IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

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

Digital Object Identifier 10.1109/ACCESS.2017.DOI

Multi-Skills Resource Constrained and Personality Traits based Project Scheduling

SAEED AKBAR[1], IFTIKHAR AHMAD [2], RIZWAN KHAN [1], IVANDRO ORTET LOPES [3], and RAHMAT ULLAH[4,*]

[1]School of Computer Science and Mathematics, Zhejiang Normal University, Jinhua 321004, China.
[2]James Watt School of Engineering, University of Glasgow, United Kingdom.
[3]Núcleo Operacional para a Sociedade de Informação, Cabo Verde.
[4]CEMET, Faculty of Computing, Engineering and Science, University of South Wales, UK.

*Corresponding author: Rahmat Ullah (e-mail: rahmat.ullah@southwales.ac.uk)

ABSTRACT One of the primary jobs of a software project manager is to assign available resources to software development tasks in such a way that results in a high-quality product at a low cost. Software Project Scheduling (SPS) allocates the most appropriate human resource to project activities at the right time to reduce software project failure risks and minimise project makespan. In literature, the SPS problem is referred to as the Multiple Resource-Constrained Project Scheduling Problem (MRCPSP). The MRCPSP assigns human resources with multiple skills and proficiency levels to various project activities. Human abilities can be distinguished into technical/hard and non-technical/soft skills. The former describes the skills related to technology, tools, etc. While the latter deals with the skills related to the personality, such as being introvert, extrovert, sensing, etc. Recent studies have shown that some tasks may require specific soft skills. Moreover, the efficiency and productivity of the assigned resource significantly reduce if the soft skill requirements are ignored during task allocation. Ultimately, the development process might end up in lower-quality software products with higher development costs; worst case, the project may even fail. Several MRCPSP-based SPS approaches have been designed to reduce the development costs of software projects. These mechanisms consider the hard skills of a human resource with different proficiency levels. However, they overlook the soft skills required leading to the inefficiency of the allocated human resources. This will increase the project makespan and may cause higher development costs or even project failures. Therefore, to fill this gap, we propose Multi-Skill Resource Constrained and Personality Traits based Project Scheduling (MSRCPPS) considering the soft skills as well as the technical skills of a human resource during SPS. The main objective is to minimize software project makespan. Finally, the effectiveness of our proposed approach is evaluated against existing state-of-the-art using extensive simulations.

INDEX TERMS Project Management, Personality Traits, Resource Scheduling, Soft Skills, Personality Models

I. INTRODUCTION

One of the crucial part of software project management (SPM) is to finish software projects (SPs) within their set deadlines and budget constraints while ensuring a high-quality end product. SPM consists of complex tasks such as scheduling, planning, monitoring and controlling project tasks [1]. Among these tasks, Scheduling SP tasks is one of the critical steps, which deals with calculating the optimal schedule for the project's activities, ensuring that no predefined constraints such as resource constraints, time constraints and precedence constraints are violated [2]. According to Ref. [3], Software project scheduling (SPS) includes five main processes: 1) identifying project activities, 2) identifying activity dependencies, 3) estimating resources for activities, 4) allocating people to activities based on their skills and 5) creating project charts. Software project scheduling problem (SPSP) requires an optimal solution for arranging software engineers to complete the project within the required constraints, i.e. time and budget [4]. As stated in [5], SPSP is an NP-hard problem due to its complexity.

Therefore, it is beneficial to design a scheduling strategy to assist SP managers during the allocation of resources to SP tasks.

It is essential to have a precise timeline before executing a software project to achieve its intended goals. One of the key parts of project scheduling is human resources, varying in technical skills and personality traits. This relationship between human resources and the project activities having precedence relations makes project scheduling even more tedious. Hence, project scheduling is considered a primary problem due to the following constraints, i.e. precedence, resource, deadline, and skill constraints [1].

Any software organization aims to produce and deliver a quality product to the end user before the deadline, within the budget and according to customer needs [6]. In software development, the human factor is considered the most critical factor in the success of any project [7]. The personality of team members can significantly impact the effectiveness and productivity of software development teams [8]. Team combination relies on the communication skills and abilities of the members, and that's why the productivity of a software organization is highly affected by the level of social interaction between the team members [9]. Capretz et al. proposed a framework for mapping the personality traits of software engineers with the main stages of the software life cycle using the Myers-Briggs Type Indicator (MBTI) personality model. Their framework shows that system analysts should be extroverts or people with feeling personality traits. Similarly, they map programming with an introvert, sensing and thinking traits of the MBTI personality model [10].

The Myers-Briggs Type Indicator is one of the most trusted and commonly used tools for assessing an individual's personality. MBTI ranks individuals according to their personality traits by eight factors: "Extroversion vs Introversion, Sensing vs Intuition, Thinking vs Feeling and Judging vs Perceiving [11]. This model is described in table 1 in detail.

Though it is an essential factor, the majority of the studies and practices only deal with technological or process related factors instead of focusing on organizational, social or psychological factors [12]. People are the fundamental and critical factor in the success of an SP. Therefore, software engineering generally focuses on software development teams. "Development team" is classified as a team which is defined as [13] "an arrangement between two or more people to work together to produce an identifiable good or service in such a manner that the group members are highly interdependent". Software development teams differ in physical characteristics from the teams of other domains, as they vary in the nature, purpose and required environment [14]. Most software development projects are team-based and may result in conflicts among the team members during project development. According to the literature, human resource is one of the most critical factors in software development; however, existing literature related to MRCPSP disregards this fact and only deal with technological or process-related factors instead of focusing on organizational, social or psychological factors [12]. Also, the weights assigned to human resource against their technical skills is not realistic. To solve the stated problems, we design a scheduling heuristic that considers the human resource's soft skills and technical skills during the project task allocation. Moreover, unlike existing studies, we assign weights to human resources between 0 and 1, where 1 indicates the human resource performs certain task with 100% efficiency.

In short, our contributions in this paper are as follows:

- We propose an MSRCPPS approach that, unlike existing literature, considers realistic weights (between 0 and 1) for resources against their technical skills based on their proficiency level.
- The proposed MSRCPPS design also regards soft skills (personality traits) and technical skills as a requirement for performing a certain activity.
- The proposed scheme considers executing tasks in parallel if the precedence constraints are not violated. Moreover, extra resources (if any) are also allocated to minimize the makespan of SPs and avoid resource wastage.
- We argue the efficacy of our proposed MSRCPPS scheme in minimizing project makespan compared to the PSS heuristic using simulations.

The remainder of the paper is organized as follows. Section II delivers a summary of the related work of personality psychology in software engineering and project scheduling. Section III puts forward the mathematical formulation of the MSRCPPS problem. Section IV provides a brief discussion of the proposed framework for project scheduling and resource staffing. Finally, Sections V and VIpresent the simulation results and conclusion of this work, respectively.

II. RELATED WORK

Researchers have been trying to establish a relationship between personality and performance for the past many years, [16]. Human behaviour is generally affected by the characteristics of one's personality. Although many people agree in the change in behaviour according to different situations, some individuals are noticed for their consistent patterns in behaviour. Many approaches are used for organizing individuals according to their personality characteristics. For capturing individual non-context-dependent behaviour, psychometric tests are used. These tests are commonly used for assigning the right person to the right job by considering their soft skills.

A. PERSONALITY IN SOFTWARE ENGINEERING

Nevertheless, observing individuals' personalities may also reveal their personality characteristics. The big five inventory (BFI) or Five-factor model (FFM) is a framework widely used by researchers focusing on personality traits [8]. This framework considers five factors for the classifying of individual personality characteristics, these factors are "Openness, Conscientiousness, Extroversion, Agreeableness, Neu-

TABLE 1. MBTI Personality Model

Attributes/Factors	Characters	Applications	Measurement Method	Advantages	Ref
Extroversion	Ambitious Lively Talkative, Socially active, Enthusiast	Philosophy psychology Social Sciences Technology Science and Arts Linguistics	The measurement method used for MBTI is usually a survey based on 100 questions, these questions have two options, YES and NO	It offers insightful information on how team members' personalities will affect their behaviour at work, allowing team leaders to comprehend how the members of the team work together, address problems, and control their emotions during a project.	[11] [16] [17] [18] [19]
Introversion	Quiet, Thinking, Self-observer, Shy				
Sensing	Practical				
Intuition	Innovative, Flexible				
Thinking	Gentle, Calm				
Feeling	Enthusiastic, Good interpersonal relationship				
Judging	Organized, Arbiter				
Perceiving	Flexibility, Spontaneous				

roticism" [17]. Myers-Briggs Type Indicator (MBTI), defined by Myers et. al., is one of the widely used psychometric instruments for analysing personality tests comprising about hundred forced-choice objects regarding four personality dimensions [18]. These dimensions are "extroversion vs introversion (E/I), intuition vs sensing (N/S), thinking vs feeling (T/F) and judging vs perceiving (J/P)". Another prevalent approach used for personality assessment is Keirsey Temperament Sorter (KTS). KTS is closely linked with MBTI; the reason is that it is the extension of traditional MBTI. It contains four temperaments which are "Idealist, Guardian, Rational and Artisan" [13].

People are critical and fundamental factors in the success and failure of SPs. There is recognition that personality significantly impacts software process productivity and efficiency. T. Acuna et al. stated that only a few empirical studies compare and categorise team factors to the product's high effectiveness [20]. Kost et al. have done an empirical study to determine the influences among emotional intelligence and work preferences [21]. In [22] an interactive personality, the profiling approach proposed a structure for an effective software team using the BFI model. Another approach is proposed in OR Mathematical programming formulation for Multiple Team Formation Problems (MTFP), focusing on the allocation of multiple people to multiple teams or groups using BFI personality model [23]. In addition, several empirical studies have focused on the effect of personality on pair programming using different personality assessment models [24] [25]. Varona et al. perform a survey to explore the existing trends in software engineering using the MBTI personality model [26]. The authors identified different traits and factors like sensing, feeling, extroverts, and perceiving that exist among software engineers and developers.

Similarly, Personality types in the software engineering domain concerning the MBTI trait model have been studied in [6]. The authors argue that in the last two to three decades ago, people misunderstood the field of software engineering and software development. However, with the advancement of the software engineering domain, which is as much diverse as the medical profession. Different types of personalities and job roles exist, such as designers, system analysts, testers, and programmers. Experimental results depict that 57% of professionals are introverts compared to extroverts (i.e. 43%). The same is the case with sensing and other types of indicators. Personality types and their impact on development are presented in the literature survey in [7]. The study focuses on different personality types and methods and techniques presented by different researchers from 1970 to 2010. The authors concluded the most discussed and investigated areas in terms of personality aspects in software engineering are pair programming and team coordination and building. The study also highlighted some contradictions among the personality traits and techniques that can result in complexities in the software development process.

In [27], another survey-based study was also performed by Varona et al., which deals with identifying students by their personality who were most likely to complete their bachelor's degree. Many researchers have used different personality models to catch students' personalities, work quality and performance [28] [29] [30]. A multi-agent tool is proposed in [31] to determine the effect of employees and task allocation approaches in a different dynamic environment. Recently, A hyper-heuristic based ensemble genetic programming (HH-EGP) method is proposed for solving stochastic resource-constrained project scheduling problem (SRCPSP) by evolving an ensemble of priority rules [44]. In addition, a sequence voting mechanism is designed to deal with collaborative decision-making in the scheduling process. The benchmark PSPLIB is performed to verify the advantage over heuristics, meta-heuristics and other approaches. An extension of this approach, called the hyper-heuristic-based filtering genetic programming (HH-FGP) framework, is proposed for evolving priority rules to deal with a multi-project scheduling problem [45]. The proposed framework, namely, Stochastic Resource Constrained Multi-Project Scheduling Problem with New Project Insertions (SRCMPSP-NPI), consider stochastic activity duration and new project insertion together within heuristic computation time. The genetic and local search is improved to meet the depth constraints. A multi-objective evaluation mechanism is used to achieve effective filtering.

B. SOFTWARE PROJECT SCHEDULING

SPS is a way of organizing and managing SPs to achieve defined milestones [32] [33] while considering time and budget constraints. Scheduling is integrated with resources, project scope, budget, and requirements for software product development. It helps the project managers plan the activities accordingly and assign the most effective human resources to achieve milestones. The main goal is to complete the project on time [2] [34].

Assignment of the resources to project activities is an

TABLE 2. Basic notations used

Symbols	Definition
τ	Set of software project tasks
R	Set of available resources
t_ς	Set of technical/hard skills
ρ	Set of soft skills or personality traits
E	Set of edges denoting precedence constraints
τ_{t_ς}	Technical skills required by τ
τ_p	Personality Required by τ
$\tau_{R\rightarrow t_\varsigma}$	Matrix denoting the number of resources required for each task $\in \tau$
Ex^t	Execution time of a task
SP_{ms}	Software project makespan
$R_j\tau_i$	Equals 1 if R_j is assigned to τ_i, 0 otherwise
$R_j\tau_i t_{\varsigma_k}$	Equals 1 if R_j is assigned to τ_i against t_{ς_k}, 0 otherwise
E^r	Effort required to finish a task within its deadline

essential step in the scheduling process. Due to different constraint during the resource allocation process, a different problem arises. These constraints vary in nature due to the variance of the project and its equivalent goals [35]. To optimize the project makespan, an Ant Colony Optimization Algorithm (ACO) is used by considering the resources' hard skills, workload and salaries while not considering the soft skills or personality of the resources [1]. Similarly, for Multi-Skills Resources Constrained Project Scheduling Problem (MSRCPSP), a Parallel Scheduling Scheme (PSS) is proposed to minimize the project's completion time. But this study poorly defines the weight assigned to the skills and does not consider the soft skills [2]. Rahimi et al. proposed a Meta-Heuristic Algorithm for the optimization of MSRCPSP, and they also neglected the soft skills factor of human resources [36].

An algorithm was proposed by Chen et al. for Resource Constrained Multi-Project Scheduling Problem (RCMPSP) in [37]. The aim was to minimise project cost and duration by optimizing resource allocation. The psychological factor of human resources was not considered.

A multi-skilled-based extension of the resource-constrained project scheduling problem is presented by [38]. The proposed technique considers the depth and breadth of skill sets. It applies a genetic algorithm to find the best possible distribution of resources across different project activities and tasks. The proposed technique focuses more on the depth and breadth of the skills of resources. Moreover, a new crossover mechanism and local search methods are implemented in the genetic algorithm to determine the depth and breadth of skill resources over the makespan of the project. The proposed technique is computationally expensive due to the utilization of local search mechanisms and crossover functions. Genetic programming and a hyper-heuristic-based approach have been presented in [39]. In the proposed technique genetic algorithm is applied to the low-level heuristics to solve the multi-skilled resource-constrained project scheduling problem. The proposed approach focuses more on the skill set of resources and tries to allocate them for different activities based on their skill.

In [40] the authors proposed a variable neighbourhood search (VNS) based algorithm to develop a solution for a multi-mode and multi-skill scheduling model that takes into account the different abilities of employees and available resources for a particular project. The proposed technique has two dimensions of resource-constrained project scheduling projects that are multi-mode and multi-skill. The resource transfer time in multi-skilled constrained project scheduling has been studied in [41]. The proposed technique takes into account the time delay in the transfer of resources from one activity to another activity. This transfer usually does not occur smoothly, and delays and blockers can happen due to different parameters. To solve this problem, a genetic algorithm-based heuristic has been developed to tackle the issues of resource transfer. The proposed technique takes very few parameters in the evaluation process and tends to be computationally expensive.

A more realistic mathematical approach for scheduling dynamic software projects was proposed in [48] . The proposed approach recognises the skill proficiency of employees over time, motivation, and learning capacity. Under practical constraints, the employee's stratification is considered alongside project time, cost, robustness, and stability objectives. A proactive-rescheduling solution for changing software project schedules based on Q-learning is proposed. Experiments were performed on 18 dynamic benchmark samples and three software development project situations. The proposed approach was compared to seven existing meta-heuristic methods that improved convergence performance in dynamic environments while preserving solution distribution and spread. A similar mode for SPS was developed that combines the time-varying development of employee experience and learning ability [46]. The experimental results on 24 problem examples, including six real-world situations, demonstrated that the proposed SPS model shortens project duration by 40% while staying within budget. The results demonstrate that considering the evolution of expertise during the reallocation of activities in response to dynamic events significantly improves project schedules.

A cooperative coevolutionary multi-objective genetic algorithm is proposed in [47] to solve a mathematical model for the large-scale multi-objective software project scheduling problem. The proposed model addresses two efficiency-related goals, length and cost, in addition to resilience against uncertainty and employee satisfaction. The performance of the proposed model was evaluated on 15 randomly generated large-scale software project scheduling examples with up to 2048 decision factors and three instances derived from real-world software projects. The results indicated that the proposed algorithm achieved better convergence performance.

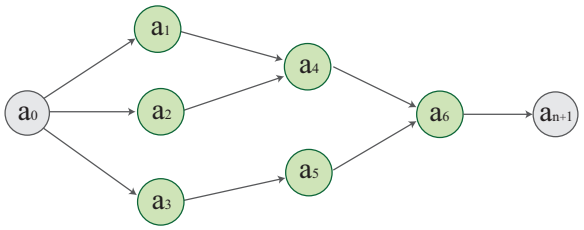


FIGURE 1. Task Precedence Graph

III. MULTI-SKILLS RESOURCE CONSTRAINED AND PERSONALITY TRAITS BASED PROJECT SCHEDULING (MSRCPPS)

The basic symbols and notations used in the remainder of the paper are listed in Table 1.

A. PROBLEM FORMULATION

Consider a SP denoted by a set of tasks $\tau = \{\tau_0, \tau_1, \tau_2, \tau_3, \ldots, \tau_n, \tau_{n+1}\}$ where τ_0 and τ_{n+1} are dummy activities representing the start and end of the project, respectively. While τ_i represent i^{th} task in τ. Let $E^r_{\tau_i}$, $\bar{d}_{\tau_i}$, and $\dot{D}_{\tau_i}$ denote the effort required, the duration, and the deadline of a task τ_i, respectively. Moreover, there is a strict relationship among the activities of a SP as shown by a Task Precedence Graph (TPG) in Fig. 1. A SP with multiple tasks with precedence relations can be defined mathematically as:

$$TPG = (\tau, E) \tag{1}$$

where

$$E = \{(\tau_i, \tau_j) \mid \tau_i : \tau_j \in \tau \wedge (\tau_i \rightarrow \tau_j)\} \tag{2}$$

where τ represents vertices and E denotes the dependency/precedence constraints among the elements of τ as a set of edges within the TPG. Here, $i, j \in \{0, 1, 2, \ldots, n, n+1\}$ and $\tau_i \rightarrow \tau_j$ depicts the precedence constraint between the tasks τ_i and τ_j; τ_j cannot be started ahead or in parallel of τ_i.

Furthermore, a SP tasks need to be allocated a set of available resources. Let $R = \{r_1, r_2, \ldots\ldots, r_n\}$ be the set representing the resources that can be assigned to the project. Each resource can de denoted by r_j, where $j \in \{1, 2, \ldots, n\}$. Each resource may have multiple technical/hard skills represented by a skill set $t\varsigma_{r_j} = \{t\varsigma_{r_{j1}}, t\varsigma_{r_{j2}}, \ldots, t\varsigma_{r_{im}}\}$ where $k \in \{1, 2, \ldots, m\}$ and $t\varsigma_{r_{jk}}$ represents the k^{th} technical skill of resource r_j. Moreover, each resource must possess a certain personality denoted by a set of soft skills $\rho_{r_j} = \{\rho_{r_{j1}}, \rho_{r_{j2}}, \ldots, \rho_{r_{jl}}\}$. The MBTI personality traits modelling the soft skills can be defined by the set $\rho = \{[i, e], [s, n], [t, f], [j, p]\}$ where $i, e, s, n, t, f, j,$ *and* p represent introvert, extrovert, sensing, intuitive, thinker, and perceiving, respectively. The value of $\{t\varsigma_{r_{jk}}$ and $\rho_{r_{jo}}$ are defined in a range of 0-1 representing the proficiency level of r_j in performing his k^{th} technical skill and o^{th} soft skill where $o \in \{1, 2, \ldots, l\}$, respectively. Moreover, each pair of soft skills $[x, y]$ in ρ has one important constraint when representing the personality of any r_j in R; a resource cannot have both x and y skills as they are opposite in nature. This means a resource can not be both introvert and extrovert. Hence, the personality of a resource r_j can be represented by any combination of these soft skills, provided the aforementioned constraint is not violated.

Each project activity requires a different skill set with a distinct level of expertise against each talent for successful execution. Apart from technical skills, human resources have some soft skills called personality traits. Each activity may also require a suitable personality trait and technical skills. Therefore, the following activity skill matrix denotes the required technical skills against each activity.

$$\tau_{t\varsigma} = \begin{bmatrix} \tau_1 t\varsigma_1 & \tau_1 t\varsigma_2 & \ldots & \tau_1 t\varsigma_m \\ \tau_1 t\varsigma_1 & \tau_1 t\varsigma_2 & \ldots & \tau_1 t\varsigma_m \\ \vdots & \vdots & \vdots & \vdots \\ \tau_1 t\varsigma_1 & \tau_1 t\varsigma_2 & \ldots & \tau_n t\varsigma_m \end{bmatrix}$$

And to represent the required personality traits for different tasks of an SP, we use a task personality matrix as follows.

$$\tau_{\rho} = \begin{bmatrix} \tau_1 \rho_1 & \tau_1 \rho_2 & \ldots & \tau_1 \rho_m \\ \tau_1 \rho_1 & \tau_1 \rho_2 & \ldots & \tau_1 \rho_m \\ \vdots & \vdots & \vdots & \vdots \\ \tau_1 \rho_1 & \tau_1 \rho_2 & \ldots & \tau_n \rho_m \end{bmatrix}$$

Similarly, the same data structures can denote a resource's technical and soft skills in R. Moreover, each task may require more than one resource having required technical skills. This can be defined by the following matrix.

$$\tau_{R \rightarrow t\varsigma} = \begin{bmatrix} \tau_1 t\varsigma_1 & \tau_1 t\varsigma_2 & \ldots & \tau_1 t\varsigma_m \\ \tau_1 t\varsigma_1 & \tau_1 t\varsigma_2 & \ldots & \tau_1 t\varsigma_m \\ \vdots & \vdots & \vdots & \vdots \\ \tau_1 t\varsigma_1 & \tau_1 t\varsigma_2 & \ldots & \tau_n t\varsigma_m \end{bmatrix}$$

This matrix is similar to the first one, however, with one main difference; the values are integers representing the number of resources required against each technical skill. Finally, the proficiency of all resources in R in performing the technical skills can be represented by the same matrix as first. Also, the soft skills of all resources in R can be represented by the second matrix.

The time to perform a task depends on the required resources and assigned resources against each skill. For instance, let $Ex^t_{\tau_i}$ denote the execution time of a task τ_i. Hence, the execution time of a task can be calculated as:

$$Ex^t_{\tau_i} = max(E^t_{\tau_i t\varsigma_j}) \tag{3}$$

where $E^t_{\tau_i t\varsigma_j}$ denotes the execution time against j^{th} technical skill and is given by:

$$Ex^t_{\tau_i t\varsigma_j} = \frac{ReqResources * E^r_{\tau_i}}{AssResources} * \frac{ReqSoftSkill}{AssSoftSkill} \tag{4}$$

The required resources are calculated by multiplying the number of resources required in $\tau_R \rightarrow t_\varsigma$ and the required skill proficiency level in τ_{t_ς}. To quantify the soft skills of the human resource available, we use a scale of 0 and 1. For instance, a soft skill with a value of 0 means the person does not possess the specified soft skill, while 1 means the person has the specified soft skill.

B. THE OBJECTIVE FUNCTION

Considering the above formulation, we define the SPS problem as the assignment of R to an SP denoted by a set of tasks to minimize the SP makespan (SP_{ms}) while regarding the precedence constraints among the project activities. Mathematically:

$$minimize(SP_{\text{ms}}) \tag{5}$$

subject to:

$$\forall(\tau_i, \tau_j) \in E, \quad \dot{S}_{\tau_j} >= \dot{F}_{\tau_i} \tag{6}$$

$$\forall\tau_i \in \tau, \quad \dot{F}_{\tau_i} <= \dot{D}_{\tau_i} \tag{7}$$

$$\sum_{i=1}^{n} R_j\tau_i = 1 \tag{8}$$

$$\sum_{i=1}^{n} R_j\tau_i t_{\varsigma_k} = 1 \tag{9}$$

Where Eq. (6) depicts that the starting time of τ_j must start after τ_i is finished. This ensures that the precedence constraints defined by each edge is satisfied. Similarly, Eq. (7) states that each $\tau_i \in \tau$ must be finished before its deadline. Eq. (8) and (9) define the constraint that a resource R_j should be assigned to only against one skill requirement and to only one task at a time.

IV. DESCRIPTION OF THE PROPOSED ALGORITHM

There are two main steps in minimizing the overall makespan of the project: (1) similar to existing strategies, our proposed strategy can run multiple tasks in parallel if the precedence constraint is not violated, (2) unlike existing strategies, it tries to avoid resource wastage by assigning the free resources to tasks that help minimize the overall makespan. The proposed scheduling system is divided into three algorithms: (1) Schedule_Project, (2) Schedule_Tasks, and (3) Assign_Resources. The main entry point of the proposed design is Algorithm 1.

A. ALGORITHM 1: SCHEDULE SOFTWARE PROJECT

Algorithm 1formalizess the main steps of the proposed algorithm. It takes $\tau, E, R, t_\varsigma, \rho$ denoting the set of tasks, edges, available resources, technical skills required to perform the tasks, and the personality traits required against each task, respectively.

Algorithm 1: Schedule_Project

Input: τ, E, R, t_ς, ρ

```
1  τ_parallel = [ ]
2  while τ is not empty do
3      τ_sub = [ ]
4      while not end of τ do
5          t = next(τ)
6          for e ∈ E do
7              if the right vertex of the e is t then
8                  continue 2nd loop //
9              end
10         end
11         τ_sub = τ_sub + t
12         τ = τ/t
13         remove all edges from E whose left vertex is t
14     end
15     τ_parallel = τ_parallel + τ_sub
16 end
17 return SP_ms = Schedule_Tasks(τ_parallel, R)
```

The algorithm starts by initializing $\tau_{parallel}$ as an empty array at the line 1. Each element in $\tau_{parallel}$ will be an array of tasks denoting a group of parallel tasks. After that, the algorithm finds all the groups of parallel tasks and adds them to the $\tau_{parallel}$ (from line 2 to 2). A group of parallel tasks is denoted by τ_{sub}. The condition for each element in τ_{sub} is that there is no edge $e \in E$ whose right vertex is equal to t. This condition is checked from line 6 to 13. If there is an edge whose right vertex is t, the algorithm does not add this task to the current group and checks the next task in τ. Otherwise, the task is added to the group of parallel tasks and removes all the edges whole left vertex is t at line 13.

The process of finding parallel executable tasks and grouping them is iterative; each iteration produces one group of parallel tasks until the τ is empty. After finding all the groups of parallel executable tasks, the algorithm calls the Schedule_Tasks algorithm to schedule the project represented by an array of parallel executable tasks groups $\tau_{parallel}$ at line 17 and returns the SP makespan SP_{ms} received from the called algorithm.

B. ALGORITHM 2: SCHEDULING TASKS IN PARALLEL

Algorithm 2 schedules each group of parallel executable tasks received from Algorithm 1 until all the groups in $\tau_{parallel}$ are executed completely. The algorithm takes $\tau_{parallel}$ and R as an input, where R represents the set of all available resources. First, the algorithm initializes a set of free resources by R. Next, from line 3 to 21, the algorithm takes each group of parallel executable tasks τ_{sub} one by one and assigns resources to tasks in τ_{sub} until all tasks are executed.

Furthermore, for each task $t \in \tau_{sub}$, the algorithm calls

Algorithm 2: Schedule_Tasks

Input: $\tau_{parallel}, R$

1 $R_{free} = R$
2 $SP_{ms} = 0$
3 **for** $\tau_{sub} \in \tau_{parallel}$ **do**
4 $\tau_{sched} = [\,]$
5 **while** τ_{sub} *is not empty* **do**
6 **for** $t \in \tau_{sub}$ **do**
7 **if** $assign\_resources(t, R_{free}, 0) == 1$ **then**
8 $\tau_{sched} = \tau_{sched} + t$
9 $\tau_{sub} = \tau_{sub}/t$
10 **end**
11 **end**
12 **sort** τ_{sched} in ascending order of their finishing time
13 **for** $t \in \tau_{sched}$ **do**
14 **if** R_{free} *is not empty* **then**
15 $assign\_resources(t, R_{free}, 1)$
16 **end**
17 **end**
18 $\tau_{ef} = get\_earliest\_finishing\_task(\tau_{sub})$
19 $SP_{ms} = SP_{ms} + get\_finishing\_time(\tau_{ef})$
20 $\tau_{sub} = \tau_{sub}/\tau_{ef}$
21 $R_{free} = R_{free} + free\_resources(\tau_{ef})$
22 **end**
23 **end**
24 $return\ SP_{ms}$

Algorithm 3: Assign_Resources

Input: $t \in \tau_{sub}, R_{free}, assign\_extra$
Output: *returns* 0 if resources are not enough for the selected task and 1 otherwise.

1 **for** $\tau_i t\varsigma_j \in \tau_i t\varsigma$ **do**
2 **while** *TRUE* **do**
3 **if** $assign\_extra == 0$ **then**
4 $\dot{F}_t = calculate\_finishing\_time()$
5 **if** $ft \leq \dot{D}_t$ **then**
6 break;
7 **end**
8 **end**
9 **while** R_{free} *is not empty* **do**
10 $R^* = find\_resource(R_{free}, \tau_i t\varsigma_j)$
11 **if** R^* *is null &* $assign\_extra == 0$ **then**
12 set all assigned resources free as there are not enough resources
13 $return\ 0$
14 **end**
15 **if** R^* *is not null* **then**
16 assign R^* to t against $\tau_i t\varsigma_j$
17 remove R^* from R_{free}
18 break
19 **end**
20 **end**
21 **end**
22 **end**
23 $return\ 1$

Algorithm 3 to assign resources to finish the task within the deadline (line 6 - 11). If the resource assignment is successful (line 7), the algorithm adds the scheduled task to τ_{sched} and removes it from τ_{sub}. Next, the algorithm sorts all the tasks in τ_{sched} based on their finishing time in ascending order.

After assigning enough resources to each task, if there are still free resources available, the algorithm tries to assign those free resources minimise the makespan and reduce resource wastage. This is depicted from line 13 to 17. Finally, the algorithm finds the earliest finishing task, adds its duration to the makespan, removes the task from τ_{sub}, and sets the resources free. After executing all the groups of parallel executable tasks, the algorithm returns the makespan.

C. ALGORITHM 3: ASSIGNING RESOURCES TO INDIVIDUAL TASKS

Algorithm 3 depicts the steps performed by the proposed scheduling scheme to assign resources to the selected task $t \in \tau_{sub}$. It takes t, R_{free}, and $assign\_extra$ as input parameters, and $assign\_extra$ is a boolean value to represent whether the algorithm is assigning extra or required resources. $assign\_extra = 1$ means that the algorithm should allocate extra resources. Hence, Algorithm 3 can be divided into two paths based on this parameter.

The first path is for assigning only required resources when the input parameter $assign\_extra = 0$. The algorithm takes each skill of the task and checks if the suitable resource is available to fulfil the requirements. If no suitable resource is found, the algorithm sets all the resources free that are assigned to this task and return 0, indicating that the available resources are not enough to fulfil the requirements (line 11-14). Otherwise, it assigns resources against the selected skill (lines 15-19) until the finishing time against the skill chosen is within the deadline (lines 3-8).

The second path is for assigning extra resources to already scheduled tasks when the input parameter $assign\_extra = 1$. The algorithm skips lines 3-14 as they are meant to allocate required resources. However, lines 15-19 are followed to assign extra resources. The algorithm first checks if the available resources contain suitable resources to assign and then assigns the resource to the selected task. Then, the assigned resource is removed from the free resources

TABLE 3. Experimental Set

	No. Of Tasks	No. Of Res.	No. Of Skills	(Pred, Succ)	$\bar{d}_\tau$	Max.Res. / Skill	Max.Res. / Task
Experiment 1	10	8	4	3,3	1-10	3	5
Experiment 2	20	8	4	3,3	1-10	3	5
Experiment 3	10	12	4	3,3	1-10	3	5
Experiment 4	10	8	2	3,3	1-10	3	5
Experiment 5	10	8	4	2,2	1-10	3	5
Experiment 6	10	8	4	3,3	1-15	3	5
Experiment 7	10	8	4	3,3	1-10	1	5
Experiment 8	10	8	4	3,3	1-10	3	3

set. This procedure is repeated until resources are allocated against each skill. Finally, the algorithm return 1, indicating successful assignment of resources after iterating over all the skill set.

V. EXPERIMENTS

To assess the appropriateness of the proposed scheduling design, we conduct extensive simulations to study the impact of variation in different parameters such as the complexity of the TPG, number of tasks, number of maximum predecessors or successors, etc. The details of each experiment with parameter settings are summarised in the table 3. Each experiment considers 50 projects to schedule.

A. BASELINE EXPERIMENT

Experiment 1 provides global parameter settings and is a baseline for the following experiments as we keep changing one parameter at a time in each experiment and keep other parameters the same as global ones. By doing so, we are able to compare the difference due to each parameter in their corresponding experiments. In this experiment, the proposed technique MSRCPPS is validated against the PSS heuristics proposed in [2]. Figure 2 depicts the simulation results obtained.

Moreover, the average completion time of our proposed algorithm is 47.26, and that of the PSS heuristic is 48.46. The Proposed algorithm competes the PSS heuristic 31 times out of 50 projects. It shows that the proposed algorithm completes 62% of the projects earlier than the PSS heuristic.The proposed technique better utilizes free resources with the average free resources of 0.63 while the average free resources of PSS heuristics was 2.13.

B. IMPACT OF NUMBER OF ACTIVITIES

This experiment studies the impact of a number of project tasks on the project makespan while keeping all the other settings constant. Figure 3 shows that varying the number of activities slightly impacts the relative appropriateness of the

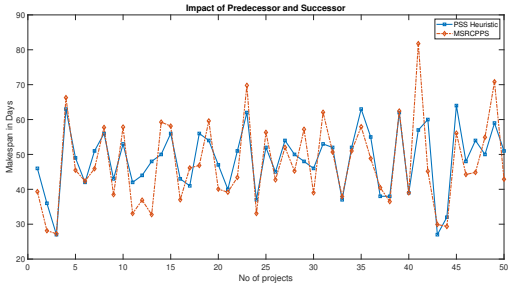


FIGURE 2. Experiment 1: Constant

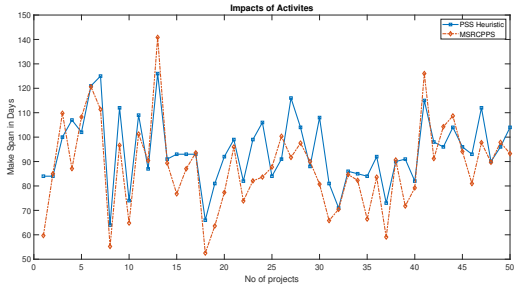


FIGURE 3. Experiment 2: Impact of number of activities

proposed strategy against the PSS heuristic in minimizing the project makespan.

The average completion time of the proposed technique is 89.44 for each project, while the average completion time of the PSS heuristic is 96.20. Their average difference was 6.76. Out of 50 projects, the proposed algorithm achieved better results in 37 projects. Moreover, the proposed algorithm utilizess free resources and the average free resources per project was 0.54 compared with the PSS heuristic, with average free resources of 2.15 per project.

C. IMPACT OF RESOURCES

In this experiment, the impact of several available resources on the appropriateness of the proposed strategy against the PSS heuristic is studied. Results depicted in Fig. 4 demonstrate that the proposed technique is more efficient than the "PS" heuristic in reducing the makespan of a project. Out of 50 projects, the proposed technique completed 43 projects earlier than the PSS heuristic, with an average completion time of 28.40. In contrast, the average completion time of the PSS is 36.48. The average difference was 8.08.

D. IMPACT OF NUMBER OF SKILLS REQUIRED

In experiment 4, we study the impact of several skills on the effectiveness of both the proposed and the PSS techniques. Simulation results depicted in Fig. 5 reveal that the number of skills required for each project greatly impacts the proposed technique. The average completion time of the proposed algorithm was 39.03, while the average completion time of the PSS heuristic was 44.34, with the average difference of

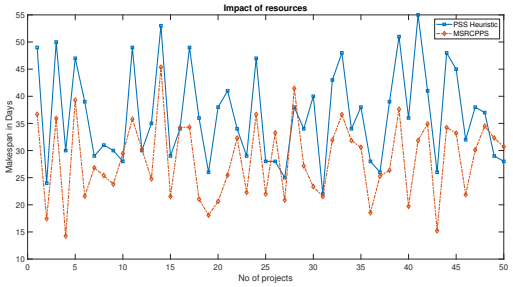


FIGURE 4. Experiment 3: Impact of Resources

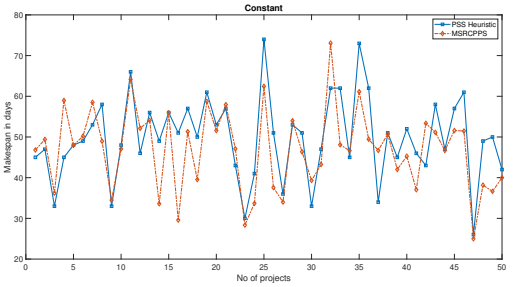


FIGURE 6. Experiment 4: Impact of Number of Predecessor and Successor

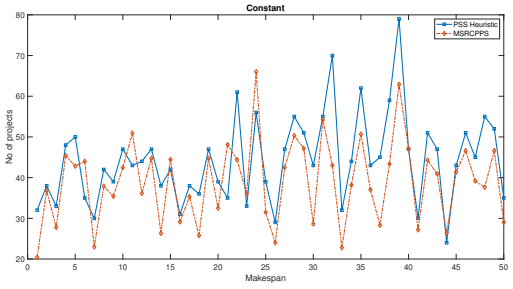


FIGURE 5. Experiment 4: Impact of Number of Skills Required

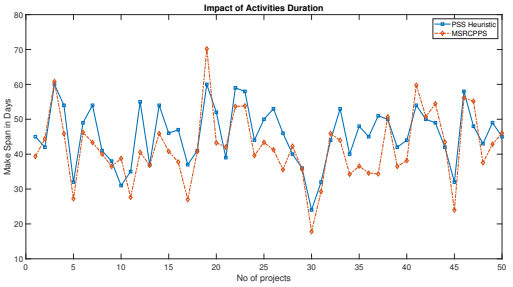


FIGURE 7. Experiment 6: Impact of Number of Duration of Activities

5.31.

In the simulation results, out of 50 projects, 44 were completed earlier by the proposed technique, with a percentage of 88%. From the experiment, we conclude that when the required number of skills for activities is increased, the average makespan of the project will be maximized. Therefore, the variety of available resources with multiple skills will be helpful for the execution of the project.

E. IMPACT OF NUMBER OF MAXIMUM PREDECESSORS AND SUCCESSOR

s In this experiment, we analyze the impact of the number of maximum predecessors and successors for each project activity. The number of predecessors and successors is decreased. Simulation results show that when the number of predecessors and successors is less, the proposed algorithm is more effective in reducing the makespan compared to its counterpart, as depicted in Fig. 6.

The graph clearly shows that the proposed technique, MSRCPPS results in a lower SP makespan compared to the PSS heuristic in more than 65% of the cases for the global settings. The average completion time of the PSS heuristic is 50.72 days, while the average completion time of our proposed design is 47.98 days. The average difference in days was 2.74. Out of 50 projects, the proposed algorithm performs better than the PSS heuristic in 36 projects.

F. IMPACT OF DURATION OF ACTIVITIES

This experiment aims to study the impact of the duration of activities on the effectiveness of our proposed algorithm and the PSS heuristic. It is observed that the proposed algorithm MSRCPPS provides performs more efficiently than the PSS heuristic when the duration of project activities, as shown in Fig. 7.

The average completion time of the proposed algorithm is 41.85, and the average completion time of the PSS heuristic is 45.56. These numbers depict that the proposed algorithm completed the projects earlier than the PSS heuristic, with an average difference of 3.71. The simulation results also revealed that the proposed algorithm completed 35 out of 50 projects earlier than the PSS heuristic.

G. IMPACT OF RESOURCES REQUIRED PER SKILL

Fig. 8 presents the results of experiment 7, where we analyze the impact of the number of resources required against each skill on the effectiveness of our proposed algorithm and the PSS heuristic.

Simulation results demonstrate that the MSRCPPS algorithm is more efficient than the PSS heuristic, as the average completion time of the proposed algorithm is 29.69, and the average completion time of the PSS heuristic is 35.66. The average difference in completion time is 5.97. Moreover, the proposed algorithm completed 40 projects out of 50 earlier than the PSS heuristic. Finally, the proposed algorithm better utilizes resources with the average free resources of 0.69 and the average free resources of PSS is 2.75.

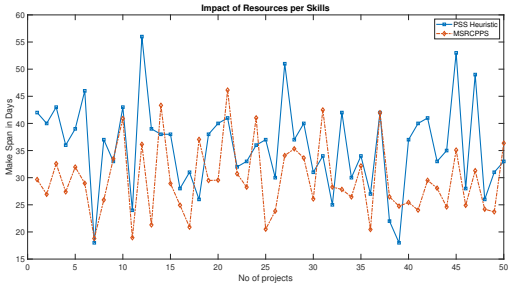


FIGURE 8. Experiment 7: Impact of Number of Resources Required per Skill

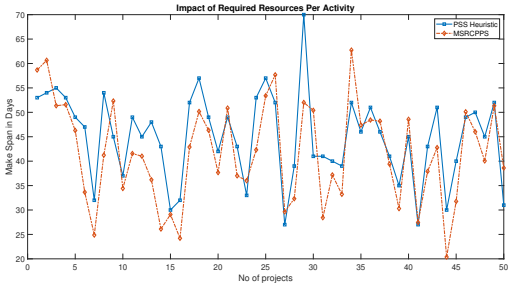


FIGURE 9. Experiment 8: Impact of Number of Resources Required Per Activity

H. IMPACT OF NUMBER OF RESOURCES REQUIRED PER ACTIVITY

In this experiment, we observe the impact of required resources per activity. Fig. 9 depicts the results obtained. The figure clearly demonstrates that when the number of resources required for each project activity is minimized, the proposed algorithm MSRCPPS performs better than the PSS heuristic. Simulation results show that out of 50 projects, 38 (76%) were completed earlier by the proposed algorithm. The average completion time of the proposed algorithm is 41.65, while the average completion time of the PSS heuristic is 44.88. The average difference in completion time is 3.23.

Furthermore, it is observed that the proposed algorithm better utilizes the free resources with an average of 0.53. At the same time, the average free resources of the PSS heuristic is 2.11. Hence, the simulation results clearly show that the proposed algorithm works more efficiently than the PSS heuristic when the number of resources required for each activity is reduced.

VI. CONCLUSION

This paper presents an algorithm for Multi-skill Resource Constrained Project Scheduling Problem (MSRCPSP). In our proposed technique, we consider heterogeneous resources and parallel executable activities. Furthermore, unlike existing literature, we consider soft skills, also known as the personality traits of human resources. To optimize the project makespan better, our proposed algorithm assigns resources to the activities satisfying both hard skills and personality traits requirements.

In order to argue the effectiveness of our proposed scheduling technique, we compare its performance with the PSS heuristic. A total of 8 experiments are performed where in each experiment, we vary a single parameter to study the effect of the parameter on the appropriateness of the proposed technique and the PSS scheduling algorithm in MSRCPSPs. A total of 50 projects are considered in each experiment.

Simulation results confirm that the proposed algorithm minimises the project makespan compared to the PSS heuristic. This is because the proposed algorithm utilizes free resources to minimize the project makespan by assigning them to the tasks in hand. Moreover, tasks are assigned to the earliest finishing tasks so that resources are freed and assigned to other tasks minimising the waiting time for tasks. Finally, ignoring resources' personalities can negatively impact a project's makespan. Therefore, the proposed algorithm outperforms its counterpart by assigning resources to tasks regarding technical skills and personality traits.

REFERENCES

[1] 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.

[2] 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.

[3] X. Shen, L. L. Minku, R. Bahsoon, and X. Yao, Dynamic software project scheduling through a proactive-rescheduling method, IEEE Transactions on Software Engineering, vol. 42, no. 7, pp. 658-686, 2016.

[4] K. Vitekar, S. Dhanawe, and D. Hanchate Review of solving software project scheduling problem with ant colony optimization, International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering, vol. 2, no. 4, pp. 1177-1182, 2013.

[5] M. R. Garey, DS Johnson Computers and intractability, A Guide to Theory NP-Completeness, 1979.

[6] B. Curtis, Human factors in software development,Encyclopedia of software engineering, 2002.

[7] R. Colomo-Palacios, C. Casado-Lumbreras, P. Soto-Acosta, S. Misra, and F. J. Garcá-Peñalvo, Analyzingg human resource management practices within the gsd context, Journal of Global Information Technology Management, vol. 15, no. 3, pp. 30-54, 2012.

[8] M. Yilmaz, R. V. O'Connor, R. Colomo-Palacios, and P. Clarke, An examination of personality traits and how they impact on software development teams, Information and Software Technology, vol. 86, pp. 101-122, 2017.

[9] S. Ryan and R. V. O'Connor, Development of a team measure for tacit knowledge in software development teams, Journal of Systems and Software, vol. 82, no. 2, pp. 229-240, 2009.

[10] L. F. Capretz and F. Ahmed, Making sense of software development and personality types, IT professional, vol. 12, no. 1, 2010.

[11] J. Gulati, P. Bhardwaj, and B. Suri,"Comparative study of personality models in software engineering," in Proceedings of the Third International Symposium on Women in Computing and Informatics, pp. 209–216, ACM, 2015.

[12] P. Lenberg, R. Feldt, and L. G. Wallgren, Behavioral software engineering: A definition and systematic literature review, Journal of Systems and software, vol. 107, pp. 15-37, 2015.

[13] A. B. Soomro, N. Salleh, E. Mendes, J. Grundy, G. Burch, and A. Nordin, The effect of software engineers personality traits on team climate and performance: A systematic literature review, Information and software technology, vol. 73, pp. 52-65, 2016.

[14] G. A. Dafoulas and L. A. Macaulay, Facilitating group formation and role allocation in software engineering groups, in ACS/IEEE International Conference on Computer Systems and Applications, 2001, pp. 352-359, IEEE, 2001.

[15] S. Sawyer, Effects of intra-group conflict on packaged software development team performance,Information Systems Journal, vol. 11, no. 2, pp. 155-178, 2001.

[16] S. T. Acuna, M. Gomez, and N. Juristo, How do personality, team processes and task characteristics relate to job satisfaction and software quality? Information and Software Technology, vol. 51, no. 3, pp. 627-639, 2009.

[17] R. R. McCrae and O. P. John, An introduction to the five-factor model and its applications, Journal of personality, vol. 60, no. 2, pp. 175-215, 1992.

[18] M. H. McCaulley, Mbti manual: A guide to the development and use of the myers-briggs type indicator, Mountain View, CA: CCP, 1998.

[19] J. Jia, P. Zhang, and R. Zhang, "A comparative study of three personality assessment models in software engineering field," in Software Engineering and Service Science (ICSESS), 2015 6th IEEE International Conference, pp. 7–10, IEEE, 2015.

[20] S. T. Acuna, M. N. Gomez, J. E. Hannay, N. Juristo, and D. Pfahl, Are team personality and climate related to satisfaction and software quality? Aggregating results from a twice replicated experiment, Information and Software Technology, vol. 57, pp. 141-156, 2015.

[21] M. V. Kosti, R. Feldt, and L. Angelis, Personality, emotional intelligence and work preferences in software engineering: An empirical study, Information and Software Technology, vol. 56, no. 8, pp. 973-990, 2014.

[22] M. Yilmaz, R. V. O'Connor, R. Colomo-Palacios, and P. Clarke, An examination of personality traits and how they impact on software development teams, Information and Software Technology, vol. 86, pp. 101-122, 2017.

[23] J. H. Gutierrez, C. A. Astudillo, P. Ballesteros-Perez, D. Mora-Melia, and A. Candia-Vejar, The multiple team formation problem using sociometry, Computers & Operations Research, vol. 75, pp. 150-162, 2016.

[24] P. Sfetsos, I. Stamelos, L. Angelis, and I. Deligiannis, An experimental investigation of personality types impact on pair effectiveness in pair programming, Empirical Software Engineering, vol. 14, no. 2, p. 187, 2009

[25] J. E. Hannay, E. Arisholm, H. Engvik, and D. I. Sjoberg, Effects of personality on pair programming, IEEE Transactions on Software Engineering, vol. 36, no. 1, pp. 61-80, 2010.

[26] D. Varona, L. F. Capretz, Y. Pinero, and A. Raza, Evolution of software engineers' personality profile, ACM SIGSOFT Software Engineering Notes, vol. 37, no. 1, pp. 1–5, 2012.

[27] D. Varona, Y. Lizama-Mue, and L. F. Capretz, A comparison of junior and senior software engineering student's personalities, in Proceedings of the 7th International Workshop on Cooperative and Human Aspects of Software Engineering, pp. 131-132, ACM, 2014.

[28] V. R. Montequn, J. V. Balsera, J. M. M. Fernández, and A. G. Nieto, Using myers-briggs type indicator (mbti) as a tool for setting up student teams for information technology projects, Journal of Information Technology and Application in Education (JITAE), JITAE, vol. 1, no. 1, pp. 28-34, 2012.

[29] N. Salleh, E. Mendes, and J. Grundy, The effects of openness to experience on pair programming in a higher education context, in Software Engineering Education and Training (CSEE&T), 2011 24th IEEE-CS Conference on, pp. 149-158, IEEE, 2011.

[30] V. Venkatesan and A. Sankar, Investigation of student's personality on pair programming to enhance the learning activity in the academia, Journal of Computer Science, vol. 10, no. 10, p. 2020-2028, 2014.

[31] H. Paredes et al., Advances in Social Computing and Multiagent Systems, Commun. Comput. Inf. Sci., vol. 541, no. August 2016, pp. 66-76, 2015.

[32] C. K. Wilson, Information technology. Aspens. Advis. Nurse Exec., vol. 11, no. 8, p. 2, 1996.

[33] R. Kolisch and R. Padman, An integrated perspective of project scheduling, in 65 Manuskripte aus den Instituten fur Betriebswirtschaftslehre der Universitat Kiel 463, pp. 0-34, 1997.

[34] Alhumrani, Sultan A., and Rizwan J. Qureshi., Novel approach to solve resource constrained project scheduling problem (RCPSP), International Journal of Modern Education & Computer Science vol. 8, no. 9, 2016.

[35] R. Kolisch and R. Padman, An integrated survey of deterministic project scheduling, Omega International Journal Management Science., vol. 29, no. 3, pp. 249-272, 2001.

[36] A. Rahimi, H. Karimi, and B. Afshar-Nadjafi, Using meta-heuristics for project scheduling under mode identity constraints, Applied Soft Computing, vol. 13, no. 4, pp. 2124-2135, 2013.

[37] J.-j. Chen, J.-l. Zhu, and D.-n. Zhang, "Multi-project scheduling problem with human resources based on dynamic programming and staff time coefficient," in Management Science & Engineering (ICMSE), 2014 International Conference, pp. 1012–1018, IEEE, 2014.

[38] Snauwaert, J., & Vanhoucke, M., A new algorithm for resource-constrained project scheduling with breadth and depth of skills, European Journal of Operational Research, vol. 292, no. 1, pp. 43-59, 2021.

[39] Lin, J., Zhu, L., & Gao, K., A genetic programming hyper-heuristic approach for the multi-skill resource constrained project scheduling problem, Expert Systems with Applications, vol. 140, pp. 112915, 2020.

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

[41] Cai, J., Peng, Z., Ding, S., & Sun, J., A Robust Genetic Algorithm to Solve Multi-Skill Resource Constrained Project Scheduling Problem with Transfer Time and Uncertainty Skills, in 2020 IEEE 16th International Conference on Control & Automation (ICCA), pp. 1584-1589, IEEE, 2020.

[42] Capretz, L. F., Personality types in software engineering, International Journal of Human-Computer Studies, vol. 58, no. 2, pp. 207-214, 2003.

[43] Cruz, S. S., da Silva, F. Q., Monteiro, C. V., Santos, P., Rossilei, I., & dos Santos, M. T., Personality in software engineering: Preliminary findings from a systematic literature review, in 15th annual conference on Evaluation & assessment in software engineering (EASE 2011), pp. 1-10, IET, 2011.

[44] Chen, HaoJie, Guofu Ding, Shengfeng Qin, and Jian Zhang. "A hyper-heuristic based ensemble genetic programming approach for stochastic resource constrained project scheduling problem." Expert Systems with Applications 167 (2021): 114174.

[45] Chen, HaoJie, Guofu Ding, Jian Zhang, Rong Li, Lei Jiang, and Shengfeng Qin. "A filtering genetic programming framework for stochastic resource constrained multi-project scheduling problem under new project insertions." Expert Systems with Applications 198 (2022): 116911.

[46] Nigar, Natasha, Muhammad Kashif Shahzad, Shahid Islam, Satish Kumar, and Abdul Jaleel. "Modeling Human Resource Experience Evolution for Multiobjective Project Scheduling in Large Scale Software Projects." IEEE Access 10 (2022): 44677-44690.

[47] Shen, Xiaoning, Yinan Guo, and Aimin Li. "Cooperative coevolution with an improved resource allocation for large-scale multi-objective software project scheduling." Applied Soft Computing 88 (2020): 106059.

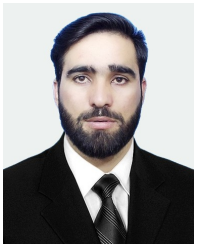
[48] Shen, Xiao-Ning, Leandro L. Minku, Naresh Marturi, Yi-Nan Guo, and Ying Han. "A Q-learning-based memetic algorithm for multi-objective dynamic software project scheduling." Information Sciences 428 (2018): 1-29.

SAEED AKBAR received his PhD in Computer Science from the School of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan, China in 2022. After completing completed his MS(SE) 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 Mathematics, Zhejiang Normal University, China. His research interest includes Resource Scheduling, Cloud Computing, Dynamic Thermal-management of Data Centers, Algorithmic Game Theory, and High Performance Computing.

IFTIKHAR AHMAD obtained his Bachelor's degree in Computer Science from the University of Malakand, Chakdara, Dir Lower, Pakistan, in 2014 and completed his MS in Software Engineering from COMSATS University, Islamabad, Pakistan, in 2019. He worked as a research assistant and as a teacher assistant in the department of Computer Science at COMSATS University Islamabad, Pakistan. He also served as a lecturer at the Department of Computer Science and Information Technology at the University of Lahore Islamabad campus and is currently pursuing his PhD with the James Watt School of Engineering at the University of Glasgow, United Kingdom. His research interests include Unmanned Aerial Vehicle (UAV) Assisted Next Generation Wireless Networks, 5G and Beyond Wireless Communication, Machine Learning for Wireless Communication, Software Project Management, Software Project Scheduling, Behavioural Software Engineering, and Global Software Engineering. He is a student member of the IEEE, Professional Member of Association for Computing Machinery (ACM) and a member of the IEEE Antenna Propagation Society (APS).

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.

IVANDRO ORTET LOPES received the B.Sc. in Information System Engineering from Jean Piaget University, Cabo Verde and the M.Sc degree in Information and Communication Engineering from Huazhong University of Science and Technology, China. He is currently pursuing a Ph.D. degree in Cyberspace Security from Huazhong University of Science and Technology. His research interests include Intrusion Detection, Deep Learning, Security in SDN, and NFV..



RAHMAT did his master's in Software Engineering from COMSATS University, Islamabad, Pakistan. Before commencing his PhD studies in 2018, he worked as a Lecturer in the Department of Computer Science at Iqra National University, Pakistan. He has recently completed his PhD degree in the School of Engineering, University of Edinburgh, United Kingdom. He is currently working as a research and development associate at the Faculty of Computing, Engineering and Science, University of South Wales, UK. His research interest includes software project management, machine learning, cloud & cluster computing, digital health and big-health data. His work has beenrecognizedd and published in several high-impact journals and presented at international conferences.