IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 17 February 2023, accepted 5 April 2023, date of publication 10 April 2023, date of current version 26 April 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3265716

RESEARCH ARTICLE

Multi-Objective Dynamic Software Project Scheduling: A Novel Approach to Handle Employee's Addition

NATASHA NIGAR[1], MUHAMMAD KASHIF SHAHZAD[2], SHAHID ISLAM[1], OLUKAYODE OKI[3], (Member, IEEE), AND JOSE MANAPPATTUKUNNEL LUKOSE[3]

[1]Department of Computer Science (RCET), University of Engineering and Technology, Lahore 54000, Pakistan
[2]Ministry of Energy, Power Division, Government of Pakistan, Power Information Technology Company (PITC), Lahore 54000, Pakistan
[3]Department of Information Technology, Walter Sisulu University, Mthatha 5117, South Africa

Corresponding author: Natasha Nigar (natasha@uet.edu.pk)

ABSTRACT The software project scheduling (SPS) problem deals with optimal allocation of employees to tasks such that the project is completed within budget and schedule. This is well addressed in the absence of dynamic events and has been demonstrated in SPS research with the static scenarios only. The software development methodologies have evolved where customer requirements are not initially locked but are flexible to take into account ongoing additional requirements or changes. In this regard, 'new employee addition' has emerged as one of the critical dynamic events that could significantly disrupt the project budget and schedule. In this paper, the SPS is modelled as multi-objective optimization problem and a novel approach is presented to deal with 'new employee addition' dynamic event; that uses domain knowledge to generate a robust schedule. The proposed heuristic is further evaluated on 18 dynamic benchmark scenarios and 3 real-world instances. The results show that the proposed heuristic besides its capability to handle dynamic events also demonstrates good convergence while maintaining good distribution of solutions in a dynamic environment. The principal target audience for this research are project managers in software development organizations who will get optimal project reschedule under 'new employee addition' dynamic event particularly in complex, large scale and multi-project settings.

INDEX TERMS Dynamic software project scheduling, multi-objective optimization, mathematical model, metaheuristics.

I. INTRODUCTION

In SPS, tasks are optimally allocated to the resources for their completion within budget and time under uncertain and dynamic environment [1]. In this regard, decisions are made about who does what during the whole software development life cycle [2] by optimally matching the required skill set against tasks in multi-project settings. The objectives of SPS are to generate an effective (robust and stable) and optimal (cost and duration) schedule under uncertain and/or dynamic scenarios resulting from unpredictable human behaviour while balancing the employees workload [3].

The traditional SPS approaches are based on deterministic and static environment with little place for uncertain or dynamic events [4], [5], [6]. It is assumed that task effort and the skills of each employee are known in advance and remain unchanged [7]; however, in reality, a software project may face hard-to-predict dynamic events that can disturb a software project schedule, effecting the project duration and budget [8]. Consequently, resulting project schedule under uncertain and dynamic events results in late software project deliveries, high development and maintenance costs, and unsatisfied customers [9]. Therefore, as dynamic environment changes, software project managers must react quickly and organizations need to be flexible to respond.

The SPS problem can be solved using critical path method (CPM) [10] and program evaluation and review technique

The associate editor coordinating the review of this manuscript and approving it for publication was Zhenzhou Tang.

(PERT) [10], traditional techniques being used to generate optimal project schedules. However, these techniques may fail due to distinctive characteristics of software projects such as nature of the team, application being built and customers. Moreover, these cannot effectively handle situations in which two or more projects share available resources, and also require subjective information to generate an effective project schedule [11]. Further, to ensure the sustainable software development in an uncertain and dynamic environment, the risk mitigation strategies must be adopted to ensure business continuity. This shows that SPS problem under dynamic environment is challenging.

It is worth noting that the literature about project scheduling with uncertainty is increasing particularly in the domains of Operations Research and Management Science [12], [13], [14]. However, little effort is observed that captures the dynamic events in real-world software projects as these are subject to environmental changes and unforeseen dynamic events e.g. new employee addition, task effort estimation uncertainty, and volatile software requirements [15]. This becomes critical and a limiting factor particularly during complex large scale software projects in multi-project settings [1]. In such case, a deterministic scheduling technique will fail to generate an optimal project schedule that addresses performance deterioration and infeasibility when faced with disruptions [7]. Therefore, it is required to employ dynamic scheduling techniques which takes into account the dynamic events and uncertainties to ensure success of a project. In this regard, search-based software engineering (SBSE) [16] field has excelled. It has reformulated software engineering complex issues as search problems and applies meta-heuristics techniques such as evolutionary algorithms (EAs) and simulated annealing (SA). The SPS is one of those NP-hard combinatorial optimization problems for which EAs have been widely applied [1] and help the project managers for near-optimal software project schedules.

In literature, few studies address the dynamic software project scheduling (DSPS) which is an emerging area in the field of software engineering. The studies in [1] and [7] use a rescheduling approach to deal with software project dynamic events, e.g., employee leave/return and new task arrival. Besides this, Xiao et al. [17] proposed a multi-objective genetic algorithm under disruptive environment such as requirement changes. There is no work which considers 'new employee addition' that occurs as dynamic event consequent upon additional requirement or employee resignation when project rescheduling fails to find the best match between existing employees skill set and remaining tasks in multi-project settings. In this paper, project rescheduling optimally allocates the tasks to available resources and highlights the need for additional employee(s) with target skill set. Consequently, software companies may hire a new employee to replace a leaving team member or to augment the team's capacity [18]. This new team member may also overcome the shortage of existing employees' skills demanded by current projects. In summary, we make the following main contributions:

- New Heuristic Approach: We present a new heuristic (h_NEA) to deal with a new practical dynamic event 'new employee addition' for the SPS problem. This heuristic approach optimizes and reduces project duration and cost with robustness and stability objectives.
- Data Set: The proposed approach is evaluated on 18 benchmark dynamic scenarios and 3 real-world instances.

The paper is structured into seven sections. Section II provides an overview to the background and related work. Section III presents problem formulation and mathematical model. Section IV presents our proposed approach followed by experimental design and experimental results in section V. The limitations to the validity of this study are presented in section VI. This paper is concluded in section VII with future directions.

II. LITERATURE REVIEW

A. BACKGROUND

1) STATE-OF-THE-ART APPROACH

Shen et al. [7] proposed a dynamic version of the SPS problem by formulating a mathematical model with multiple objectives considering dynamic events and uncertainties that often occur during software project development. The proposed model considers one uncertainty and three dynamic events. It deals with tasks efforts uncertainty which implies that modifications in task specifications by the customer or inaccurate initial estimates may cause changes in estimated task effort. Moreover, the customer may raise new requirements request during the software development life cycle. The model also considers that employees can leave or return (from holiday) during the project execution. To handle the uncertainty and dynamic events, Shen et al. [7] also proposed a multi-objective approach based on ϵ-MOEA [19] algorithm. A scenario-based approach is used to handle the task effort uncertainty and very basic heuristic strategies are designed to handle the dynamic events.

The model proposed by Alba and Chicano [2] considers a maximum number of employees assigned to a task to avoid communication overhead; however, it can be violated with a corresponding penalty if task skills are not fulfilled by allocated number of employees. The above literature highlights the fact that there is still space to add more objectives and dynamic features in their work such as new employee addition, changing objectives, task removal, variation in task precedence, and task due-date etc. The scalability of such approaches also requires to be verified.

The software development starts with conception of desired software through to the final manifestation of the software, sometimes in a planned and structured process [20]. During this process, customer may demand change in requirements, some of which may not be fulfilled by using existing employees' skills or an employee may leave. Hence,

it escalates the need for new employee to be hired with those desired skills. Moreover, hiring a skilled and experienced candidate may also reduce training cost. Therefore, a practical dynamic version of the problem is presented which considers 'new employee addition' dynamic event.

2) DECISION MAKER

In this paper, the decision maker selects a single solution from a set of non-dominated solutions among those resulting from evolutionary algorithms. In practice, at each scheduling point t', project manager selects a solution from a set of non-dominated solutions identified by the EA for implementation. However, if the decision to choose the solution for implementation is left at the subjective judgement of project manager, ten it may lead the project drifting from the benchmark KPIs. This becomes more critical in case of complex large scale projects and multi-project settings. Therefore, an automated decision making method proposed in [21] is adopted and its procedure is briefly presented below:

- Step i (Pairwise Comparison Matrix): In this step, a pairwise comparison matrix is constructed. Let's say, there are $N_o = 4$ objectives to be optimized, then there are $N_o \times (N_o - 1)/2 = 4(4-1)/2 = 6$ comparisons. The pairwise comparison matrix $C_1 = (c_{ij})_{N_o \times N_o}$ can be constructed which describes the degree of the preference for one objective versus another.
- Step ii (Weight Vector Estimation): A weight vector $w = (w_i)_{N_o \times 1}$ for multiple objectives is estimated using least squares method [22]. The geometric mean of each row is computed in the matrix C_1, which is then normalized by dividing it with the sum of them.
- Step iii (Objective Values Normalization): Each objective is normalized as:

$$n\_f_i(x) = f_i^{max} - f_i(x)/f_i^{max} - f_i^{min}, \quad i = 1, 2, \ldots, N_o. \tag{1}$$

  where f_i^{max} and f_i^{min} denotes the maximum and minimum objective values among all the non-dominated solutions obtained at the current scheduling point.
- Step iv (Utility Value Calculation): The utility value for each non-dominated solution is found using weighted geometric mean of the multiple objective values as follows:

$$U(x) = \prod_{i=1}^{N_o} n\_f_i(x)^{w_i / \sum_i^{N_o} w_i} \tag{2}$$

- Step v (Solution Selection): The solution with maximum utility value is selected as the final schedule.

Notably, the pairwise comparison matrix and the weight vector determined in Steps i and ii are calculated in advanced and is not changed during the dynamic process. Only Steps iii, iv, and v are performed at each scheduling point during the project execution.

Following example explains this procedure in detail. Let's say that there are four objectives to be optimized namely, project duration, cost, robustness, and stability. The software project manager gives equal importance to duration and cost objectives, and considers that robustness and stability are of the equal importance. Thus, the pairwise comparison matrix for the four objectives is constructed as follows:

$$C_1 = (c_{ij})_{4\times 1} = \begin{bmatrix} 1 & 1 & 2 & 2 \\ 1 & 1 & 2 & 2 \\ 1/2 & 1/2 & 1 & 1 \\ 1/2 & 1/2 & 1 & 1 \end{bmatrix}$$

The weight vector $w = (w_i)_{4\times 1} = [0.3333, 0.3333, 0.1667, 0.1667]^T$ can be obtained according to the above Step ii. After that each objective is normalized according to (1), and the utility value for each non-dominated solution can be calculated using (2). Then, the non-dominated solution with the highest utility value is chosen as final schedule.

B. RELATED WORKS

The human resources play a crucial role in the successful completion of software projects. Therefore, ensuring the suitable and correct assignment of employees to tasks is an immediate requirement. The software projects go through various unforeseen disruptions and dynamic events. Table 1 presents the significance of frequently occurring dynamic events in literature.

To address the DSPS problem, Hepke et al. [27] presented a fuzzy model for the SPS system. In this model, activity time parameters were modelled by means of L-R fuzzy numbers due to uncertainty, and the fuzzy problem was transformed into a set of associate deterministic problems. A simulation based method aiming to limit the impact of uncertainties on the overall project success, was proposed by Lazarova-Molnar and Mizouni [28] that selects the most appropriate remedial action based on the project goal. Gueorguiev et al. [29] proposed a multi-objective evolutionary algorithm (MOEA) to provide robustness under uncertainty for rescheduled software project. The robustness is defined as the difference between the total time and estimated time when tasks duration are extended or new tasks are added. The Pareto front found by their proposed MOEA represents the trade-off between project completion time and robustness. Antoniol et al. [30] used a tandem genetic algorithm (GA) to process work packages in best sequence and allocate the staff to project teams in optimal fashion. The results obtained by GA were analysed by a queuing simulator which guides the search process to determine whether a negotiation of further employees and successive iteration of the tandem GA process was required or not. To obtain an optimal solution, the whole process repeat multiple times but their work does not consider stability and other dynamic events.

Ge [31] proposed a rescheduling method under uncertainty that deals with two objectives (efficiency and stability) and is based on GAs; however, both objectives were handled as a single objective function through weighted sums. Xiao et al. [17] work deals with resource management in disrupted

TABLE 1. Significance of dynamic events in literature.

Dynamic events	Significance
Task-related 1. Task addition 2. Task deletion 3. Task effort uncertainty	- Agile 2^{nd} principle [23] is: "Welcome changing requirements, even late in development". - Reasons for Late Software Delivery by Pressman [23]. "Changing customer requirements that are not reflected in schedule changes." - Requirements uncertainty/velocity [24], [25] - Underestimation of the effort amount and/or the required number of resources to do the job, by Pressman [23].
Employee-related 4. Employee addition 5. Employee resigns 6. Employee leave 7. Employee return	- Staff turnover by Boehm [26] - Human difficulties that seem not have been predicted, by Pressman [23]

environment e.g. requirement changes and urgent bug fixing. They used Little-JIL process definition language [32] to define software project activities, activities dependencies, and need for resources. The authors did not consider continuous rescheduling and incorporates capability constraints, availability constraints, and activity execution order constraints only. Other constraints, such as different activities demanding the same resource in multi-project settings are not considered.

Chicano et al. [33] proposed a new multi-objective formulation considering employees' productivity at performing different tasks for project scheduling. Their solution proved to be robust against inaccuracies in the task-cost estimations. Chand et al. [34] addressed resource-constrained project scheduling problem (RCPSP) with stochastic activities duration and resource availability under uncertainties as a single-objective problem. Shen et al. [7] proposed a mathematical model with four objectives and three dynamic events (employee leaving/returning and new task addition) using a proactive-scheduling method which is presented in detain in next section. Shen et al. [8] also proposed a mathematical model for large-scale projects with similar dynamic events. As an improvement of their work [1], authors proposed a memetic algorithm dealing with five objectives and aforementioned dynamic events. Cheng et al. [35] considered tasks rework, dynamic skill proficiency, and employee leave and return. Ge and Xu [36] proposed a GA and hill climbing (HC) based optimizer. In their software project staffing model, authors considered dynamic elements of staff productivity as well as both stability and efficiency as a single objective function through weighted sum. However, the proposed model does not incorporate the interactions between parallel tasks. Besides these, some studies [15], [37] provide a survey of research on the SPS problem.

The above literature highlights the fact that static approaches lack the consideration of unpredictable scenarios while dynamic ones are also restricted to employee leaving and returning as events. In real-world, many dynamic events could occur which make the scheduling problem highly complex. To resolve these issues, a more practical dynamic version of problem should be considered.

III. DYNAMIC SOFTWARE PROJECT SCHEDULING

A. GENERIC PROBLEM FORMULATION

Managing human resources and total budget in an optimal way for a successful project is critical. Suppose a software company has 5 employees $E = \{e_1, e_2, e_3, \ldots.., e_5\}$ with employee's skills set $SK = \{s_1, s_2, s_3, s_4, s_5\}$. They need to develop a software application comprising of 7 tasks $T = \{t_1, t_2, t_3, \ldots.., t_7\}$. The term $e^{skill} \subseteq SK$ denotes an employee's skill and e^{maxded} represents maximum dedication to the project, which is defined as the ratio of amount of hours dedicated to the project divided by full length of the employee's working day [2]. The salary e^{salary} is expressed in fictitious currency units. This is explained by a simplified scenario as shown in Fig. 1. The employee e_1 is a programmer (s_1) and UML (s_2) expert with salary \$ 3,000. Another employee e_2 has UML (s_2), testing (s_3), and data modelling (s_4) expertise whereas e_4 is also an UML expert (s_2). The employees e_4 and e_5 also have leadership skills (s_5). The employees e_1, e_2, e_5 can spend all their day on the project with maximum dedication equal to one. Another employee e_3 with programming experience (s_1) dedicates half of his/her working day developing the software application. This employee may be working part time or he/she is looking after some administrative tasks as part of their dedication. The employee e_4 can work overtime with his/her maximum dedication greater than one, e.g., ($e^{maxded} = 1.3$). This implies that he/she can work on the project up to 30% more than in a normal working day.

Each task is associated with certain required skills denoted as t^{skills} and an effort t^{effort} expressed in person-month (PM). The tasks are performed according to a task precedence graph (TPG). This indicates which tasks must be completed before a new task begins. TPG is an acyclic directed graph $G(T, A)$ to represent the dependency among tasks. In TPG, the set of nodes represents the set of tasks T. The precedence relation among the tasks is denoted by set of arcs A. In the given example, task t_1 (design model document) requires UML expertise (skill s_2) and 3 person-months effort to complete the subsequent tasks as illustrated in Fig. 2. In the same figure, it can also be observed that the last task t_7 (User manual creation) cannot be instantiated until all the previous tasks are completed.

e_1^{skills} = {s1, s2}	e_2^{skills} = {s2, s3, s4}	e_3^{skills} = {s1}	e_4^{skills} = {s2, s5}	e_5^{skills} = {s3, s5}
e_1^{maxded} = 1.0	e_2^{maxded} = 1.0	e_3^{maxded} = 0.5	e_4^{maxded} = 1.3	e_5^{maxded} = 1.0
$e_1^{salrary}$ = $3,000	$e_2^{salrary}$ = $4,000	$e_3^{salrary}$ = $1,500	$e_4^{salrary}$ = £3,000	$e_5^{salrary}$ = $2,500

FIGURE 1. Example of software company employees with different skill set, salary, and dedication level.

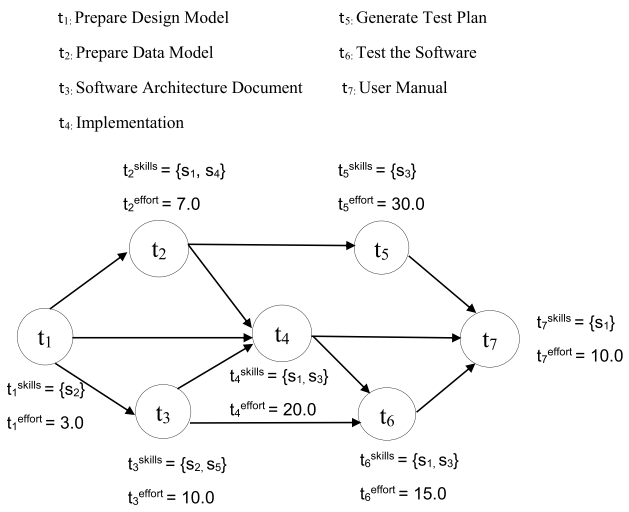


FIGURE 2. Task precedence graph of the software application.

Dynamic feature - new employee addition: Once software project has started, a new employee may join it. The software development is an ongoing process and requirement change is inevitable [20]. Hence, a request for software requirement change may be raised by the customer due to enhanced understanding of software, organizational restructure, and availability of new technologies. For example, the requirement change needs some special skills which none/very few of existing employees have. In this case, the software company hires a new employee having those skills. The new employee may be proficient in all or some skills and has higher/lower salary as compared to existing employees. Further, it is assumed that new employee is hired somewhere in the middle of project, and does not violates Brook's law which states that adding manpower to late project makes it later [38].

Employee's proficiency calculation: The proficiency of an employee e_i for a task t_j can be defined as the employee's capability level to perform that task. It is denoted as e_{ij}^{prof} and ranges between [0,1]. According to Chang [39] calculation, if an employee has zero proficiency against a skill related to a task then employee's total proficiency becomes zero, which should not be a case. In reality, if employee's proficiency for one skill is zero while having other skills for a task, then employee could still perform that task. Therefore, in this work, we define employee to task proficiency as:

$$e_{ij}^{prof} = \sum_{k \in req_j} \frac{prof\_value_i^k}{C} \tag{3}$$

The term req_j (3) denotes task required skills and each skill has a proficiency value against it. The C is a constant and is set '$C = 5$' according to [39]. The average proficiency is computed as follows:

$$e_{ij}^{prof} = \frac{e_{ij}^{prof}}{total\_skills\_required\_by\_task} \tag{4}$$

B. SOLUTION REPRESENTATION

To represent the SPS problem solution, a dedication matrix $X = (x_{ij})$ of size m x n where $x_{ij} \geq 0$ is used. The element m represents number of employees, n denotes number of tasks, and x_{ij} denotes the degree of dedication of an employee e_i to task t_j. An employee e_i allocated on a task t_j with dedication degree of '1' means that he/she spends his/her all working hours of a day to that task. If dedication degree is '0', it indicates that employee will not perform that task. The project duration and starting and finishing time of each task is calculated using this information. According to the TPG and dedication matrix, a Gantt chart of the project is drawn as shown in Fig. 3. The task's duration are calculated according to formulation in [2]. Once duration of project is calculated, project cost using the dedication matrix and employees' salary is computed.

C. OBJECTIVE FUNCTIONS

In this DSPS optimization problem, two objectives are considered as duration and cost to see how our proposed approach (h_NEA) works for new dynamic event as compared to

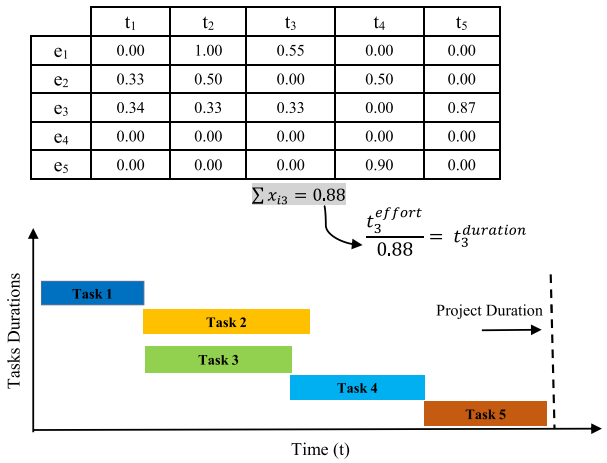


FIGURE 3. Tentative solution to the SPS problem.

current state-of-the-art algorithm [7]. The mathematical formulation and detail of each objective is according to [7].

$$\min F(t) = [f_1, f_2, f_3, f_4] \tag{5}$$

where f_1, f_2, f_3, f_4 denotes duration, cost, robustness, and stability as project objectives respectively.

Duration: Project duration is time required to complete the project and is taken as the maximum finishing time of the last task.

$$f_1(t) = duration_I = max(T_j^{end}) - min(T_j^{start}) \tag{6}$$

The terms T_j^{end} and T_j^{start} denote the starting and ending time of each task. The end time for tasks without precedence is computed by adding the start time to respective duration of each task. The start time of a task with precedence is the end time of longest corresponding preceding task. The dependency between tasks is represented using the TPG. Moreover, duration (6) for the whole project is the maximum finishing time of last task.

Cost: Project cost is the total expenses required for the project completion.

$$f_2(t) = cost_I = \sum_{e_i \in E\_ava\_set} e\_cost_i \tag{7}$$

The term E_ava_set (7) represents the set of available employees. Suppose an employee e_i is assigned to a task t_j with x_{ij} dedication. The employee's normal monthly salary is denoted as $e_i^{norm\_salary}$. If employee's dedication x_{ij} is greater than 1, it indicates that he/she overworks for the project and consequently, overtime salary $e_i^{overwork\_salary}$ is also added to the total salary. The expenses paid to an employee e_i at t' month are calculated as follows:

$$if \sum_{j \in T\_active\_set} x_{ij} \leq 1.0$$

$$e\_cost_i = e_i^{norm\_salary} \cdot t' \cdot \sum_{j \in T\_active\_set} x_{ij} \tag{8}$$

$$if \sum_{j \in T\_active\_set} x_{ij} > 1.0$$

$$e\_cost_i = e_i^{norm\_salary} \cdot t' + e_i^{overwork\_salary} \cdot t' \cdot \left(\sum_{j \in T\_active\_set} x_{ij} - 1 \right) \tag{9}$$

The term T_active_set (8-9) represents the set of active tasks which are being developed at time moment t', where t' denotes the current project development month. At time t', a task that does not have any preceding unfinished task is known as active task. Further, subscript I (6-7) represents the initial scenario without any uncertainty in the tasks efforts. It means that initially estimated tasks efforts are correct.

To analyse the effect of our approach to other objective values, four objectives are considered including robustness and stability with assumption that improvement in duration and cost objectives may improve or deteriorate other objectives.

Robustness: It is project schedule's ability to cope with the task effort uncertainties for the software project.

$$f_3(t) = \sqrt{\frac{1}{N}\sum_{q=1}^{N}\left(max\left(0, \frac{duration_q(t') - duration_I(t')}{duration_I(t')}\right)\right)^2} + \sqrt{\frac{1}{N}\sum_{q=1}^{N}\left(max\left(0, \frac{cost_q(t') - cost_I(t')}{cost_I(t')}\right)\right)^2} \tag{10}$$

The terms $duration_I$ and $cost_I$ (10) are the initial duration and cost calculated in equations (6) and (7) respectively, with assumption that there is no task effort variance. To cope up with the task effort uncertainties, a scenario-based approach is used. We sample a set of task effort scenarios $\{Q_q \mid q = 1, 2, 3, \ldots .N\}$, where N is sample size. In our case, $N = 30$. The terms $duration_q$ and $cost_q$ are efficiency objectives under value Q_q. A 'max' function is used to truncate the variances of duration and cost decreases from initial value. The sampled effort for task $T_j^{effort_q}$ is calculated using the normal distribution [1].

$$T_j^{effort_q} = T_j^{effort_{mean}} + T_j^{effort_{varaince}} * z \tag{11}$$

The term 'z' (11) is a random scalar drawn from the normal distribution. At scheduling point t', $T_j^{effort_q}$ is calculated multiple times until the condition $T_j^{effort_q} > T_j^{finished\_effort}$ is satisfied and then set $Q_q = \{T_j^{rem\_effort_q} \mid T_j^{rem_effort_q} = T_j^{effort_q} - T_j^{finished\_effort}\}$ where $T_j^{rem\_effort_q}$ means the q_{th} sampled remaining effort of T_j at scheduling point t'.

Stability: The purpose of this objective is to measure the deviation between new and initial project schedule. The term t' represents the time when a dynamic event occurs and t indicates initial schedule time. It ensures that at each scheduling point, there is not too much variation in employees'

assignment as compared to initial schedule.

$$f_4(t) = stability = \sum_{i=1}^{E} \sum_{j=1}^{T} \mid x_{ij}(t') - x_{ij}(t) \mid * Penalty_{ij} \quad (12)$$

where the values of $Penalty_{ij}$ are set as follows:

$$\begin{cases} 2, & \text{if } x_{ij}(t') > 0 \text{ and } x_{ij}(t) = 0, \\ 1.5, & \text{if } x_{ij}(t') = 0 \text{ and } x_{ij}(t) > 0, \\ 1, & \text{else.} \end{cases} \quad (13)$$

These penalty values are according to [7]. In first case, a large penalty value of '2' is given if employee e_i performs a new task t_j at scheduling point t'. He/she may need additional time to know about task specifications. In such scenario, the employee's efficiency level to do work may be decreased. In second case, a medium penalty of '1.5' is given if employee is not allocated to the same task for which he was working in initial schedule. The employee might have received training and such training would be fruitless if employee is not performing that task anymore. A small penalty of '1' is given if a task is performed with a different dedication level.

D. CONSTRAINTS

The DSPS problem is subject to following constraints [7]. The (i)-(iii) are hard constraints whereas (iv) is a soft constraint.

i) All tasks allocation constraint: Each task should be allocated to at least one available employee. The term T_j_Dedication represents employees total dedication for that task.

$$\forall e_i \in \text{E \_ava \_set(t)}, \quad \sum T_j\text{\_Dedication} \neq 0 \quad (14)$$

ii) Task skills constraint: All the available employees allocated on a task must collectively cover all the skills required by that task.

$$T_j\text{\_Skills} \subseteq \cup e_i \{ \text{ Skills} \mid x_{ij} > 0 \} \quad (15)$$

iii) No overwork constraint: No employee overworks for a project means that employee should not work for the project more than his/her maximum dedication.

$$\sum_{j \in T} x_{ij} \leq e_i^{max_d ed} \quad (16)$$

iv) Task head count constraint: According to the best practices in software engineering, the number of employees assigned on a task should not exceed a limit [40]. Each task has a maximum number of employees in this DSPS problem. The term $T_j^{no\_of\_emp}$ denotes the number of allocated employees on a task and $T_j^{max_h eadcount}$ represents task maximum headcount and is computed using the formula in [26].

$$\forall T_j, T_j^{no\_of\_emp} \leq T_j^{max_h eadcount} \quad (17)$$

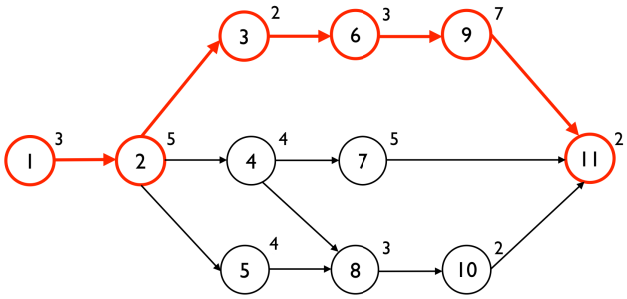


FIGURE 4. Example of critical path.

IV. THE PROPOSED HEURISTIC

This section describes a heuristic method (h_NEA) to deal with a new dynamic event i.e. 'new employee addition', in order to address the shortcomings in an existing state-of-the-art algorithm [7] that they are unable to handle other real-events. We make an assumption here that new employee is added somewhere in the middle of project; however, Brook's law does not apply which states that adding manpower to late project makes it later [38].

A. HEURISTIC METHODS

The objective of the heuristic method is to produce near-optimal solutions in a reasonable time frame that solves the SPS problem for 'new employee addition' dynamic event. The SPS is an NP-hard combinatorial optimization problem [2]. The results about NP-hard problems make heuristics the only viable option for complex optimization problems that need to be solved in real-world applications [41]. These are designed for solving a problem where classic methods are too slow or may fail when finding an approximate solution. The proposed heuristic (h_NEA) is divided into duration and cost heuristic. The purpose of duration heuristic is appropriate allocation of new employee to available critical tasks to reduce project duration whereas cost heuristic allocates new employee with an objective to minimize the project cost.

1) DURATION HEURISTIC

For new employee joining the software company, the purpose of duration heuristic is to reduce project duration. Detailed steps are given below:

Step1: Identify the critical path in the TPG. The TPG indicates that a task should finish before starting a new preceding task. The complexity of each task is associated with it. The critical path (Fig. 4) consists of all the tasks, in which any kind of task delay will ultimately delay the whole project. This determines the shortest time possible to complete the project. The tasks in critical path are dependent on each other with zero float [10].

Step2: The new employee is allocated to a critical task with the maximum dedication as long as his/her proficiency to that task is not zero.

Let us consider an example in Fig. 4, where there are eleven tasks with corresponding TPG and duration. The tasks

Algorithm 1 Duration Heuristic

Input : e_n: new employee, T: set of available tasks, X: dedication matrix, $G(T, A)$: Task precedence graph
Output: D_{hs}: Heuristic solutions with minimized project duration
for $\forall T_j \in T$ **do**
  CT = critical tasks /* a subroutine findCriticalTasks() is called which finds all critical tasks using task precedence graph */
end
for $\forall c_j \in CT$ **do**
  if $e_{nj}^{prof} > 0$ **then**
    x_{ij}: assign employee e_i to c_j with maximum dedication /* a heuristic solution is generated */
  end
end
return D_{hs}

Algorithm 2 Cost Heuristic

Input : E: set of available employee, e_n: new employee, T: set of available tasks, X: dedication matrix
Output: C_{hs}: Heuristic solutions with minimized project cost
for $\forall T_j \in T$ **do**
  for $\forall e_i \in E$ **do**
    if $e_n^{prof} > e_i^{prof}$ &$e_n^{salary} < e_i^{salary}$ **then**
      $x_{ij} \leftarrow 0$
      if $Task\_skill\_constraint(x_{ij}) = 0$ **then**
        $x_{nj} \leftarrow \sum x_{ij}$
      end
      if $Employee\_overwork\_constraint(x_{nj}) > 0$ **then**
        $x_{nj} \leftarrow e_n^{maxded}$
      end
    end
    if $e_n^{prof} < e_i^{prof}$ &$e_n^{salary} < e_i^{salary}$ **then**
      $x_{nj} \leftarrow x_{ij}$
      $lowerTaskcost(e_n, e_i)$ /* a subroutine is called */
      return e_{lower_cost}
    end
    if $e_n^{prof} > e_i^{prof}$ &$e_n^{salary} > e_i^{saalry}$ **then**
      $x_{nj} \leftarrow x_{ij}$
      $lowerTaskcost(e_n, e_i)$ /* a subroutine is called */
      return e_{lower_cost}
    end
  end
end
return C_{hs}

{1, 2, 3, 6, 9, 11} are critical with zero float. The red color path shows the critical path. Hence, if the new employee is allocated to these tasks in the critical path with maximum dedication, it will reduce the project duration. For each critical task, a new heuristic solution is generated depending on new employee proficiency for that task. For duration heuristic, 'task head count' constraint is checked at algorithmic level. The pseudo code for the duration heuristic is given in Algorithm 1.

2) COST HEURISTIC

This heuristic method (Algorithm 2) is designed in such a way to optimally allocate new employee to available tasks and reduce the project cost. According to [20], software engineer's salary (a.k.a effort cost) is the main attribute that will have impact on project cost. Besides, if employee's proficiency is higher for a task, employee will finish that task in a shorter time. Based on these facts, three different cases are considered to generate cost heuristic solutions as shown in Fig. 5. For each available task, a new heuristic solution is generated depending on the cases stated below:

- **Case a**: If new employee's proficiency level is higher and salary is lower than all the existing employees, then replace all the employees who can be replaced (without violating task skills constraint) with new employee. The new employee's dedication is the sum of dedications of replaced employees. If new employee's dedication exceeds the employee's maximum dedication, then 'no employee overwork' constraint is considered here as well.
- **Case b**: If new employee's proficiency level is lower and salary is also lower, then select between the existing and new employee who has lowest cost to perform a task. In this case, dedication for new employee and old employee is same.
- **Case c**: If new employee's proficiency level is higher and salary is also higher, then select between the existing and new employee who has lowest cost for performing a task. In this case, dedication for new employee and old employee is also same.

For the cost heuristic, new employee is replaced with the existing one if this replacement causes a significant difference in the project cost. It is assumed that this difference is 5% of the total cost. If difference is too small, there will be no benefit for replacement as new employee may not be proficient for that task as the old one.

B. MOEA-BASED METHOD FOR DSPS PROBLEM

A set of Pareto optimal solutions in a well-converged and well-distributed form, is an important issue in multi-objective evolutionary optimization. Many evolutionary algorithms do not use domain knowledge to generate the initial population.

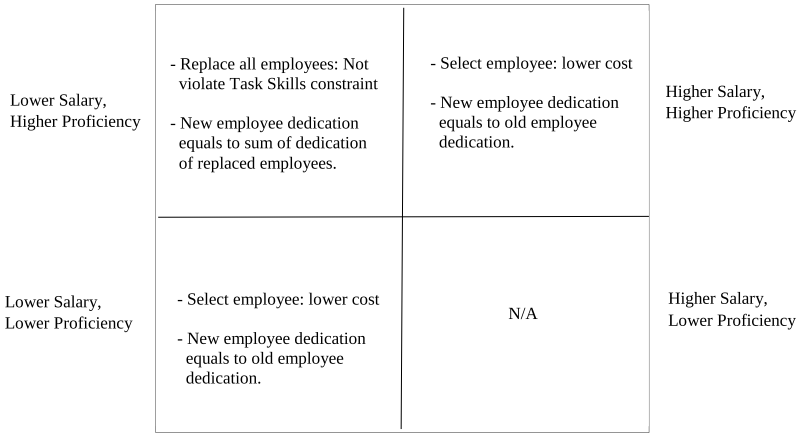


FIGURE 5. Cases for cost heuristic.

It is known fact that good initial estimates may generate better solutions with faster convergence depending on the prior knowledge existence, or if it can be generated at a low computational cost [42]. Therefore, in our approach, we design a heuristic method (h_NEA) using domain knowledge for population initialization. It is based on the concept of ϵ-dominance [19] to deal with 'new employee addition' as new dynamic event. The procedure of the algorithm at scheduling point t' is given in Fig. 6. The initial population is formed using the following combination of solutions in step 1 of the algorithm.

$$50\%Heuristic + 1\%Baseline + 49\%Random \tag{18}$$

In our study, 50% are duration heuristic and cost heuristic solutions produced by Algorithm 1 and Algorithm 2 respectively; and its variants using mutation operator. 1% is initial schedule/baseline/history solutions and 49% are random solutions. We are using trade-off between random initialization and heuristic procedure. Random initialization will give diversification on the generated solutions while heuristics/baseline solution will converge the solutions rapidly. At initial time t_0, when the project starts, the initial population is generated using 100% random solutions in the initial step of algorithm and only duration, cost, and robustness as project objectives are solved.

V. EXPERIMENTAL RESULTS

This section answers the following research question by empirically investigating the proposed method:

- RQ1. Are existing methods capable to deal with 'new employee addition' dynamic event efficiently?
- RQ2. Does proposed approach generate effective project plans in terms of duration and cost when dealing with 'new employee addition' dynamic event in contrast to existing approach?
- RQ3. Does the improvement on objectives (duration and cost) made by the proposed heuristics compromise other objectives, e.g., project robustness and stability?

A. EXPERIMENTAL SETUP

1) DSPS INSTANCES

In this study, we use 18 dynamic benchmark and 3 real-world data instances.

a: BENCHMARK

These 18 dynamic instances are derived from Alba and Chicano's benchmark [2] which are gathered from different software projects. The reason for choosing this benchmark data set is that these instances include variants of three important factors (number of employees, number of tasks and number of employee skills) for the DSPS problem as in real-world scenario. To induce more reality, the dynamic data instances differentiate themselves from the static ones in [2] with the following keys aspects: task maximum headcount, task effort uncertainties, part-time jobs, and overworking of employees. In the project, it is assumed that part-time employees are 20 percent of the total employees whose maximum dedications are in the interval [0.5, 1); employees doing overtime work are 20 percent, whose maximum dedications are generated uniformly from (1, 1.5] at random; and remaining employees are full time, their maximum dedication is set to 1.0. If an employee possesses a skill, then relevant proficiency score for that skill is sampled uniformly from (0, 5] in a random manner. If employee does not have any specific skill, then proficiency score is set to 0 for that skill. Following the practice in [2], an employee's normal monthly salary is sampled from a normal distribution with the mean of 10,000 and standard deviation of 1,000.

Further, tasks efforts variances are assumed to follow a normal distribution. These tasks efforts are calculated depending on different values of mean and standard deviation; and vary uniformly in interval [8], [12] and [4], [6] respectively. On average, the mean of a task effort is 10 and the standard deviation is 5 [7].

b: REAL-WORLD

Three real-world instances derived from business software construction projects for a departmental store [43] are

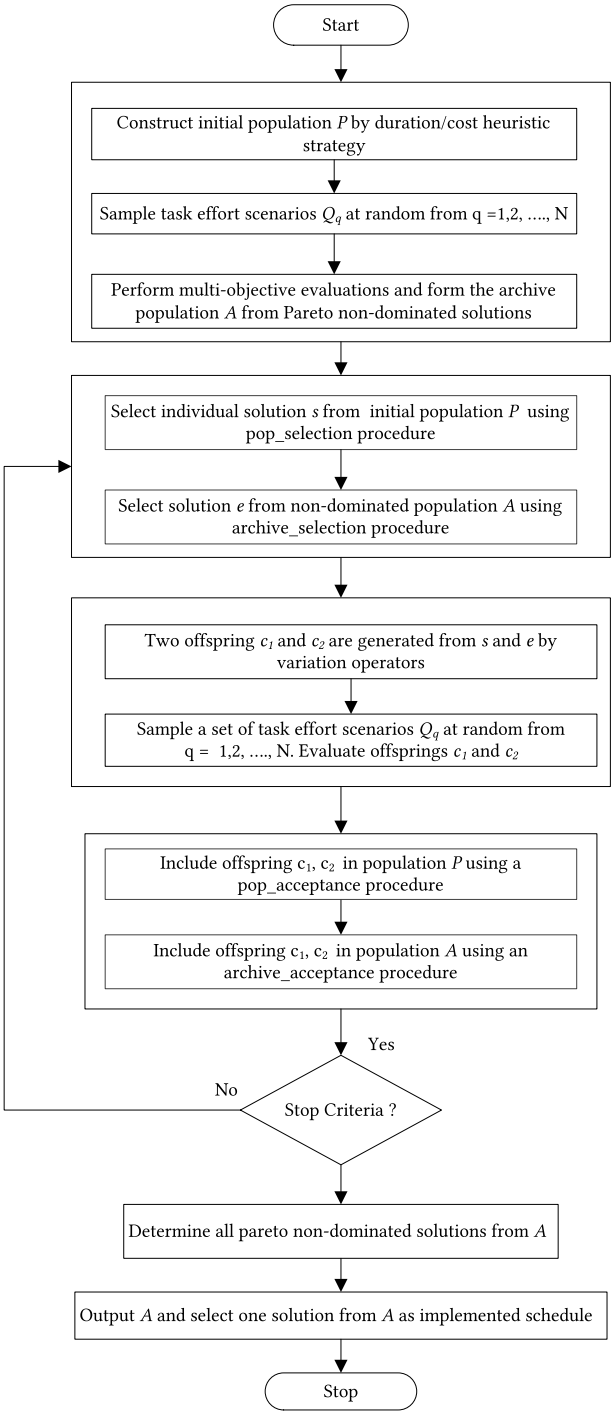


FIGURE 6. Procedure of algorithm at scheduling point *'t''*.

also used in our experiments. These instances are having 10 employees (each employee with 5 different skills) and 15 tasks each except Real_3 which consists of 12 tasks.

The data instances are denoted as 'T30_E5_SK6-7', whereas the notations 'T30' and 'E5' represent total number of tasks and number of employees respectively. 'SK6-7' means number of skills possessed by an employee in the project. The 3 real-world instances are named as Real_1, Real_2 and Real_3.

Currently, we do not have access to real-world software project data containing information about their dynamic events. In this work, we have used data instances with simulated dynamic and uncertain events which is threat to validity of this study. Once real-world data with known dynamic events become available, further analyses should be performed.

TABLE 2. Parametrization (L = Individual length).

State-of-the-art algorithm [7]	
Population size	100 individuals
Crossover	SBX, pc = 0.9
Mutation	polynomial, pm = 1.0 / L
No of evaluations	10,000
Proposed approach (h_NEA)	
Population size	100 individuals
Crossover	SBX, pc = 0.9
Mutation	polynomial, pm = 1.0 / L
No of evaluations	10,000

2) PARAMETER SETTINGS

A set of parameter values for a rational comparison among algorithms are presented in Table 2.

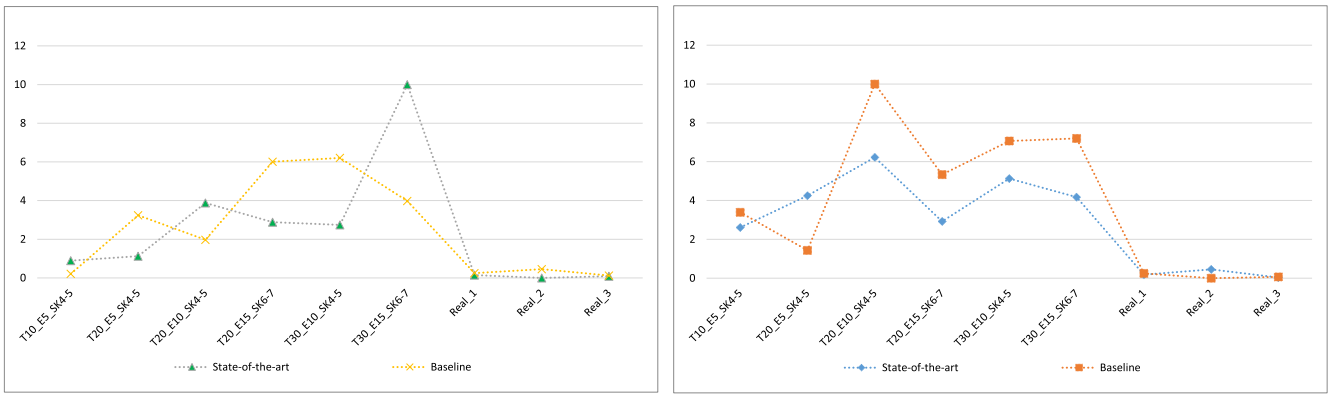
In this study, for each algorithm on each problem instance (18 benchmark and 3 real-world), 30 independent runs were executed to obtain all the results with the termination criterion of 10,000 evaluations.

3) EVALUATION OF SOLUTION QUALITY

In this study, hypervolume (HV) indicator is used to compare algorithms' performance. The HV is used for the problems whose Pareto front are unknown as in our case. Therefore, it is more suitable in real-world scenarios as compared to other indicators [44]. Algorithms with higher HV values are desired. Here, we use a normalised HV value (also called hypervolume ratio (HVR)). With this normalisation, the range of all the obtained results is [0, 1], where 1 represents the optimal value. In the calculation of HV, the reference point determination is a crucial issue. To calculate the reference point, we extract the maximal values for all objectives from the Pareto approximations found by all algorithms used in comparison. In addition, the reference point should be set to slightly worse than the maximal value on each objective of the estimated Pareto front (e.g., 1.1 times of it) [44]. In experiments, single employee and one rescheduling point is considered.

B. PERFORMANCE ANALYSIS OF EXISTING ALGORITHMS

This section analyses that whether existing algorithms can handle 'new employee addition' dynamic event efficiently. Both state-of-the-art (SOA) [7] and baseline algorithm for new dynamic event are investigated. Baseline algorithm is a MOEA-based complete rescheduling method [45] and generates initial population randomly at each scheduling point



(a) Duration Objective

(b) Cost Objective

FIGURE 7. State-of-the-art and baseline algorithm performance for 'new employee addition' dynamic event.

TABLE 3. Duration and cost values comparison for data instances. The selected solution against each instance is according to decision maker result [21].

	Real_1		Real_2		Real_3	
	SOA	Baseline	SOA	Baseline	SOA	Baseline
Duration	**4.57**	5.06	**3.31**	4.53	3.80	**3.56**
Cost	**1.15E+05**	1.72E+05	3.55E+04	**2.84E+04**	**8.40E+04**	1.02E+05
	T10_E5_SK4-5		T20_E10_SK4-5		T30_E10_SK4-5	
	SOA	Baseline	SOA	Baseline	SOA	Baseline
Duration	2.72E+01	**2.17E+01**	**4.72E+01**	7.38E+01	**3.95E+01**	5.32E+01
Cost	**1.51E+05**	5.26E+05	**1.11E+06**	2.16E+06	**1.53E+06**	3.43E+06

whereas SOA also includes the history solutions at the scheduling point. The use of baseline algorithm implies that new schedule is generated from scratch. The comparison results for some data instances are depicted in Fig. 7. These data instances are choosen randomly. The objective values are normalized to be on the same scale. It is obvious that there is no clear pattern of results for both algorithms to deal with new dynamic event. For some instances, baseline outperforms state-of-the-art algorithm and vice versa. Furthermore, the objective values against data instances returned from both algorithms for 'new employee addition' dynamic event are presented in Table 3. It is not clear from results that which algorithm performs better for both duration and cost objectives. All these key factors necessitate the need for a new approach to deal with this event.

Since EAs provide a set of non-dominated solutions to solve NP-hard problem [46]. Therefore, only single solution among feasible solutions is presented, according to decision maker (DM) choice [21]. In practice, a non-dominated solutions set found by EAs, is provided to the software project manager subject to manager's choice. The project manager may select a solution considering best duration, best cost or trade-off among all objectives. However, the involvement of a person for taking decisions is not practical in our experiments; hence, an automated decision making method [21] is adopted.

C. COMPARISON OF PROPOSED HEURISTIC

The purpose of this comparison is to investigate that how significantly our proposed heuristic (h_NEA) deals with 'new employee addition' dynamic event. To show the effectiveness of our proposed approach, we present heuristic solutions produced by h_NEA for Real_1 and Real_2 data instances. We compare it with a state-of-the-art algorithm [7] using two objectives duration and cost. Since the baseline algorithm generates the initial population from scratch, therefore, we only choose state-of-the-art algorithm [7] for comparison. For a fair comparison, the baseline/history solution (implemented schedule) for both algorithms is same. Also, the set of employees is the same for both algorithms. For h_NEA, at scheduling point t', the population is composed of 1% history/baseline solution (implemented schedule), 50% heuristic solutions and 49% random solutions. For

TABLE 4. Real_1 data instance: h_NEA vs existing algorithm solutions.

h_NEA		Existing Algorithm [7]	
Duration	Cost	Duration	Cost
History Solution		History Solution	
4.20E+00	1.69E+05	4.20E+00	1.69E+05
Duration Heuristic Solutions		Existing Algorithm Solutions	
4.18E+00	1.70E+05	4.31E+00	1.68E+05
4.19E+00	1.66E+05	4.20E+00	1.69E+05
4.16E+00	1.67E+05	4.20E+00	1.69E+05
Cost Heuristic Solution		4.20E+00	1.69E+05
4.32E+00	**1.68E+05**	4.19E+00	1.69E+05
4.20E+00	**1.66E+05**		

TABLE 5. Real_2 data instance: h_NEA vs existing algorithm solutions.

h_NEA		Existing Algorithm [7]	
Duration	Cost	Duration	Cost
History Solution		History Solution	
2.44E+00	6.05E+04	2.44E+00	6.05E+04
Duration Heuristic Solutions		Existing Algorithm Solutions	
2.42E+00	6.00E+04	2.29E+00	5.82E+04
2.39E+00	5.88E+04	2.58E+00	6.29E+04
Cost Heuristic Solution		2.44E+00	6.05E+04
2.16E+00	**5.88E+04**	2.45E+00	6.06E+04
2.20E+00	**5.74E+04**	2.64E+00	6.38E+04
2.20E+00	**5.65E+04**		

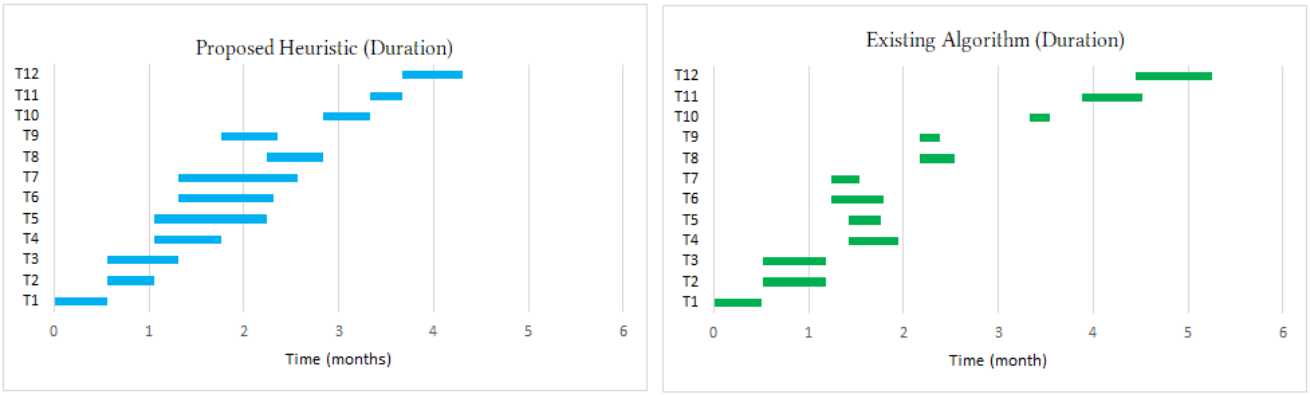


FIGURE 8. Gantt chart for Real_3: h_NEA vs existing algorithm. The selected solution is returned by decision maker result [21].

state-of-the-art algorithm, the distribution is 1% history/ baseline solution, 50% history variants and 49% random solutions.

The duration and cost heuristics optimize protect duration and cost by appropriate allocation of employees to the tasks. The results indicate (Table 4 and Table 5) that proposed approach significantly reduces both duration and cost objectives as compared with the existing ones.

For the sake of simplicity, the gantt chart is also drawn from the results of both algorithms in Fig. 8 for Real_3 data instance. The results indicate that proposed method returns shorter project duration as compared to state-of-the-art algorithm. Based on HVR values in Table 6, it is also obvious that h_NEA returns higher HVR value for 80% data instances and results are better.

Wilcoxon's rank sum test [47] is also performed. It gives statically sound conclusion that results are significantly different from each other. The notation "†" (Table 6) indicates that both algorithms are significantly different from each other at 0.05 significance level.

TABLE 6. Mean value of HVR Indicator - 2 objectives, best mean is in boldface.

Instance	h-NEA	Existing Algorithm [7]
T10_E5_SK4-5	**3.99E-01** †	3.81E-01
T10_E10_SK4-5	**2.33E-01**†	2.09E-01
T10_E15_SK4-5	**5.22E-01**†	3.76E-01
T10_E5_SK6-7	5.77E-01 †	**5.92E-01**
T10_E10_SK6-7	**5.77E-01**†	5.38E-01
T10_E15_SK6-7	**5.60E-01**†	5.20E-01
T20_E5_SK4-5	**6.23E-01**†	6.01E-01
T20_E10_SK4-5	**6.00E-01**†	5.62E-01
T20_E15_SK4-5	**2.67E-01**†	2.52E-01
T20_E5_SK6-7	**3.81E-01**†	3.69E-01
T20_E10_SK6-7	4.07E-01†	**4.10E-01**
T20_E15_SK6-7	**1.77E-01**†	1.70E-01
T30_E5_SK4-5	**5.31E-01**†	4.92E-01
T30_E10_SK4-5	5.06E-01†	**5.22E-01**
T30_E15_SK4-5	4.94E-01†	**5.07E-01**
T30_E5_SK6-7	**4.98E-01**†	4.07E-01
T30_E10_SK6-7	**5.55E-01**†	5.51E-01
T30_E15_SK6-7	**5.18E-01**†	4.96E-01
Real_1	**6.85E-01**†	6.65E-01
Real_2	**4.23E-01**†	3.80E-01
Real_3	**4.13E-01**†	3.19E-01

"†" indicates that both algorithms are significantly different at significance level of 0.05 by Wilcoxon's rank sum test.

D. EFFECT TO OTHER OBJECTIVES

This section explores that whether proposed approach (h_NEA) affects the consideration of other two objectives (robustness and stability). It means that while improving duration and cost objective, other objectives are not deteriorated. To answer this, h_NEA is analysed using four project objectives as duration, cost, robustness, and stability. The Table 7 gives objective vector for data instances chosen randomly. It is clear from the result that robustness and stability objectives have also been improved along with duration and cost. This is due to the reason that robustness objective also has some dependency on duration and cost objectives and stability assures that there is not too much variation in employees' assignment. The proposed algorithm optimizes all objectives and selects a best trade-off among solutions. This analysis may be helpful for a manager with deeper insights about various trade-offs among multiple objectives.

To evaluate the effectiveness of our approach with four objectives, HVR quality indicator is also applied. In Table 8, proposed approach outperforms existing algorithm for 67% data instances including three real instances by returning higher HVR values.

VI. THREATS TO VALIDITY

There are some limitations to this work. First, dynamic nature of the SPS problem needs to be studied in-depth. In this paper, only one type of dynamic event and uncertainty is considered; however, in real-world projects, many dynamic events may occur at the same time. Second, in this paper, only one critical path is considered whereas in real-world software projects can have multiple critical paths. Third, the scalability of proposed approach needs to be verified for large scale software projects. Moreover, resiliency of SPS solutions requires to be validated to check if solution generated by

TABLE 7. Objective values comparison for data instances. The selected solution is according to decision maker result [21].

	Duration	% Difference	Cost	% Difference	Robustness	% Difference	Stability	% Difference
				T10_E5_SK4-5				
h_NEA	**2.49E+01**	↓ 4%	**5.03E+05**	↓ 3%	**4.68E-01**	↓ 14%	**6.17E-02**	↓ 59%
SOA [7]	2.50E+01		5.18E+05		5.43E-01		1.49E-01	
				T20_E15_SK4-5				
h_NEA	**2.94E+01**	↓ 18%	**8.57E+05**	↓ 0.4 %	**4.20E-01**	↓ 14%	**3.70E+00**	↓ 36%
SOA [7]	3.60E+01		8.60E+05		4.92E-01		5.86E+00	
				T30_E5_SK6-7				
h_NEA	**4.02E+01**	↓ 45%	**1.40E+06**	↓ 0.7%	**4.61E-01**	↓ 31%	**6.38E-01**	↓ 13%
SOA [7]	7.33E+01		1.41E+06		6.70E-01		7.41E+00	
				Real_3				
h_NEA	**3.95E+00**	↓ 48%	**9.20E+04**	↓ 3.6%	**8.18E-02**	↓ 17%	**2.19E+00**	↓ 10.6%
SOA [7]	7.65E+00		9.54E+04		9.88E-02		2.45E+00	

TABLE 8. Mean value of HVR Indicator - 4 objectives, best mean is in boldface.

Instance	h-NEA	Existing Algorithm [7]
T10_E5_SK4-5	**6.85E-01** †	6.53E-01
T10_E10_SK4-5	6.14E-01†	**6.19E-01**
T10_E15_SK4-5	**6.43E-01** †	6.25E-01
T10_E5_SK6-7	**7.09E-01**†	7.07E-01
T10_E10_SK6-7	6.24E-01 †	**6.36E-01**
T10_E15_SK6-7	**6.76E-01** †	6.28E-01
T20_E5_SK4-5	**7.49E-01** †	7.39E-01
T20_E10_SK4-5	7.38E-01 †	**7.73E-01**
T20_E15_SK4-5	**7.07E-01** †	6.95E-01
T20_E5_SK6-7	6.56E-01 †	**6.66E-01**
T20_E10_SK6-7	**7.31E-01** †	7.21E-01
T20_E15_SK6-7	**7.19E-01** †	6.85E-01
T30_E5_SK4-5	7.17E-01 †	**7.47E-01**
T30_E10_SK4-5	**6.89E-01** †	6.75E-01
T30_E15_SK4-5	**7.33E-01** †	7.18E-01
T30_E5_SK6-7	**6.87E-01** †	6.74E-01
T30_E10_SK6-7	7.03E-01 †	**7.18E-01**
T30_E15_SK6-7	7.32E-01 †	**7.55E-01**
Real_1	**6.55E-01** †	6.35E-01
Real_2	**6.46E-01** †	5.47E-01
Real_3	**7.53E-01** †	6.99E-01

"†" indicates that both algorithms are significantly different at significance level of 0.05 by Wilcoxon's rank sum test.

proposed approach is identical to software project manager's choice. The comparison with Shen et al. [7] approach is based on the fact that this is the first paper which considered 3 dynamic events and 1 uncertainty. However, comparison with the latest approaches need to be conducted.

VII. CONCLUDING REMARKS

The growth in IT industry in a competitive market and volatile nature of software requirements needs new employees for new/existing projects. The software development methodologies have evolved where customer requirements are not initially locked but are flexible to take into account ongoing additional requirements or changes. This is an uncertain event which triggers project rescheduling by first optimally allocating tasks to existing resources, if not feasible, new employee additional event is triggered which is dynamic in nature. It is also worth noting that culture, context, dedication, and work environment plays very important role and are ensured through best HR practices; however, skill set mapping is one of the most critical success factor for the projects particularly when new employee is added against an employee turnover.

In this paper, a new heuristic approach for SPS is proposed to deal with "new employee addition" dynamic event and the project success is evaluated on technical skill set based limitations. The said approach is focused on the addition of new employee in the middle of the project where budget and time constraints are limitations. Moreover, one resource is engaged in multiple projects at the same time. This makes the addition of new employee with right skill set in multi project settings a complicated task. The proposed approach also employs domain knowledge for population initialization in order to generate high-quality solutions. The experimental results show that our approach is effective to generate software project plans with lower duration and cost compared to other existing approaches.

The principal target audience for this research are project managers in software development organizations who will get optimal project reschedule under 'employee addition' dynamic event particularly in complex, large scale and multi-project settings. The proposed approach is meant to be used in recommender system where project managers will be served with optimal project reschedule for further decision support. In future, more factors and other dynamic events faced in the SPS problem will be considered e.g., task's slack time and software requirements cancellation.

REFERENCES

[1] X.-N. Shen, L. L. Minku, N. Marturi, Y.-N. Guo, and Y. Han, "A *Q*-learning-based memetic algorithm for multi-objective dynamic software project scheduling," *Inf. Sci.*, vol. 428, pp. 1–29, Feb. 2018.
[2] E. Alba and J. Franciscochicano, "Software project management with GAs," *Inf. Sci.*, vol. 177, no. 11, pp. 2380–2401, Jun. 2007.
[3] H. Van Dyke Parunak, "Characterizing the manufacturing scheduling problem," *J. Manuf. Syst.*, vol. 10, no. 3, pp. 241–259, 1991.
[4] N. Nigar, "Model-based dynamic software project scheduling," in *Proc. 11th Joint Meeting Found. Softw. Eng.*, Aug. 2017, pp. 1042–1045.
[5] N. Nigar, "Multi-objective dynamic software project scheduling: An evolutionary approach for uncertain environments," Ph.D. dissertation, Dept. Comput. Sci., Univ. Birmingham, Birmingham, U.K., 2021.
[6] 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. 44677–44690, 2022.

[7] X. Shen, L. L. Minku, R. Bahsoon, and X. Yao, "Dynamic software project scheduling through a proactive-rescheduling method," *IEEE Trans. Softw. Eng.*, vol. 42, no. 7, pp. 658–686, Jul. 2016.

[8] X. Shen, Y. Guo, and A. Li, "Cooperative coevolution with an improved resource allocation for large-scale multi-objective software project scheduling," *Appl. Soft Comput.*, vol. 88, Mar. 2020, Art. no. 106059.

[9] A. L. Lederer and J. Prasad, "Causes of inaccurate software development cost estimates," *J. Syst. Softw.*, vol. 31, no. 2, pp. 125–134, Nov. 1995.

[10] J. Wiest and F. Levy, *A Management Guide to PERT/CPM: With GERT/PDM/DCPM and Other Networks*. Upper Saddle River, NJ, USA: Prentice-Hall, 1977.

[11] B. H. Rao, A. Gandhy, and R. R. Rathod, "A brief view of project scheduling techniques," *Int. J. Eng. Res. Technol.*, vol. 2, no. 12, pp. 1555–1559, 2013.

[12] W. Herrorlen and R. Leus, "Project scheduling under uncertainty: Survey and research potentials," *Eur. J. Oper. Res.*, vol. 165, no. 2, pp. 289–306, Sep. 2005.

[13] H. Ke and B. Liu, "Project scheduling problem with mixed uncertainty of randomness and fuzziness," *Eur. J. Oper. Res.*, vol. 183, no. 1, pp. 135–147, Nov. 2007.

[14] L. D. Long and A. Ohsato, "Fuzzy critical chain method for project scheduling under resource constraints and uncertainty," *Int. J. Project Manage.*, vol. 26, no. 6, pp. 688–698, Aug. 2008.

[15] A. V. Rezende, L. Silva, A. Britto, and R. Amaral, "Software project scheduling problem in the context of search-based software engineering: A systematic review," *J. Syst. Softw.*, vol. 155, pp. 43–56, Sep. 2019.

[16] M. Harman, S. A. Mansouri, and Y. Zhang, "Search based software engineering: A comprehensive analysis and review of trends techniques and applications," Dept. Comput. Sci., King's College London, London, U.K., Tech. Rep. TR-09-03, 2009, p. 23.

[17] J. Xiao, L. J. Osterweil, Q. Wang, and M. Li, "Dynamic resource scheduling in disruption-prone software development environments," in *Proc. Int. Conf. Fundam. Approaches Softw. Eng.* Berlin, Germany: Springer, 2010, pp. 107–122.

[18] J. Buchan, S. G. MacDonell, and J. Yang, "Effective team onboarding in agile software development: Techniques and goals," in *Proc. ACM/IEEE Int. Symp. Empirical Softw. Eng. Meas. (ESEM)*, Sep. 2019, pp. 1–11.

[19] K. Deb, M. Mohan, and S. Mishra, "Evaluating the ϵ-domination based multi-objective evolutionary algorithm for a quick computation of pareto-optimal solutions," *Evol. Comput.*, vol. 13, no. 4, pp. 501–525, 2005.

[20] I. Sommerville, *Software Engineering*, 9th ed. Reading, MA, USA: Addison-Wesley, 2010.

[21] X. N. Shen and X. Yao, "Mathematical modeling and multi-objective evolutionary algorithms applied to dynamic flexible job shop scheduling problems," *Inf. Sci.*, vol. 298, pp. 198–224, May 2015.

[22] T. L. Saaty and L. G. Vargas, "Comparison of eigenvalue, logarithmic least squares and least squares methods in estimating ratios," *Math. Model.*, vol. 5, no. 5, pp. 309–324, 1984.

[23] R. S. Pressman, *Software Engineering: A Practitioner's Approach*. New York, NY, USA: Macmillan, 2005.

[24] Z. Li and G. Ruhe, "Uncertainty handling in tabular-based requirements using rough sets," in *Proc. Int. Workshop Rough Sets, Fuzzy Sets, Data Mining, Granular-Soft Comput.* Heidelberg, Germany: Springer, 2005, pp. 678–687.

[25] N. Nan and D. E. Harter, "Impact of budget and schedule pressure on software development cycle time and effort," *IEEE Trans. Softw. Eng.*, vol. 35, no. 5, pp. 624–637, Sep. 2009.

[26] B. W. Boehm, "Software risk management: Principles and practices," *IEEE Softw.*, vol. 8, no. 1, pp. 32–41, Jan. 1991.

[27] M. Hapke, A. Jaszkiewicz, and R. Slowinski, "Fuzzy project scheduling system for software development," *Fuzzy Sets Syst.*, vol. 67, no. 1, pp. 101–117, Oct. 1994.

[28] S. Lazarova-Molnar and R. Mizouni, "A simulation-based approach to enhancing project schedules by the inclusion of remedial action scenarios," in *Proc. Winter Simulation Conf. (WSC)*, Dec. 2011, pp. 761–772.

[29] S. Gueorguiev, M. Harman, and G. Antoniol, "Software project planning for robustness and completion time in the presence of uncertainty using multi objective search based software engineering," in *Proc. 11th Annu. Conf. Genet. Evol. Comput.*, Jul. 2009, pp. 1673–1680.

[30] G. Antoniol, M. Di Penta, and M. Harman, "A robust search-based approach to project management in the presence of abandonment, rework, error and uncertainty," in *Proc. 10th Int. Symp. Softw. Metrics*, 2004, pp. 172–183.

[31] Y. Ge, "Software project rescheduling with genetic algorithms," in *Proc. Int. Conf. Artif. Intell. Comput. Intell.*, vol. 1, 2009, pp. 439–443.

[32] A. Wise, "Little-JiL 1.5 language report," Dept. Comput. Sci., Univ. Massachusetts, Amherst, MA, USA, Tech. Rep. UM-CS-2006-51, 2006.

[33] F. Chicano, A. Cervantes, F. Luna, and G. Recio, "A novel multiobjective formulation of the robust software project scheduling problem," in *Proc. Eur. Conf. Appl. Evol. Comput.* Heidelberg, Germany: Springer, 2012, pp. 497–507.

[34] S. Chand, H. K. Singh, and T. Ray, "Finding robust solutions for resource constrained project scheduling problems involving uncertainties," in *Proc. IEEE Congr. Evol. Comput. (CEC)*, Jul. 2016, pp. 225–232.

[35] J. Cheng, J. Ji, Y.-N. Guo, and J. Ji, "Dynamic multiobjective software project scheduling optimization method based on firework algorithm," *Math. Problems Eng.*, vol. 2019, pp. 1–13, Jul. 2019.

[36] Y. Ge and B. Xu, "Dynamic staffing and rescheduling in software project management: A hybrid approach," *PLoS ONE*, vol. 11, no. 6, Jun. 2016, Art. no. e0157104.

[37] M. Á. Vega-Velázquez, A. García-Nájera, and H. Cervantes, "A survey on the software project scheduling problem," *Int. J. Prod. Econ.*, vol. 202, pp. 145–161, Aug. 2018.

[38] F. P. Brooks, *The Mythical Man-Month* Reading, MA, USA: Addison-Wesley, 1995.

[39] C. K. Chang, H.-Y. Jiang, Y. Di, D. Zhu, and Y. Ge, "Time-line based model for software project scheduling with genetic algorithms," *Inf. Softw. Technol.*, vol. 50, no. 11, pp. 1142–1154, Oct. 2008.

[40] F. P. Brooks Jr., *The Mythical Man-Month: Essays on Software Engineering*. London, U.K.: Pearson, 1995.

[41] J. Žerovnik, "Heuristics for NP-hard optimization problems–simpler is better!" *Logistics Sustain. Transp.*, vol. 6, no. 1, pp. 1–10, Nov. 2015.

[42] S. Rahnamayan, H. R. Tizhoosh, and M. M. A. Salama, "A novel population initialization method for accelerating evolutionary algorithms," *Comput. Math. with Appl.*, vol. 53, no. 10, pp. 1605–1614, May 2007.

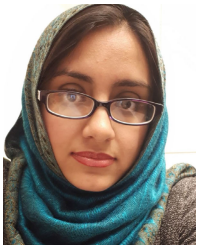
[43] D. Golenko-Ginzburg and A. Gonik, "Stochastic network project scheduling with non-consumable limited resources," *Int. J. Prod. Econ.*, vol. 48, no. 1, pp. 29–37, Jan. 1997.

[44] M. Li and X. Yao, "Quality evaluation of solution sets in multiobjective optimisation: A survey," *ACM Comput. Surv.*, vol. 52, no. 2, p. 26, 2019.

[45] D. Ouelhadj and S. Petrovic, "A survey of dynamic scheduling in manufacturing systems," *J. Scheduling*, vol. 12, no. 4, p. 417, 2009.

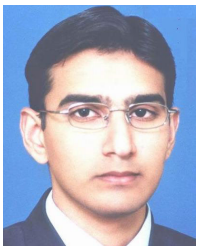
[46] F. Chicano, F. Luna, A. J. Nebro, and E. Alba, "Using multi-objective metaheuristics to solve the software project scheduling problem," in *Proc. 13th Annu. Conf. Genetic Evol. Comput.*, Jul. 2011, pp. 1915–1922.

[47] E. Zitzler, J. Knowles, and L. Thiele, "Quality assessment of Pareto set approximations," in *Multiobjective Optimization*. Heidelberg, Germany: Springer, 2008, pp. 373–404.



NATASHA NIGAR received the Ph.D. degree from the School of Computer Science, University of Birmingham, U.K., in 2021. From 2008 to 2009, she was a Lab Engineer with the National University of Computer and Emerging Sciences, Lahore, Pakistan. She was a Software Engineer with Palmchip Pvt. Ltd., Lahore, from 2009 to 2011; and a Senior Software Quality Assurance Engineer with Netsol Technologies, Lahore, from 2011 to 2013. She is currently an Assistant Professor with the Department of Computer Science, University of Engineering and Technology, Lahore. Her research interests include computational intelligence, optimization in dynamic and uncertain environments, and machine learning. She was awarded a Faculty Development Program Scholarship to pursue her Ph.D. studies.

MUHAMMAD KASHIF SHAHZAD received the bachelor's degree in engineering from University of Engineering and Technology (UET), Lahore, Pakistan, in 2000, and the master's and Ph.D. degrees in industrial systems engineering from the University of Grenoble, France, in 2008 and 2012, respectively. He is currently the Chief Technical Officer (CTO) with the Ministry of Energy, Power Division, Government of Pakistan, Power Information Technology Company (PITC). He has vast experience working in large-scale European research and development projects IMPROVE and INTEGRATE. He specializes in designing and delivering technology-driven smart grid solutions and is working with USAID in developing solutions to improve Pakistan's power sector in the deregulated market. He has more than 20 years of professional experience designing, business process re-engineering, and managing large-scale software development projects. He has 38 publications and two book chapters. His research interests include data model interoperability, advanced software engineering, technology, smart grid solutions, and engineering data management.



SHAHID ISLAM received the B.S. and M.S. degrees in computer science from the University of Engineering and Technology, Lahore, Pakistan, in 2003 and 2008, respectively. He is currently an Assistant Professor with the Rachna College, University of Engineering and Technology. His research interests include cloud computing, machine learning, semantic web, m-learning, and intelligent agent applications.

OLUKAYODE OKI (Member, IEEE) received the Ph.D. degree from the University of Zululand, South Africa, in 2019. He is currently a Lecturer with the Department of Information Technology, Walter Sisulu University, South Africa. He has received several grants both for research and development and to attend conferences. He has authored more than 40 articles. His research interests include biologically inspired computation, ICT4D, communication networks, the IoT, machine learning, data analytics, and climate-smart agriculture. He is a member of the IEEE South Africa Subsection. He was a recipient of the South Africa National Research Foundation (NRF) Rated Researcher Award and an Honorary Rosalind Member of the London Journal Press.

JOSE MANAPPATTUKUNNEL LUKOSE received the Ph.D. degree from Fort Hare University, South Africa. He is currently a Senior Lecturer and the Head of the Department of Information Technology, Walter Sisulu University, South Africa. His research interests include ICT in education and ICT for sustainable development.