the task set $T$ into $q$ subsets, indexed by $i$. Tasks within each subset $T_i$ have the potential for parallel execution depending on the availability of adequate resources. Any given task is exclusively associated with a specific subset $T_i$ – the intersection between any two subsets of $T$ is null. The conditions governing this partitioning mechanism are formally expressed through the following mathematical formulation:

$$|T| = |T_1| + |T_2| + \cdots + |T_i| + \cdots + |T_q| = n$$

while $T_{\shortparallel} = \{T_1, T_2, \dots, t_m\}, T_i \cap T_j = \emptyset \; \forall i, j \in$ subsets of $T$. For a subset $T_i$, there should be no precedence constraint among any pair of tasks $t_j$ and $t_k$ where $t_k, t_l \in T_i$. To form the subsets of parallel executable tasks, the proposed approach iteratively examines each vertex $V_{t_k} \in V$ to check whether it is dependent on any other vertex in $V$. A vertex $V_{t_k}$ is considered an independent vertex if $\forall (a, b) \in E \wedge b \neq V_{t_k}$. At the end of the iteration, all the vertices that are not dependent on any vertex in $V$ are grouped to form a subset $T_i$ and are removed from $V$. The algorithm repeats the procedure to create groups of parallel executable tasks until $V$ becomes empty.

### C. RESOURCE WEIGHT

Within software projects, human resources are often equipped with a diverse set of skills, enabling them to undertake various project tasks. In the proposed methodology, we take into account the presence of multi-skilled and heterogeneous human resources as a foundational element for crafting an optimal schedule. These resources exhibit varying proficiency levels across different skills. To maximize the effective utilization of human resources for project tasks, we employ a system where each resource is assigned weights corresponding to their proficiency in each skill they possess.

A realistic weighting mechanism is utilized where a skill proficiency score (within the [0,1] range) is attributed to each resource against each skill they possess. The skill proficiency score of a resource reflects its competence in utilizing the corresponding skill. Consequently, each human resource possesses a range of multiple skills, each accompanied by a

**Algorithm 1:** GPS

```
Input: $T, S, R, E$
Output: project_makespan
1  $makespan \leftarrow 0$
2  $T_{\parallel}[\,] \leftarrow$ CreateParallelTasksGroups$(T, E)$
3  foreach $T_{\parallel} \in T_{\parallel}[\,]$ do
4      $R_{avail} \leftarrow R$
5      if $T_{\parallel}$ contains $(t_0 \mid t_{n+1})$ then
6          continue
7      end
8      sort$(T_{\parallel})$
9      $makespan\ += $ ScheduleTasksSubset$(T_{\parallel}, R_{avail})$
10 end
11 return $makespan$
```

**Algorithm 2:** CreateParallelTasksGroups

```
Input: $T, E$
Output: $T_{\parallel}[\,]$
1  $T_{\parallel}[\,] \leftarrow \emptyset$
2  while $T$ is not empty do
3      $T_{\parallel} \leftarrow \emptyset$
4      foreach $t_i \in T$ do
5          $parallel\_vertix \leftarrow true$
6          foreach $e_{t_i} \in E$ do
7              if $RV(e_{t_i})$ is $t_i$ then
8                  $parallel\_vertix \leftarrow false$
9                  break
10             end
11         end
12         if $parallel\_vertix == true$ then
13             $T_{\parallel} \leftarrow T_{\parallel} + t_i$
14         end
15     end
16     foreach $t_i \in T_{\parallel}$ do
17         $T \leftarrow T - t_i$
18         foreach $e_{t_i} \in E$ do
19             if $LV(e_{t_i})$ is $t_i$ then
20                 $E \leftarrow E - e_{t_i}$
21             end
22         end
23     end
24     $T_{\parallel}[\,] \leftarrow T_{\parallel}[\,] + T_{\parallel}$
25 end
26 return $T_{\parallel}[\,]$
```

distinct proficiency score. For instance, a resource assigned a weight of 0.8 for a specific skill signifies that this resource can execute 80% of the associated work within a standard working day. Conversely, a highly proficient resource, characterized by a weight of 1, is adept at accomplishing 100% of the work associated with a skill within the same time-frame.

### D. PROPOSED SCHEDULING TECHNIQUE

To enhance readability and comprehension, we break down the proposed scheduling Algorithm into five sub-algorithms. Algorithm 1 serves as the starting point for the proposed strategy. It takes the set of project tasks $T$, the required skill set $S$, the available resources $R$, and the set of edges $E$, specifying the precedence constraints among the tasks in $T$ as input. The algorithm returns the resulting makespan as its output.

Algorithm 1 starts by initializing the project makespan, denoted by variable $makespan$, to 0. Next, it calls the Algorithm 2 to obtain an array of parallel executable tasks $T_{\parallel}$. $T_{\parallel}$ signifies a collection of tasks that are executable simultaneously if sufficient resources are available. The algorithm then schedules each set of parallel executable tasks by looping through the array (see Line 3 - 10). The algorithm invokes Algorithm 3 at Line 9 to schedule the tasks in selected $T_{\parallel}$. Note that Algorithm 3 returns the time spent on completing the tasks in $T_{\parallel}$ after completing the scheduling process. At Line 9, the Algorithm 1 takes the returned value and adds it to the total makespan of the project. However, before calling the 3, it sorts the tasks in $T_{\parallel}$ based on their finishing time – earliest finishing tasks are scheduled first. Also, if $T_{\parallel}$ contains the dummy tasks, representing the start and end of the project, they are neglected by moving on to the next element in $T_{||}[\,]$ (refer to Line 5-7). Finally, the main algorithm returns the total makespan after scheduling all the project tasks (at Line 11).

Algorithm 2 formalizes the grouping of parallel executable tasks. It takes the set of project tasks $T$ and edges $E$ as input and returns an array of parallel executable tasks. A group of parallel executable tasks, denoted by $T_{\parallel}$, consists of all the tasks that can be executed at the same time if enough resources are available. The algorithm starts by creating an empty array of $T_{\parallel}$. It iterates through the elements of $T$ (until there are no more tasks left in $T$) to find and group tasks that can be run in parallel (From Line 2-25). In each iteration, it takes each element in $T$ denoted by $t_i$ at Line 4, and checks whether a $t_i$ represents the right vertex of any edge $e$ in $E$ is $t_i$. If yes, it means $t_i$ depends on some other task in $T$ and therefore cannot be run in parallel (see Lines 6-9). Otherwise, $t_i$ can be added to the set of parallel executable tasks $T_{\parallel}$. This process is repeated until all the elements in $T$ are traversed.

After executing Line 15, the algorithm finds a set of parallel executable tasks in $T$. Next, the algorithm needs to add the $T_{\parallel}$ to the array of parallel executable tasks and remove all the edges from $E$ whose left vertex is in $T_{\parallel}$ (refer to Line 16-23). Finally, the algorithm adds $T_{\parallel}$ to the array of $T_{\parallel}$. It repeats this process until all the elements are grouped, and there is no task left in $T$. Finally, it returns the array of $T_{\parallel}$ at Line 26.

Algorithm 3 outlines the steps followed to schedule all tasks in $T_{\parallel}$. It takes $T_{\parallel}$ and the set of available resources $R_{avail}$ as input, and returns the time $C_{T_{\parallel}}$ spent on completing

**Algorithm 3:** ScheduleTasksSubset

```
Input: $T_{\parallel} \in T_{\parallel}[\,], R_{avail} \in R$
Output: $C_{T_{\parallel}}$
1  $C_{T_{\parallel}} \leftarrow 0$
2  while $T_{\parallel} > 0$ do
3      foreach $t_i \in T_{\parallel}$ do
4          if $|R_{t_i}*| == 0$ or $F_{t_i} > d_{t_i}$ then
5              AssignResources$(t_i, R_{avail})$
6          end
7      end
8      AssignExtraResources$(T_{\parallel}, R_{avail})$
9      $t_{ef} \leftarrow null$
10     foreach $t_i \in T_{\parallel}$ do
11         $F_{t_i} \leftarrow max(F_{t_i s_l})$
12         if $t_{ef}$ is null OR $F_{t_i} < F_{t_{ef}}$ then
13             $t_{ef} \leftarrow t_i$
14             $F_{t_{ef}} \leftarrow F_{t_i}$
15         end
16     end
17     $R_{avail} \leftarrow R_{avail} + R_{t_{ef}}$
18     $T_{\parallel} \leftarrow T_{\parallel} - t_{ef}$
19     foreach $t_i \in T_{\parallel}$ do
20         foreach $s_l \in S_{t_i}$ do
21             $eff^*_{t_i s_l} += (F_{t_{ef}} - C_{t_i}) \times \sum_{w_{r_k}}^{|R|} r_k s_l$
22         end
23         $C_{t_i} \leftarrow F_{t_{ef}}$
24     end
25     $C_{T_{\parallel}} \leftarrow F_{t_{ef}}$
26 end
27 return $C_{T_{\parallel}}$
```

**Algorithm 4:** AssignResources

```
Input: $t_i \in T, R_{avail} \in R$
Output: $Boolean[true|false]$
1  foreach $s_l \in S_{t_i}$ do
2      $eff^{prev}_{t_i s_l} \leftarrow eff^*_{t_i s_l}$
3      while $|R_{avail}| > 0$ do
4          $r \leftarrow findMaxW_{r_k}(R_{avail}, s_l)$
5          if $r$ is null then
6              $R_{avail} = R_{avail} + R^*_{t_i}$
7              return false
8          end
9          $R^*_{t_i} \leftarrow R^*_{t_i} + r$
10         $R_{avail} \leftarrow R_{avail} - r$
11         $F_{t_i s_l} \leftarrow C_{t_i s_l} + \frac{eff^*_{t_i s_l} - eff^{prev}_{t_i s_l}}{\sum_{W_{r_k}}^{|R|} r_k t_i s_l}$
12         if $F_{t_i s_l} \leq d_{t_i}$ then
13             break
14         end
15     end
16 end
17 return true
```

**Algorithm 5:** AssignExtraResources

```
Input: $T_{\parallel} \in T_{\parallel}, R_{avail} \in R$
Output: Assignment of free resources to the tasks in
        $T_{\parallel}$
1  $T_{temp} \leftarrow T_{\parallel}$
2  foreach $t_i \in T_{temp}$ do
3      if $|R^*_{ti}| < 1$ then
4          $T_{temp} \leftarrow T_{temp} - t_i$
5      end
6  end
7  Top_Loop: while $|R_{avail}| > 0 \wedge |T_{temp}| > 0$ do
8      $t_{lf} \leftarrow getLatestFinishingTask(T_{temp})$
9      $s_{lf} \leftarrow getLatestFinishingSkill(t_i)$
10     while $r_k \in R_{avail}$ do
11         if $s_{lf} \in S^{r_k}$ then
12             $R^{**}_{t_{lf}} \leftarrow R^{**}_{t_{lf}} + r_k$
13             $R_{avail} \leftarrow R_{avail} - r_k$
14             continue Top_Loop
15         end
16     end
17     $T_{temp} = T_{temp} - t_i$
18 end
```

tasks in $T_{\parallel}$. It assigns resources to each task $t_i$ in $T_{\parallel}$ until there is no more task left. It does so by calling the Algorithm 4 (at Line 5) if the task has not already been scheduled and the finishing time is not less than the deadline (duration). Otherwise, it moves on to the next task and repeats the process from Lines 3-7. After assigning enough resources to each task, there might be some free resources available. The algorithm tries to assign surplus resources to scheduled tasks in $T_{\parallel}$ to minimize the makespan. To do so, it invokes Algorithm 5. Next, it finds the earliest finishing task (refer to Line 9-16).

At this point, the algorithm needs to mark the earliest finishing task completed and remove it from the set of parallel executable tasks (Lines 17 and 18). Finally, it has to calculate the effort done against each task and record the time taken $C_{t_i}$ by each $t_i$ (see Line 24). However, it is very important to note that the effort is calculated against each required skill as the finishing time is calculated based on the latest finishing skill of a task. This is done from lines 20-22. Finally, the algorithm records the time taken by all the tasks in $T_{\parallel}$ denoted by $C_{T_{\parallel}}$ at Line 25, and returns the recorded value at Line 27 after completing all the tasks by repeating the same process (From Line 2-26) until there is no element left in $T_{\parallel}$.

Algorithm 4 is used to assign available resources to a particular task. It takes a task $t_i$ and a set of available resources $R_{avail}$ as input and returns a boolean value indicating

**TABLE 2.** Experimental Set

| Experiment | Number Of Tasks | Number Of Skills | Number Of Resources | (Predecessor, Successor) | Max. Resources per Skill | Max. Resources per Task |
|---|---|---|---|---|---|---|
| 1 | 1-15 | 1-4 | 1-20 | 1-4 | 1-15 | 1-4 |
| 2 | 1-40 | 4 | 15 | 3 | 10 | 4 |
| 3 | 15 | 1-4 | 15 | 3 | 10 | 4 |
| 4 | 15 | 4 | 1-20 | 3 | 10 | 4 |
| 5 | 15 | 4 | 15 | 1-4 | 10 | 4 |
| 6 | 15 | 4 | 15 | 3 | 1-15 | 4 |
| 7 | 15 | 4 | 15 | 3 | 10 | 1-4 |

whether the assignment was successful or not. The algorithm assigns resources against each skill $s_l$ required by $t_i$. The main function of this algorithm is to assign enough resources so that the task can be finished within its deadline. Moreover, it assigns the best available resources against each skill by finding resources having the required skill with maximum proficiency.

In line 1, it takes a skill $s_l$ from required skills $S_{t_i}$. It finds the best available resource $r$ having the skill $s_l$ with the highest proficiency at Line 4. If $r$ is null, it indicates there is no resource available with the required skill. So, in line 5, it checks if $r$ is null and there is no resource assigned to $t_i$ against the required skill, then it needs to free the resources assigned to $t_i$ and return false, indicating unsuccessful assignment. Otherwise, it assigns $r$ to $t_i$, removes $r$ from $R_{avail}$, and calculates the finishing time of $t_i$ with respect to skill $s_l$ (refer to Line 9-11). If the finishing time is less than or equal to the deadline of the task, it moves on to the next required skill $s_l \in S_{t_i}$. Otherwise, it repeats the procedure from Line 4 to 15 to assign more resources until the finishing time meets the deadline constraint if the set of available resources $R_{avail}$ is not empty (see Line 3).

Algorithm 5 depicts the procedure for assigning surplus resources (if any) to scheduled activities to minimize the total makespan and reduce resource wastage. The idea is to assign more resources to the latest finishing task so that the total makespan can be minimized. The algorithm takes $T_{||}$ and $R_{avail}$ as input. The algorithm starts by initializing a variable $T_{temp}$ to $T_{||}$. Next, it removes unscheduled tasks from $T_{temp}$ (see lines 2-6). It does so by checking if a task $t_i$ in $T_{temp}$ has assigned resources in the previous step. If no resources have been assigned to $t_i$ in the previous step, it means the task has not been scheduled yet due to a lack of available resources for $t_i$. After removing unscheduled

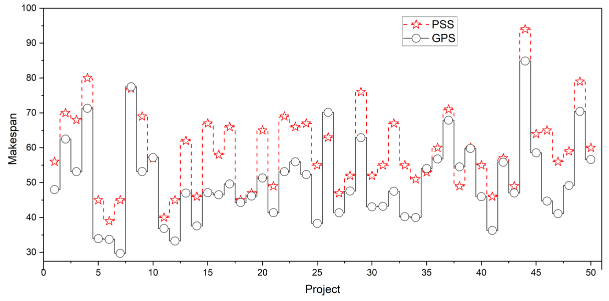

**FIGURE 2.** Experiment 1 (a): Comparison in terms of project makespan

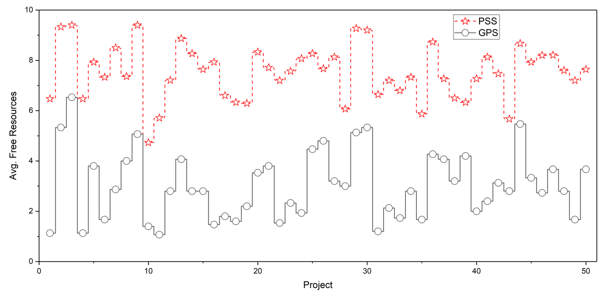

**FIGURE 3.** Experiment 1 (b): Comparison in terms of resource wastage

tasks, it tries to assign resources to the latest finishing task in $T_{temp}$. Moreover, it assigns resources against the latest finishing skill of the latest finishing task to maximize the benefits of resource assignment. Hence, in lines 8 and 9, it selects the latest finishing task and then the latest finishing skill of the task, respectively.

The resource allocation is done from lines 10-16, where it takes each resource from available resources and checks whether the resources have the required skill. If no, it moves on to the next resource. If yes, it assigns the resource to $T_i$ against $s_l$ and removes the resource from the set of available resources, and jumps to the Top_Loop. If no resource assignment occurs, it executes Line 17, indicating a lack of resources to be assigned to $t_i$ against any required skill. The algorithm follows the same procedure (from lines 7-17) for other tasks in $T_{temp}$ if there are any and the set of available resources is not empty.

## V. SIMULATION RESULTS AND DISCUSSION

For comparison, we have specifically chosen to compare our proposed GPS algorithm with the PSS heuristic due to its relevance and established use in the field of project management. The PSS heuristic serves as an appropriate benchmark, allowing us to highlight the distinct advantages and improvements our GPS algorithm offers, particularly in terms of resource optimization and project makespan efficiency, which are critical in non-real-time project management scenarios.

In order to evaluate the proposed technique against the PSS heuristic [13], we conduct 7 different experiments. Table 2 depicts the experimental setup for this study. We have

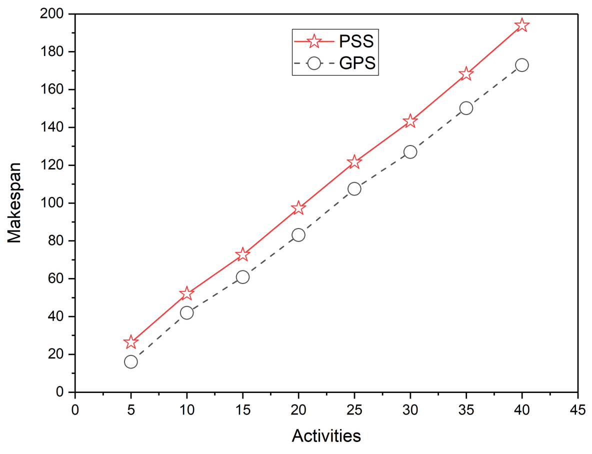

**FIGURE 4.** Experiment 2: Impact of number of tasks on project makespan

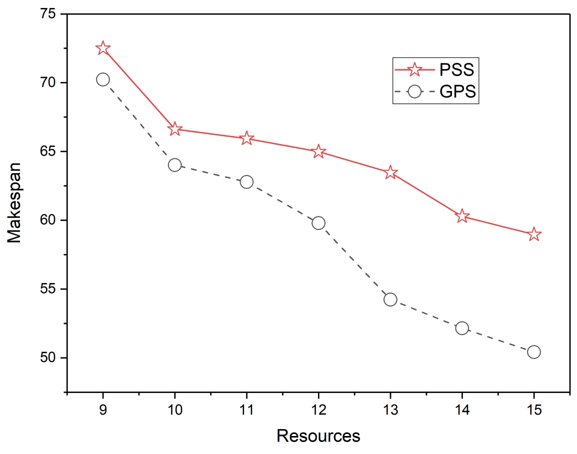

**FIGURE 6.** Experiment 4: Impact of number of available resources

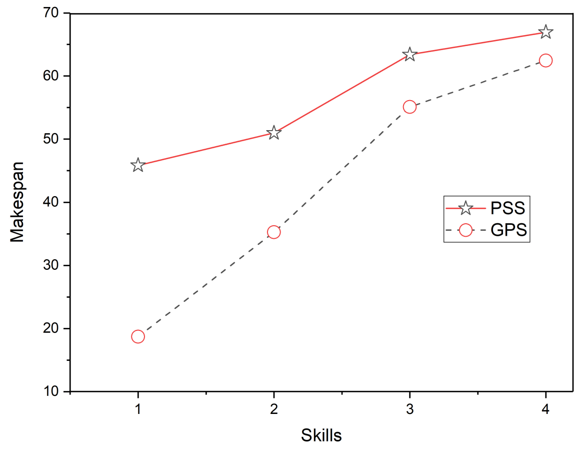

**FIGURE 5.** Experiment 3: Impact of Number of Skills Required

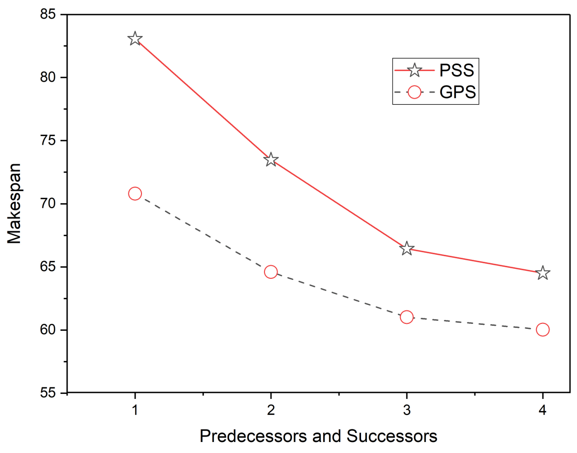

**FIGURE 7.** Experiment 5: Impact of graph complexity

considered the following six different scheduling parameters; number of tasks, number of skills required, resources, complexity of the topology graph in terms of predecessors and successors of the tasks, task duration, and resources against each skill required for a project task to be performed. Each experiment consists of 50 projects. The first experiment evaluates the proposed approach against its counterpart in 50 random projects where the values of the six parameters considered are randomly chosen in the specified range (see Table 2). The remaining 6 experiments are conducted to show the impact of different factors, such as varying the number of project tasks and available resources, on the efficacy of the proposed approach compared to the PSS heuristic. For experiments 2 to 7, we change the value of one of the scheduling parameters while keeping the other parameters constant.

Experiment 1 is set up to evaluate the performance of the proposed technique against the PSS in 50 random projects in terms of the total makespan of the projects. In this experiment, we choose random values for all the factors within their specified range, as depicted in Table 2. Compared to the PSS heuristic, the proposed GPS design optimizes the usage of the available resources and significantly reduces the project makespan, as shown in Fig. 2 and Fig. 3. In the set of 50 randomly generated projects, the PSS heuristic completes the projects with an average makespan of 58.96 days. However, the proposed GPS design completes the projects with an average project makespan of 50.42 days. The average completion gap to complete the project makespan for our technique is 9.80 days compared to the PSS heuristic, which is only 2.88 days. Many resources remain free in the PSS heuristic compared to our technique. This indicates that the GPS strategy better utilizes available resources, as shown in Fig. 3. This is due to the fact that the GPS design assigns surplus resources, if available, to the latest finishing tasks. Also, it assigns the newly freed resources from the earliest finishing tasks to the tasks that minimize the overall makespan.

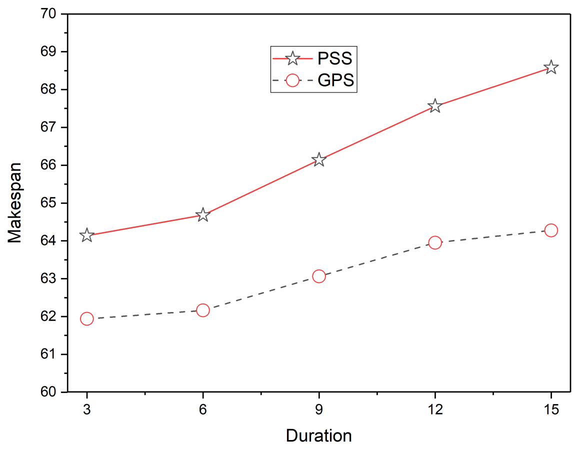

**FIGURE 8.** Experiment 6: Impact of duration of project tasks

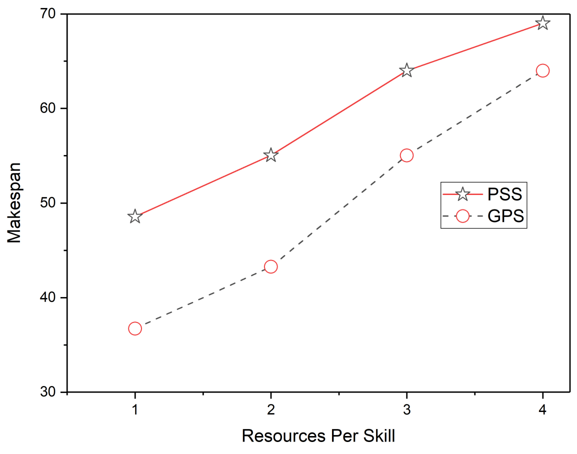

**FIGURE 9.** Experiment 7: Impact of required resources per task

In experiment set 2, we study the impact of a number of project tasks on the appropriateness of both the GPS and the PSS heuristic. We increase the number of tasks from 5 to 40 while keeping all the other factors constant. It is observed that the proposed strategy performs better than the PSS heuristic. However, there is no significant impact of the number of project tasks on the efficacy of both strategies, as depicted in Fig.4. Fig. 4 clearly shows that the project makespan increases for both the proposed and the PSS heuristic in almost similar fashion as the number of project tasks increases.

In the third experiment, we varied the number of skills required to perform each task to complete the project within time. The value for required skills against each task range from 1 to 4. In the case of our technique, the average project makespan is considerably lower than the PSS when the skill demand for project tasks is lower. As depicted in Fig. 5, when we increase the number of skills required to perform each task, the average makespan gap between the two strategies tends to reduce due to the fact that there are no surplus resources available to minimize the project makespan. In other words, for a smaller number of skill requirements, most project resources remain free in the set of available resources. The proposed technique gets maximum benefit from these free resources to optimize the project makespan. As the number of skills increases, the resource requirement also increases to perform these skills. So, due to the less availability of resources, the proposed technique has no extra resources to utilize to optimize the project makespan; hence, the difference in the average makespan decreases. However, overall, the proposed strategy outperforms the PSS heuristic even if there are no extra resources available.

In experiment 4, we increase the number of available resources, and the project's average makespan decreases in our proposed GPS design since it utilizes all available resources whenever they are free. Also, the proposal considers parallel executable tasks to optimize project makespan. While the PSS heuristic only assigns enough resources to a task against each required skill to finish it within its deadline and does not consider assigning extra available resources to optimize the project makespan. Fig. 6 depicts that the PSS heuristic is less effective in optimizing the project makespan compared to the proposed technique when the number of resources is increased as it assigns only the required number of resources to the project tasks and tries to optimize the project makespan by leveraging the concept of parallel executable tasks only. In contrast, the proposed approach takes benefits from the concept of parallel executable tasks as well as the utilization of surplus resources if available.

In experiment 5, we study the impact of network graph complexity on the appropriateness of both the GPS and the PSS heuristic. We increase the number of predecessor and successor tasks in the graph from 1 to 4. The proposed resource allocation strategy performs significantly better than the PSS heuristic when the complexity of the network graph is lower. Fig. 7 shows that the project makespan tends to decrease between the two strategies when we increase the number of predecessors and successors.

In experiment 6, the case of project task duration is considered. The proposed technique attains promising results with a minimum task duration compared to the PSS heuristic. When we increase the task duration, the project makespan gap increases between the two techniques, as shown in Fig. 7. This is because when the task duration range is high (for instance, 1-15), the proposed GPS strategy can take advantage of resources freed from earliest finishing tasks and use them to optimize the project makespan by assigning them to other tasks running in parallel.

In the last experiment, we study the impact of resource requirements against each skill required to perform a project task. As depicted in Fig. 8, the proposed GPS strategy performs significantly better than the PSS heuristic. However, with the increase in the number of resources required for each skill to perform a task, the gap between the makespan due to the GPS and PSS strategies tends to decrease.

## VI. CONCLUSION

In this paper, we study the Weighted Multi-Skill Resource Constrained Project Scheduling Problem (W-MSRCSPSP) in order to minimize the project's overall makespan. Unlike previous research, we devise a novel greedy and parallel scheduling approach that considers heterogeneous resources with different skill proficiency levels. After conducting extensive experiments, it is found that the proposed GPS technique is more effective in almost all project settings than the PSS heuristic. Six different parameters are considered during resource allocation to assess the efficacy of the proposed strategy compared to its counterpart.

In the first experiment with 50 random projects, the proposed GPS strategy outperforms the PSS heuristic in terms of project makespan. To further evaluate the effectiveness of the GPS technique in handling changes in the six factors, six additional experiments are conducted. In each experiment, the value of one of the six factors is increased while keeping the other five factors constant, and the performance of both the GPS and PSS heuristics is observed.

It is observed that when the number of project tasks is increased, there is no significant impact on the efficacy of either scheduling strategy, as the project makespan for both techniques increases in almost a similar fashion. However, when the task duration increases, the GPS strategy performs significantly better than the PSS heuristic. In addition, when the complexity of the project graph is low, the GPS strategy outperforms the PSS heuristic by a large margin. However, as the complexity of the project graph increases, the gap between the performance of the GPS and PSS techniques tends to decrease. When the minimum number of resources is required for each skill, the GPS technique performs considerably better in minimizing project makespan.

Overall, the experimental results demonstrate that the GPS technique is able to effectively assign resources and minimize project makespan, resulting in better performance compared to the PSS heuristic. In the future, the GPS technique can be further improved and optimized by considering additional factors and constraints in the resource allocation process.

## REFERENCES

[1] X. Shen, Y. Guo, and A. Li, "Cooperative coevolution with an improved resource allocation for large-scale multi-objective software project scheduling," Applied Soft Computing, vol. 88, p. 106059, 2020.

[2] N. Nigar, M. K. Shahzad, S. Islam, S. Kumar, and A. Jaleel, "Modeling human resource experience evolution for multiobjective project scheduling in large scale software projects," IEEE Access, vol. 10, pp. 44 677–44 690, 2022.

[3] M. Ghasemi, R. K. Chakrabortty, R. Shahabi-Shahmiri, and S.-A. Mirnezami, "A chance-constrained programming method with credibility measure for solving the multi-skill multi-mode resource-constrained project scheduling problem," International Journal of Construction Management, pp. 1–17, 2023.

[4] S. Akbar, I. Ahmad, R. Khan, I. O. Lopes, and R. Ullah, "Multi-skills resource constrained and personality traits based project scheduling," IEEE Access, vol. 10, pp. 131 419–131 429, 2022.

[5] H. Zhentao, C. Nanfang, H. Xuejun, and M. E. Mahaffey, "Time-and resource-based robust scheduling algorithms for multi-skilled projects," Automation in Construction, vol. 153, p. 104948, 2023.

[6] Z. Hua, Z. Liu, L. Yang, and L. Yang, "Improved genetic algorithm based on time windows decomposition for solving resource-constrained project scheduling problem," Automation in Construction, vol. 142, p. 104503, 2022.

[7] G. A. Fernandes and S. R. de Souza, "A matheuristic approach to the multi-mode resource constrained project scheduling problem," Computers & Industrial Engineering, vol. 162, p. 107592, 2021.

[8] H. Ding, C. Zhuang, and J. Liu, "Extensions of the resource-constrained project scheduling problem," Automation in Construction, vol. 153, p. 104958, 2023.

[9] J. Snauwaert and M. Vanhoucke, "A classification and new benchmark instances for the multi-skilled resource-constrained project scheduling problem," European Journal of Operational Research, vol. 307, no. 1, pp. 1–19, 2023.

[10] J. Blazewicz, J. K. Lenstra, and A. R. Kan, "Scheduling subject to resource constraints: classification and complexity," Discrete applied mathematics, vol. 5, no. 1, pp. 11–24, 1983.

[11] F. Habibi, F. Barzinpour, and S. Sadjadi, "Resource-constrained project scheduling problem: review of past and recent developments," Journal of project management, vol. 3, no. 2, pp. 55–88, 2018.

[12] A. Ghamginzadeh, A. A. Najafi, and M. Khalilzadeh, "Multi-objective multi-skill resource-constrained project scheduling problem under time uncertainty," International Journal of Fuzzy Systems, vol. 23, pp. 518–534, 2021.

[13] B. F. Almeida, I. Correia, and F. Saldanha-da Gama, "Priority-based heuristics for the multi-skill resource constrained project scheduling problem," Expert Systems with Applications, vol. 57, pp. 91–103, 2016.

[14] A. Ciupe, B. Orza, C. Florea, and A. Vlaicu, "Skill-oriented priority scheduling for solving the resource constrained project scheduling problem," in 2015 IEEE International Conference on Intelligent Computer Communication and Processing (ICCP). IEEE, 2015, pp. 85–92.

[15] J. Xiao, X.-T. Ao, and Y. Tang, "Solving software project scheduling problems with ant colony optimization," Computers & Operations Research, vol. 40, no. 1, pp. 33–46, 2013.

[16] H. Tahami and H. Fakhravar, "A literature review on combining heuristics and exact algorithms in combinatorial optimization," European Journal of Information Technologies and Computer Science, vol. 2, no. 2, pp. 6–12, 2022.

[17] S.-A. Mirnezami, R. Tavakkoli-Moghaddam, R. Shahabi-Shahmiri, and M. Ghasemi, "An integrated chance-constrained stochastic model for a preemptive multi-skilled multi-mode resource-constrained project scheduling problem: a case study of building a sports center," Engineering Applications of Artificial Intelligence, vol. 126, p. 106726, 2023.

[18] R. V. Polancos and R. R. Seva, "A risk minimization model for a multi-skilled, multi-mode resource-constrained project scheduling problem with discrete time-cost-quality-risk trade-off," Engineering Management Journal, pp. 1–17, 2023.

[19] N. Nigar, M. K. Shahzad, S. Islam, O. Oki, and J. Lukose, "Multi-objective dynamic software project scheduling: A novel approach to handle employee's addition," IEEE Access, 2023.

[20] M. Haroune, C. Dhib, E. Neron, A. Soukhal, H. Mohamed Babou, and M. F. Nanne, "Multi-project scheduling problem under shared multi-skill resource constraints," Top, vol. 31, no. 1, pp. 194–235, 2023.

[21] L. Cui, X. Liu, S. Lu, and Z. Jia, "A variable neighborhood search approach for the resource-constrained multi-project collaborative scheduling problem," Applied Soft Computing, vol. 107, p. 107480, 2021.

[22] J. L. Peng, X. Liu, C. Peng, and Y. Shao, "Multi-skill resource-constrained multi-modal project scheduling problem based on hybrid quantum algorithm," Scientific Reports, vol. 13, no. 1, p. 18502, 2023.

[23] M. Wang, G. Liu, and X. Lin, "Dynamic optimization of the multi-skilled resource-constrained project scheduling problem with uncertainty in resource availability," Mathematics, vol. 10, no. 17, p. 3070, 2022.

[24] Y.-Y. Li, J. Lin, and Z.-J. Wang, "Multi-skill resource constrained project scheduling using a multi-objective discrete jaya algorithm," Applied Intelligence, vol. 52, no. 5, pp. 5718–5738, 2022.

[25] L. Zhu, J. Lin, Y.-Y. Li, and Z.-J. Wang, "A decomposition-based multi-objective genetic programming hyper-heuristic approach for the multi-skill resource constrained project scheduling problem," Knowledge-based systems, vol. 225, p. 107099, 2021.

[26] H. Li, H. Zhu, L. Zheng, and Y. Liu, "Software project scheduling with multitasking." Economic Computation & Economic Cybernetics Studies & Research, vol. 57, no. 1, 2023.

[27] H. Dang Quoc, C. Nguyen Doan et al., "An effective hybrid algorithm based on particle swarm optimization with migration method for solving the multiskill resource-constrained project scheduling problem," Applied Computational Intelligence and Soft Computing, vol. 2022, 2022.

[28] Y. Yu, Z. Xu, D. Liu, and S. Zhao, "A two-stage approach with softmax scoring mechanism for a multi-project scheduling problem sharing multi-skilled staff," Expert Systems with Applications, vol. 203, p. 117385, 2022.

[29] Y. Ma, Z. He, N. Wang, and E. Demeulemeester, "Tabu search for proactive project scheduling problem with flexible resources," Computers & Operations Research, vol. 153, p. 106185, 2023.

[30] Y. Tian, T. Xiong, Z. Liu, Y. Mei, and L. Wan, "Multi-objective multi-skill resource-constrained project scheduling problem with skill switches: Model and evolutionary approaches," Computers & Industrial Engineering, vol. 167, p. 107897, 2022.

[31] J. C. Chen, Y.-Y. Chen, T.-L. Chen, and Y.-H. Lin, "Multi-project scheduling with multi-skilled workforce assignment considering uncertainty and learning effect for large-scale equipment manufacturer," Computers & Industrial Engineering, vol. 169, p. 108240, 2022.

[32] J. Luo, M. Vanhoucke, J. Coelho, and W. Guo, "An efficient genetic programming approach to design priority rules for resource-constrained project scheduling problem," Expert Systems with Applications, vol. 198, p. 116753, 2022.

[33] H. Chen, G. Ding, J. Zhang, R. Li, L. Jiang, and S. Qin, "A filtering genetic programming framework for stochastic resource constrained multi-project scheduling problem under new project insertions," Expert Systems with Applications, vol. 198, p. 116911, 2022.

[34] J. Chen, P. Han, Y. Zhang, T. You, and P. Zheng, "Scheduling energy consumption-constrained workflows in heterogeneous multi-processor embedded systems," Journal of Systems Architecture, vol. 142, p. 102938, 2023.

[35] J. Chen, T. Li, Y. Zhang, T. You, Y. Lu, P. Tiwari, and N. Kumar, "Global-and-local attention-based reinforcement learning for cooperative behaviour control of multiple uavs," IEEE Transactions on Vehicular Technology, 2023.

[36] F. Zarei, M. Arashpour, S.-A. Mirnezami, R. Shahabi-Shahamiri, and M. Ghasemi, "Multi-skill resource-constrained project scheduling problem considering overlapping: fuzzy multi-objective programming approach to a case study," International Journal of Construction Management, pp. 1–14, 2023.

[37] D. Seo, D. Shin, and D.-H. Bae, "Quality based software project staffing and scheduling with cost bound," in 2015 Asia-Pacific Software Engineering Conference (APSEC). IEEE, 2015, pp. 269–276.

[38] Y. Ge and B. Xu, "Dynamic staffing and rescheduling in software project management: A hybrid approach," PloS one, vol. 11, no. 6, p. e0157104, 2016.

[39] K. M. Sallam, R. K. Chakrabortty, and M. J. Ryan, "A two-stage multi-operator differential evolution algorithm for solving resource constrained project scheduling problems," Future Generation Computer Systems, vol. 108, pp. 432–444, 2020.

[40] L. Wang and X.-l. Zheng, "A knowledge-guided multi-objective fruit fly optimization algorithm for the multi-skill resource constrained project scheduling problem," Swarm and Evolutionary Computation, vol. 38, pp. 54–63, 2018.

[41] T. Zhou, Q. Long, K. M. Law, and C. Wu, "Multi-objective stochastic project scheduling with alternative execution methods: An improved quantum-behaved particle swarm optimization approach," Expert Systems with Applications, vol. 203, p. 117029, 2022.

SAEED AKBAR received his PhD degree in Computer Science from the School of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan, China in 2022. After completing his Master degree in Software Engineering from COMSATS University, Islamabad in 2017, he served as a lecturer at the Department of Computer Science, IQRA National University, Peshawar Pakistan. Currently, he is pursuing postdoc at the School of Computer Science and Technology, Zhejiang Normal University, China. His research interest includes Cloud Computing, Sustainable Computing, High Performance Computing, Dynamic Thermal-management of Data Centers, Algorithmic Game Theory, Resource Management, Deep Learning, and Computer Vision.

MUHAMMAD ZUBAIR received the bachelor's degree in software engineering from the Government College University, Faisalabad, Pakistan, in 2014, and the M.S. degree in software engineering from COMSATS University, Islamabad, Pakistan, in 2017. He is currently working as a Lecturer at the Department of Software Engineering, Lahore Garrison University, Lahore, Pakistan. He was a Lecturer at the Department of Computer Science and Information Technology, Lahore Leads University, Lahore, Pakistan. His research interests include software project scheduling, software requirement engineering, and global software engineering.

RIZWAN KHAN received his Ph.D degree in information and communication engineering from the School of Electronic Information and Communications, Huazhong University of Science and Technology, Wuhan, China. He also worked with the Wuhan National Laboratory of Optoelectronics, Wuhan. Currently, he is pursuing postdoctorate from the School of Computer Science and Mathematics, Zhejiang Normal University, Jinhua, Zhejiang China, and also working with Key Laboratory of Intelligent Education Technology and Application of Zhejiang Province, Zhejiang Normal University, Zhejiang, Jinhua, China His current research interests include computer vision, signal processing, image processing, medical image processing, healthcare applications, machine learning, and deep learning.

UBAID UL AKBAR has recently completed his Bachelor's degree in Computer Science from the Department of Computer Science, the City University of Science and Information Technology, Pakistan. His research interests include Object-Oriented Software Engineering, Visual (Block-based) Programming Languages, Software Project Scheduling, Resource Scheduling in Cloud and High-Performance Computing.

RAHMAT ULLAH is a Lecturer at the School of Computer Science and Electronic Engineering, University of Essex, United Kingdom. He previously worked as a Research & Development Associate with the Faculty of Computing, Engineering and Science, University of South Wales, U.K. Rahmat received his Master's in Software Engineering from COMSATS University, Islamabad, Pakistan, and his PhD from the School of Engineering, University of Edinburgh, U.K. His research interests include biomedical signal processing, microwave imaging, software engineering, machine learning, and digital health.

ZHONGLONG ZHENG (Member, IEEE) received the B.S. degree in electronic engineering from the China University of Petroleum, in 1999, and the Ph.D. degree from the Department of Automation, Shanghai Jiao Tong University, in 2005. He is currently a Full Professor with the College of Computer Science and Technology, Zhejiang Normal University, Jinhua, China. His research interests include machine learning, data mining, and blockchain.

• • •