IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received March 22, 2022, accepted April 12, 2022, date of publication April 22, 2022, date of current version May 2, 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3169596

Modeling Human Resource Experience Evolution for Multiobjective Project Scheduling in Large Scale Software Projects

NATASHA NIGAR[1], MUHAMMAD KASHIF SHAHZAD[2], SHAHID ISLAM[1], SATISH KUMAR[3], AND ABDUL JALEEL[1]

[1]Department of Computer Science (RCET), University of Engineering and Technology, Lahore, Lahore 39161, Pakistan
[2]Power Information Technology Company (PITC), Ministry of Energy, Power Division, Government of Pakistan, Lahore 39161, Pakistan
[3]School of Computing, University of Leeds, Leeds LS2 9JT, U.K.

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

ABSTRACT The software project scheduling (SPS) is a project-scheduling problem where limited human resources are assigned to the tasks in multi-team project settings. Besides other dynamic events, employees experience evolution has direct influence in completing large-scale projects within budget and time. In this paper, a new SPS model is developed as a dynamic multi-objective optimization problem, which incorporates employees experience evolution with their learning ability over time. The experimental results on 24 problem instances (including six real-world instances) show that the developed SPS model reduces project duration by 40% while being within budget. The results provide evidence that consideration of experience evolution while tasks reallocation under dynamic events significantly optimizes project schedules. Moreover, the developed SPS model is evaluated with six state-of-the-art algorithms as bi-criterion evolution (BCE), NSGA-II, NSGA-III, Two_Arch2, OMOPSO, speed-constrained multi-objective particle swarm optimization (SMPSO) where BCE demonstrated distinct superiority for 63% data instances.

INDEX TERMS Software project scheduling, experience, metaheuristics, multi-objective optimization.

I. INTRODUCTION

In the modern world, employee experience has been an increasingly important factor in human resource (HR) and business. A survey reports that 79% of business and HR leaders believe that relevant experience is essential and has become imperative for the project success [1]. Moreover, an analysis of 250 global organizations reveals the fact that companies who have highest score for employees experience generated two times higher average revenues, four times higher average profits and 40% low turnover [2]. The employees experience involves knowledge, skills, practice and situation familiarity. It promotes agility, service and rapid response to project activities, which are the most important drivers for success.

In the last two decades, software industry has transformed into a highly competitive market. In this competitive market, success requires efficient and effective project plan to reduce the time and cost of quality software development that meet or exceeds customer's expectation [3]. Therefore, who does what during the whole software project development life cycle, also known as SPS problem [4], becomes critical. The SPS problem consists of allocation of employees to the tasks over a project time-line. The classical methods like critical path method (CPM) [5], program evaluation and review technique (PERT) [5], and the resource-constrained project scheduling problem (RCPSP) model [6] have been intensively applied for solving the SPS problem but they are becoming obsolete due to increasing technology dependence in the software projects.

As software companies strive to get their jobs done faster with cost-cutting approaches in order to succeed [7], therefore, it has become important to consider about important factors while modelling the SPS problem [8]. In this regard, prior researches have found the existence of learning curves (static experience) in manufacturing and service industries and provide evidence for positive impact on project scheduling [9]. However, none of these studies considers and investigates

The associate editor coordinating the review of this manuscript and approving it for publication was Shih-Wei Lin.

the evolution of employees' experiences particularly for the SPS problem with inherently increasing complexity due to technology dependence. The software project development is a people intensive activity [10] that requires varying degree of skills and experience on newly emerging technologies. Therefore, we incorporate this important human attribute into the SPS model. This study also demonstrates that how significantly our developed SPS model reduces project duration and cost as compared to the existing models [11].

The SPS problem is multi-objective in nature [4] and search-based optimization approaches have become a promising way in dealing with it. The search-based software engineering (SBSE) is the practice of reformulating typical software engineering issues as search problems, and applying meta-heuristic methods such as simulated annealing (SA) and evolutionary algorithms (EAs) to find near-optimal solutions. In literature, the SPS problem is formulated as search-based optimization problem and is solved by employing EAs to help project managers for near-optimal schedules.

In this paper, employee's experience attribute is modelled as a novel formulation and is investigated with six state-of-the-art algorithms to solve the developed SPS model. In brief, this work makes the following three contributions:

i) **SPS Model:** A new model for the SPS problem is developed which provides the understanding of why the software project schedule generated using this model reduces total project duration and cost. Further, formulation of the cost objective is based on real-world scenario [12].
ii) **State-of-the-art algorithms study:** Six state-of-the-art algorithms, namely, BCE [13], NSGA-II [14], NSGA-III [15], Two_Arch2 [16], OMOPSO [17], and SMPSO [18] are compared to identify the best fit algorithm in generating an efficient project plan for the developed model.
iii) **Data Set:** Apart from 18 benchmark data instances, developed model is tested with 6 real-world data instances. However, this is the first study which has used real-world data with maximum 91 tasks to solve the SPS problem. Previous studies address the SPS problem with 15 tasks for real-world data.

This paper is organized into six sections. The section II overviews the background and related work whereas section III presents the developed model. Experimental design and results for the SPS problem are presented in section IV. The section V describes threats to validity of our study. Finally, the section VI draws conclusion with possible future directions.

II. BACKGROUND AND RELATED WORKS

A. SOFTWARE PROJECT SCHEDULING

The software project management (SPM) is a sub-discipline of project management. It involves scheduling, planning, monitoring, and controlling of software projects to achieve specific objectives, while satisfying a variety of constraints. This work mainly deals with the SPS problem under SPM. The main focus of solving this problem is to create a schedule for a project with minimal duration and cost by appropriate allocation of employees to the tasks. Generally, we define a project as the planned set of interrelated tasks that are executed within specified time and cost to achieve one or more specific objectives [19]. A software project is a complete procedure of software development within a specified period of time to achieve the desired software. A project can be characterized as: i) well-defined tasks, ii) unique and distinct goal, iii) no routine activity or day-to-day operations, iv) have start and end time, v) ends upon achieving project goal; hence, considered as a temporary phase in organization's life time, and vi) adequate resources for a project in terms of time, manpower, finance, and material. Based on above definitions, employees and tasks may be distinguished as two major elements of the SPS problem as shown in Fig. 1.

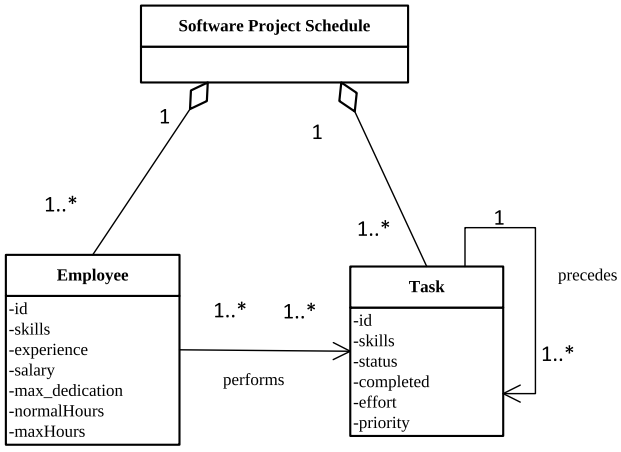


FIGURE 1. SPS main entities.

To elaborate this, let us suppose a software project comprising of five tasks $T = \{t_1, t_2, t_3, t_4, t_5\}$ and seven employees $E = \{e_1, e_2, e_3, \ldots.., e_7\}$ with employee's skills set $SK = \{s_1, s_2, s_3, s_4, s_5\}$. The skill of an employee is denoted as $e^{skill} \subseteq SK$, the monthly salary as e^{salary} and maximum dedication to the project with e^{maxded}. Each skill is associated with a proficiency value in the interval (0,5] [11]. The proficiency value of '0' implies that employee is not proficient for that skill, and a value of '5' indicates that employee is fully master in a skill. Salary is expressed in fictitious currency units, while the maximum dedication is defined as the ratio of amount of hours dedicated to the project divided by full length of the employee's working day. Here, salary and maximum dedication are real numbers. A simplified scenario example is given in Fig. 2. Employee e_1 who earns £2,000 each month, is a programming (s_1) and UML (s_2) expert. His/her colleague employee besides UML (s_2) expertise, is also a software tester (s_3) and data modeller (s_4). These employees (e_1, e_2) and employee e_5 with leadership skills (s_5), can spend all their day on the project as they possess maximum dedication equal to one; however, this does not necessarily mean that

FIGURE 2. Example of software project employees with different skill set and dedication level.

they spend all of their time. On the other hand, employee e_3 is a programmer (s_1) and can only dedicate half of his/her working day developing the software application. This may be due to the several reasons, perhaps the employee contract is part-time; or he/she has to look after some administrative tasks as part of their dedication. Employee e_4 can work overtime ($e^{maxHours}$) with his/her maximum dedication greater than one ($e^{maxded} = 1.2$). This implies that he/she can work on the project up to 20% more than in a normal working day (e^{nhours}).

Moreover, each employee has an experience attribute (e^{exp}) in the SPS problem which evolves with time. An illustration of evolving employee experience is given in Fig. 3.

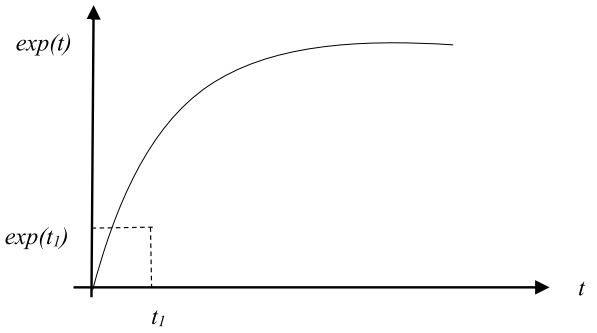


FIGURE 3. An illustration of evolving employee experience over Time '*t*'.

While modelling the tasks, each task is associated with some skills denoted as t^{skills} and an effort t^{effort} expressed in person-month (PM). The tasks are executed based on 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 between 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. $t^{completed}$ is a binary variable indicating that whether a task has been finished at time t. $t^{completed} = 1$ shows that task is still in execution whereas $t^{completed} = 0$ represents that task has been completed. Another binary variable t^{status} represents task's availability that whether a task is active at time t or it has been cancelled/suspended, e.g., due to some missing skill. If $t^{status} = 1$, it shows that task is available whereas $t^{status} = 0$ represents that task has been cancelled/suspended. t^{prio} represents task's priority level i.e. very high, high, or medium. Another variable is e_{ij}^{prof}, it indicates employee e_i proficiency level for performing a task t_j.

The Fig. 4 shows all software project tasks considered in the example. Task t_1 requires UML expertise (skill s_2) to prepare the design model document in order to be used later by the employees in completing the subsequent tasks, and it needs 3 person-months to be completed. In the same figure, it can also be seen that the last task t_7 (User manual creation) cannot be instantiated until all previous tasks are completed.

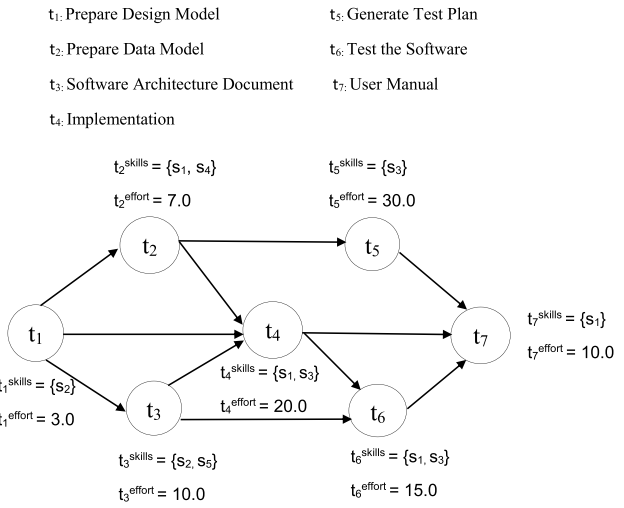


FIGURE 4. Example of task precedence graph.

1) SOLUTION TO THE SPS PROBLEM

The solution to the SPS problem is represented by a dedication matrix $X = (x_{ij})$ of size $|E|$ x $|T|$ where $x_{ij} \geq 0$. The element $|E|$ represents number of employees, $|T|$ denotes number of tasks, and x_{ij} represents the degree of dedication of an employee e_i to task t_j. For example, an employee e_i is

	t_1	t_2	t_3	t_4	t_5	t_6	t_7
e_1	1.00	0.50	0.50	0.00	0.00	1.00	0.87
e_2	0.50	1.00	1.00	0.33	0.50	0.50	0.00
e_3	0.00	0.70	0.00	0.34	0.00	0.33	0.33
e_4	0.67	0.00	0.87	0.00	0.00	0.00	0.00
e_5	0.00	0.00	1.00	0.00	0.90	0.00	0.00

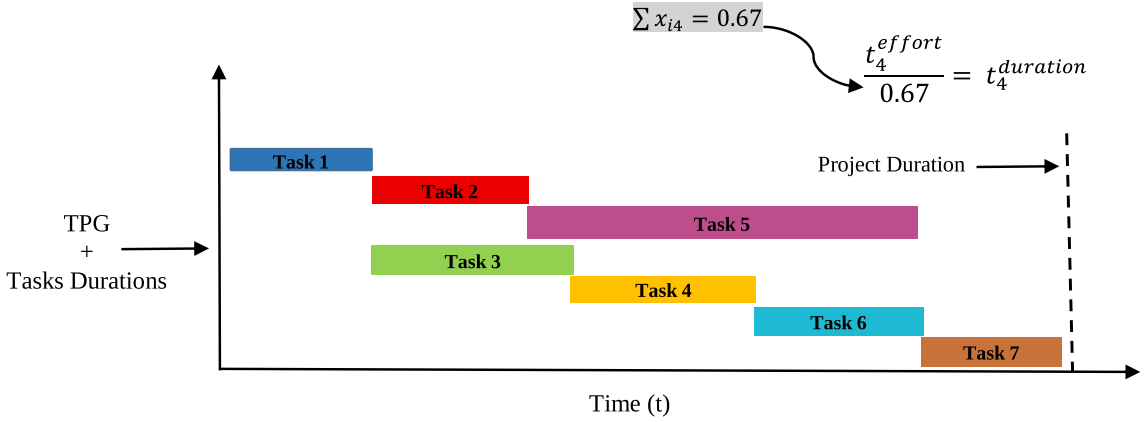


FIGURE 5. Tentative solution to the SPS problem.

allocated on a task t_j with dedication degree of '1' means that he/she spends his/her all normal working hours of a day to that task. The term $x_{ij} = 0$ means that employee will not perform that task. This information is used to calculate the duration and starting and finishing time of each task. According to the TPG and dedication matrix, Gantt chart of the project can be also be drawn as shown in Fig. 5.

A task's duration is calculated according to the formulation in [4]. The project cost is calculated using project duration, dedication matrix and employees' salary.

B. MULTI-OBJECTIVE EVOLUTIONARY ALGORITHMS

In SBSE [20], search-based techniques are applied to solve the real-world large scale problems; for example, software effort estimation [21]–[23], software defect prediction [24]–[26], and software testing [27]–[30]. The SPS is one of those problems for which EAs have been widely applied [10]. These EAs are robust and solve real-world large scale problems efficiently by providing a set of non-dominated solutions known as Pareto optimal set [31], [32]. The representation of the Pareto optimal set in the objective space is known as the Pareto front [31].

In recent years, the optimization problems involving two or more conflicting objectives have received researchers' increasing attention. These are referred as multi-objective optimization problems (MOPs) whereas search-based optimization algorithms for solving MOPs are known as multi-objective algorithms [33]. The main goal of MOPs is obtaining the Pareto front. There can be a large (possibly infinite) number of non-dominated solutions in Pareto front. In practice, a Pareto front is an approximation of a finite number of non-dominated solutions. In this context, this set of solutions must fulfil two features: convergence and diversity [31]. **Convergence** means that solutions should be as close as possible to the exact Pareto front. It ensures that we are dealing with good-enough solutions [31]. **Diversity** is good exploration of the search space that solutions are uniformly spread and no regions are left unexplored [31]. For example, for the SPS problem (where the software project manager desires a project schedule with both low project cost and smaller duration), there does not exist one single schedule that achieves both objectives.

The studies in [34]–[37] analyse the performance of classical algorithms on the same model as in [4]. Therefore, we investigate six state-of-the-art multi-objective algorithms (NSGA-II [14], NSGA-III [15], Two_Arch2 [16], BCE [13], OMOSPO [17], SMPSO [18]) for our developed model which includes an important human factor. These six multi-objective algorithms are capable of finding a set of trade-off solutions in one single run and solve MOPs in an optimal way. Therefore, they are also known as evolutionary multi-objective (EMO) algorithms. The purpose of comparing these algorithms for the SPS problem is to provide the project manager the best algorithm which gives near-optimal project schedule. The reason for selecting these specific six algorithms is that they are simple to implement for the SPS problem and they are shown to be effective on other problems as mentioned earlier, for example, software effort estimation [21]–[23], software defect prediction [24]–[26], and software testing [27]–[30]. Some of them (NSGA-II, NSGA-III,

OMOSPO, SMPSO) also have existing implementation in the jMetal framework [38]. jMetal is an object-oriented Java-based framework aimed at the development, experimentation, and study of metaheuristics for solving MOPs. The NSGA-II was introduced by Deb *et al.* [14]. A fast non-dominated sorting solution and the preservation of the solution's diversity are its two principal parts. NSGA-III [15] is an improved version of NSGA-II and is more suitable to deal with many objectives (more than three) problems. Two_Arch2 [16] is a many objective algorithm comprising of two archives focusing on convergence and diversity separately. OMOPSO [17] is a multi-objective particle swarm optimization (MOPSO) [39] variant which includes an external archive and mutation operators to expedite the convergence of swarm. SMPSO [18] is also a MOPSO which uses a velocity constriction mechanism with the aim of enhancing the search capability.

The BCE [13] is a multi-objective algorithm. Its framework is composed of two parts: pareto criterion (PC) evolution and non-pareto criterion (NPC) evolution to deal with the MOPs. These two parts work collaboratively, attempting to use their strengths to facilitate each other's evolution. The PC evolution effectively maintains the individual diversity, hence, approaching the optimal front. It can also preserve any non-dominated individual located in a sparse region.

In the BCE framework, the two populations communicate constantly and fully share and compare their information in a generational manner. NPC evolution drives the PC evolution forward while PC evolution compensates for the possible diversity loss of the NPC evolution. An individual newly produced in one population is tested and applied in the other population. The current status of the NPC evolution is reflected by information comparison of the two populations, thus, making the search more focused on some undeveloped but promising regions.

C. STATE-OF-THE-ART MODEL

This section provides brief discussion on state-of-the-art model [11] for the SPS problem. The reason for selecting this model is that it is a dynamic version of the model presented in [40], and the dedication of each employee to each task is determined dynamically. The mathematical model for two objectives is explained below:

$$\min \mathbf{F}(t) = [f_1, f_2] \tag{1}$$

where f_1, f_2 represents duration and cost objectives receptively.

$$f_1(t) = duration = max(T_j^{end}) - min(T_j^{start}) \tag{2}$$

The duration measure $f_1(t)$ (2) is maximum elapsed time required for completion of each available task. The terms T_j^{end} and T_j^{start} represent the starting and ending time of each task. Moreover, duration for the whole project is the maximum finishing time of the last task.

$$f_2(t) = cost = \sum_{e_i \in E\_ava\_set} e\_cost_i \tag{3}$$

The term $f_2(t)$ (3) represents project cost, which can be defined as the sum of all expenses payable to available employees against their dedications to project tasks. The set of available employees is denoted by E_ava_set. Let T_active_set denotes the set of active tasks which are being developed at time moment t', where t' represents any month during which the project is being developed. A task is called active if it does not have any preceding unfinished task according to the TPG at time moment t'. Therefore, the expenses paid to the employee e_i at t' month are calculated as follows:

$$e\_cost_i = \begin{cases} if \ \sum_{j \in T\_active\_set} x_{ij} \leq 1.0 \\ e_i^{norm\_salary} \cdot t' \cdot \sum\limits_{j \in T\_active\_set} x_{ij} \end{cases} \tag{4}$$

$$e\_cost_i = \begin{cases} if \ \sum_{j \in T\_active\_set} x_{ij} \geq 1.0 \\ e_i^{norm\_salary} \cdot t' \cdot 1 + e_i^{overwork\_salary} \cdot t' \\ \quad \cdot \left(\sum\limits_{j \in T\_active\_set} x_{ij} - 1 \right) \end{cases} \tag{5}$$

where $e_i^{norm\_salary}$ is employee's normal salary, and overwork salary is represented by $e_i^{overwork\_salary}$, x_{ij} is dedication of an employee e_i to task t_j. The term x_{ij} greater than 1 indicates that employee overworks for the project and overtime salary is also added to total salary.

D. RELATED WORK: EXISTING MODELS IN LITERATURE

Alba and Chicano [4] proposed a task-based model to tackle the SPS problem using genetic algorithm (GA) and solved many project scenarios. In their work, employee overwork was considered as a constraint. Minku *et al.* [7] developed an improved evolutionary algorithm using the same formulation as presented in [4] and overcome the limitation that employees cannot exceed their maximum dedication. Chang *et al.* [41] proposed software project management net (SPMnet) model solves the SPS problem by providing good-enough solutions. To induce more reality in task-based model [4], Chen and Zhang [10] regarded employees leaves/return as events and proposed an event based scheduler model using ant colony optimization (ACO) algorithm. Hegazy *et al.* [42] proposed a model which optimizes cost and dynamic project control. Their model allocates optional construction methods for each task and incorporates an integrated formulation for scheduling, estimating, resource management, and cash-flow analysis. Human factors are not considered much in their work. Their key focus is to give the best combination of construction methods. Delgoshaei *et al.* [43] decision making model mainly deals with project's objectives in the presence of uncertain events for RCPSP. Tirkolaee *et al.* [44] bi-objective model for RCPSP considers the net present value (NPV) and duration as optimizing objectives without consideration of any human factor. The studies in [45], [46] take into account

the minimization of cost and duration objectives respectively, for RCPSP. Besides these, some studies [47], [48] provide a survey of research on the SPS problem.

The above literature highlights the fact that most of the studies have used the same model as proposed in [4] without consideration of employee's experience factor. Employee's experience significantly improves organization's productivity and is imperative for project success [2], therefore, it motivates our research to consider this factor into the SPS model.

III. THE DEVELOPED MODEL

This section presents a more practical version of mathematical model for the SPS problem, which considers human factors as increasing employee experience together with project duration and cost. Different from previous work [49], firstly, it incorporates employees' experiences with employees' learning ability over time. Secondly, the cost objective formulation is according to real-world [12]. The employees and tasks are two major elements of the SPS problem instance as shown in Fig. 1. These elements and the model is described in detail as follows.

A. EMPLOYEE'S ATTRIBUTES

Software development is a people-intensive activity. In real-world, a software house keeps record of each employee like employee's wages, skills and working constraints etc. Suppose 'n' employees are working for a software project as E $\{e_1, e_2, e_3, \ldots e_n\}$. For employee e_i, the attributes in Table 1 are considered.

TABLE 1. Employee's attributes.

Symbol	Description
e^{id}	Employee's specific id.
e^{skills} [11]	The set of employee's skills in which he/she is proficient.
e^{exp}	Employee's experience falls between [0,1], '1' refers to experienced employee, whereas '0' refers to the fresh employee.
e^{basic_salary}	Employee's basic salary per month.
$e^{perhour\_salary}$	Employee's per hour salary.
$e^{overwork\_salary}$ [11]	Employee's overtime work salary.
e^{nhours}	Employee's normal working hours per month.
$e^{available}$ [11]	Employee's availability during project.
e^{maxded} [11]	The maximum dedication of employee e_i for a software project represents the percentage of full-time job that he/she is able to dedicate. '$e^{max\_ded} = 1$' indicates that he/she spends all normal working hours on the project, whereas $e^{max\_ded} = 1.2$' means that he/she can work 20% more than normal working hours.

B. TASK'S ATTRIBUTES

A software project comprises of multiple tasks 'm' as T $\{t_1, t_2, \ldots, t_m\}$. For example, tasks could be interface design, business requirements, integrate system modules, candidate release, and production plan sign off etc. These tasks are executed according to a TPG, which specifies which tasks should finish before starting a new preceding task. For task t_j, the following attributes are considered in Table 2.

TABLE 2. Task's attributes.

Symbol	Description
T^{id}	Task's specific id.
T^{skills} [11]	The set of skills required to accomplish a task.
T^{status}	Task status, '1' refers to active task whereas '0' refers to cancelled/suspended task.
T^{prio}	Task's priority for execution as mentioned in task list {Very High, High, Medium}.
T^{effort} [11]	Effort (unit: person-months) required to complete the task.
$T^{completed}$	A binary variable indicating whether a task has been completed or not. '1' means task is unfinished and '$T_j^{completed} = 0$' shows task has been completed.
TPG [11]	TPG is an acyclic directed graph G(T,A) to represent the dependency between tasks. In TPG, the set of nodes represents the set of tasks T. The precedence relations among the tasks is denoted by set of arcs A.

C. OBJECTIVE FUNCTIONS

To solve the SPS problem and provide software manager near-optimal project schedule, two project objectives namely, duration and cost are optimized.

$$\min \mathbf{F}(t) = [f_1, f_2] \tag{6}$$

where f_1, f_2 represents duration and cost objectives receptively.

Duration - It is the total time required to complete the project and is taken as the maximum finishing time of the last task.

$$f_1(t) = duration = max(T_j^{end}) - min(T_j^{start}) \tag{7}$$

The task duration is calculated in following way:

$$T_j^{dur} = \frac{T_j^{effort}}{\sum_{i \in E'} x_{ij} * (1 + k\varepsilon_t)} \tag{8}$$

ε_t is assumed to be increased with the time t as follows:

$$\varepsilon_t = \tanh(\varepsilon_o + a\triangle t) \tag{9}$$

The terms T_j^{end} and T_j^{start} (7) denote the starting and ending time of each task. The term T^{effort} (8) is the total estimated effort for a task, X is employee to task dedication matrix, 'E'' is number of employees, ε_t is employee experience at time 't' and 'k' is parameter for experience dependency of tasks i.e. how much time of task depends on experience. 'k' range is between [0,1]. To represent employee's increasing experience over time, a sigmoid (hyperbolic tangent) function (9) according to [8], where ε_o is employee initial experience, 'a' is growth rate of experience and we define $a = \frac{\mu}{\phi}$ where μ is employee learning factor between range [0.5,1.5]. ϕ is time unit parameter, it is defined as the number of time units in a month. Thus, if the time unit is a day, then $\phi = 30$. $\bigtriangleup t$ is time difference $t - t_o$ where 't' is highest value among task's predecessor durations and t_o is initial time.

The employee's experience factor distinguishes this mathematical model from existing models. It is important to note that the employee's experience evolution is not based on the assumption that tasks are consistent that will improve employee's skills. In this model, we consider employee's

diverse experience. It is independent regardless that he/she works on same tasks. Since software engineer's experience is difficult to measure [50], researchers often rely on two proxies for this abstract concept [51] i.e. length of experience and self-assessed expertise [52].

Cost - Project cost is the sum of all the expenses paid to all employees in the software development process until project completion.

$$f_2(t) = \sum_{i=1}^{E'} cost_i \tag{10}$$

In the real-world, there are two types of employees in a software house, permanent and temporary ones. An employee salary is divided into three parts: basic, per-hour normal working, and overwork salary [10]. Permanent employees have stable basic salaries every month without taking into account their dedication to the project. This basic salary is added to the per-hour salary which is based on the employee's normal working hours and his/her dedication to the project. The basic salaries are paid only to permanent employees while temporary employees have higher per-hour working salaries.

Suppose an employee e_i dedicates *nhours* to the project at t' month with x_{ij} dedication. The employee's basic salary $e_i^{basic_salary}$ is added to the per-hour salary $e_i^{perhour_salary}$ which is paid according to *nhours* and x_{ij} dedication. If employee's dedication (x_{ij}) is greater than 1, it indicates that employee overworks for the project and overtime salary $e_i^{overwork_salary}$ is also added to the employee's total salary. The expenses paid to employee e_i at t' month are calculated as follows:

$$\begin{aligned} &if \ \sum_{j\in T} x_{ij} \leq 1.0 \\ cost_i &= \left(\sum_{j\in T} x_{ij} * nhours * e_i^{perhour\_salary}\right) \cdot t' \\ &+ e_i^{basic\_salary} \cdot t' \end{aligned} \tag{11}$$

$$\begin{aligned} &if \ \sum_{j\in T} x_{ij} \geq 1.0 \\ cost_i &= \left(\sum_{j\in T} x_{ij} * nhours * e_i^{perhour\_salary}\right) \cdot t' \\ &+ e_i^{basic\_salary} \\ &\cdot t' + \left(\left(\sum_{j\in T} x_{ij} * nhours\right)\right. \\ &\left. - nhours * e_i^{overwork\_salary}\right) \cdot t' \end{aligned} \tag{12}$$

In contrast to other objective functions in the SPS literature, the duration objective is based on an important human factor 'employee experience' whereas the cost objective is designed by considering normal working hours, basic salary, per-hour salary, and overwork salary as in real-world situation [12].

D. CONSTRAINTS

In this work, we consider following two hard constraints.

i) All Tasks Allocated Constraint. All tasks should be allocated to the available employees. The term T_j_Dedication (13) represents employees total dedication for that task.

$$\forall e_i \in E_a va_s et(t), \sum T_j Dedication \neq 0 \tag{13}$$

ii) Task Skills Constraint. All the allocated employees to the task must fulfil the skills required by that task.

$$T_j\_\text{Skills} \subseteq \cup e_i \ \{Skills \mid x_{ij} > 0\} \tag{14}$$

Regarding constraint handling, a solution is penalized by multiplying objective values with a high number if any constraint is violated. In our case, we are multiplying with 1,000.

IV. EXPERIMENTAL ANALYSIS

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

- RQ1: What benefit can the developed model bring to the SPS problem?
- RQ2: How do state-of-the-art algorithms perform on developed model? Do they perform similarly, or are specific algorithms particularly suitable for this model?

A. EXPERIMENTAL SETUP

1) SPS INSTANCES

In this section, the experiments are conducted on broad range of benchmark and real-world data instances.

a: REAL-WORLD

The six real-world data instances are used in our experiments. The 3 data instances among these real instances have been derived from business software construction projects for a departmental store [53]. These 3 small real instances have 10 employees with 15 tasks maximum, and do not address inclusion of employee experience factor. For these instances, we have randomly generated values between [0,1] for employee experience attribute. To measure the scalability of our model, we also use 3 real instances obtained from a software company. These data instances comprise of 10,8,5 employees and 42,43, and 91 tasks respectively. For these data instances, employee's experience is measured by a software project manager based on two proxies as mentioned in [52] i.e. length of experience and self-assessed expertise.

b: BENCHMARK

Benchmark instances are derived from Alba and Chicano's benchmark [4]. These specific 18 data instances include variants of three important parameters (number of employees, number of tasks and number of employee skills) for the SPS problem as in real-world scenario. To induce more reality, these instances differ from static instances [4] in the following keys aspects: task maximum headcount, task effort

uncertainties, part-time jobs, overworking of employees, and employees experience. The data of 18 instances, derived from [4], are gathered from different software projects.

Each instance has different number of employees and tasks which can be (5, 10, or 15), and (10, 20, or 30) respectively. Each employee can possess different skills which ranges from 4 to 5, or from 6 to 7. The total number of different skills *SK* is 10, and a task is associated with five skills randomly selected from them. 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); another 20 percent of employees do overtime work, 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. Each skill is associated with a proficiency score. The proficiency score for a skill is sampled uniformly from (0, 5] at random. If an employee does not have any specific skill, then proficiency score for that skill is set to 0. Further, an employee's normal monthly salary is sampled from a normal distribution with the mean of 10,000 and standard deviation of 1,000 following the practice in [4]. The employee overtime salary is the normal monthly salary multiplied by 3. For benchmark instances, employee's experience is generated randomly between [0,1].

The data instances are denoted as 'T20_E10_SK4-5', whereas 'T20' means total number of tasks, 'E10' represents total number of employees, and 'SK4-5' means number of skills for an employee. As an another example, T10_E5_SK6-7 represents that the project has 10 tasks with 5 employees, each employee with 6 or 7 skills. The 6 real instances are named as 'Real_n_Tx_Ey' where n ϵ {1,2,...6} and x, y represents number of tasks and number of employees respectively.

2) PARAMETER SETTINGS

A set of parameter values for a rational comparison among algorithms are presented in Table 3.

TABLE 3. Parametrization (L = Individual Length).

NSGA-II [14], NSGA-III [15], Two_Arch2 [16], BCE [13]	
Population size	100 individuals
Selection of Parents	binary tournament + binary tournament
Recombination	simulated binary, p_c = 0.9
Mutation	polynomial, p_m = 1.0 / L
OMOPSO [17]	
Swarm size	100 particles
Mutation	uniform + non-uniform, p_m = 1.0 / L
Leader size	100
SMPSO [18]	
Swarm size	100 particles
Mutation	polynomial, p_m = 1.0 / L
Leader size	100

3) EVALUATION OF SOLUTION QUALITY

The hypervolume (HV) metric [54] is used to compare i) the performance of developed model against state-of-the-art model [11] and ii) the state-of-the-art algorithms

TABLE 4. Performance comparison of developed vs existing model. The selected solution against each instance is returned by decision maker result [61].

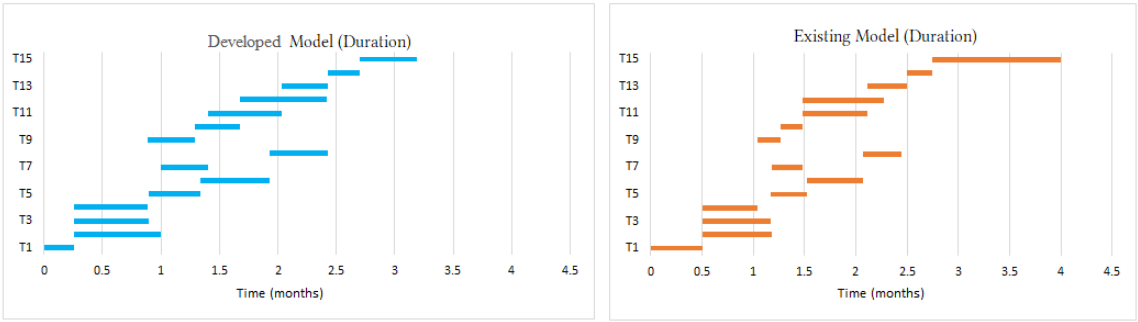
	Developed Model		Existing Model	
	Duration	Cost	Duration	Cost
Real_1_T15_E10	**3.63E+00**	**8.09E+04**	4.48E+00	1.58E+05
Real_2_T15_E10	**2.66E+00**	**2.52E+04**	8.28E+00	4.66E+04
Real_3_T12_E10	**1.91E+01**	**1.97E+04**	2.90E+01	8.35E+04
Real_4_T42_E10	**1.42E+01**	**2.84E+06**	2.52E+01	7.50E+06
Real_5_T43_E8	**4.26E+01**	**2.49E+06**	5.27E+01	6.00E+06
Real_6_T91_E5	**1.71E+01**	**4.84E+06**	2.28E+01	1.22E+07

TABLE 5. HVR values obtained by NSGA-II on the developed and existing model, best mean is highlighted in boldface.

	Developed Model	Existing Model
T10_E5_SK4-5	**1.90E+00**	0.00E+00
T10_E10_SK4-5	6.74E-01	**9.73E-01**
T10_E15_SK4-5	7.23E-01	**9.04E-01**
T10_E5_SK6-7	**8.48E-01**	0.00E+00
T10_E10_SK6-7	7.90E-01	**9.59E-01**
T10_E15_SK6-7	7.68E-01	**9.48E-01**
T20_E5_SK4-5	**5.03E-01**	0.00E+00
T20_E10_SK4-5	**6.30E-01**	5.19E-01
T20_E15_SK4-5	6.12E-01	**6.48E-01**
T20_E5_SK6-7	**9.73E-01**	1.12E-01
T20_E10_SK6-7	5.24E-01	**5.92E-01**
T20_E15_SK6-7	**7.93E-01**	3.23E-01
T30_E5_SK4-5	**1.21E-01**	0.00E+00
T30_E10_SK4-5	**5.53E-01**	4.75E-01
T30_E15_SK4-5	**5.90E-01**	4.83E-01
T30_E5_SK6-7	2.15E-01	**7.00E-01**
T30_E10_SK6-7	**5.29E-01**	4.27E-01
T30_E15_SK6-7	**7.01E-01**	3.21E-01
Real_1_T15_E10	**9.48E-01**	0.00E+00
Real_2_T15_E10	**9.59E-01**	5.75E-03
Real_3_T12_E10	**8.21E-01**	0.00E+00
Real_4_T42_E10	**9.31E-01**	0.00E+00
Real_5_T43_E8	**8.55E-01**	0.00E+00
Real_6_T91_E5	**5.70E-01**	0.00E+00

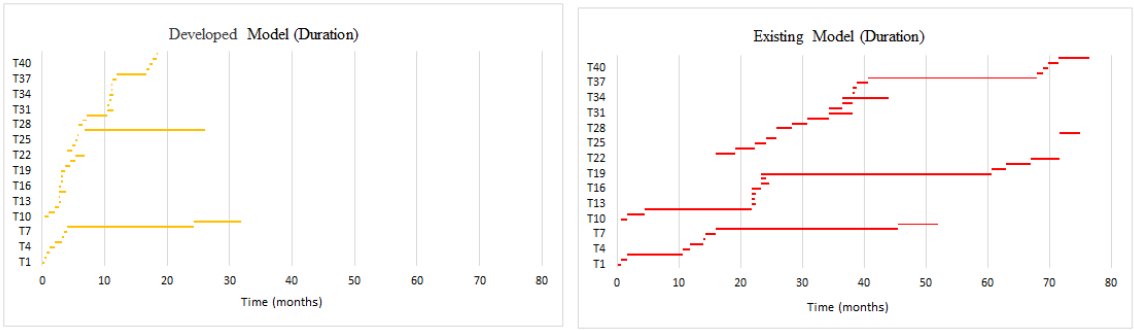
performance for the new model. HV is considered as more suitable for real-world problems whose Pareto front are unknown [54]. Here, we use a normalised HV value (also called hypervolume ratio (HVR)). With this normalisation, the range of obtained results reside in [0, 1], where 1 represents the optimal value. Usually, there are two crucial issues in the calculation of HV i.e. search space scaling [55] and the reference point choice [56], [57]. Since the objectives in the SPS problem reside in different ranges of values, therefore, the objective values of the final solution set is standardized with respect to the range of the obtained Pareto front. We set the reference point 1.1 times the upper bound of the Pareto front to balance between convergence and diversity of the obtained solution set, according to [58].

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. Furthermore, we set population size to 100 according to practice in [59], [60].



(a) Gantt chart for Real_1

(b) Gantt chart for Real_2



(c) Gantt chart for Real_4

FIGURE 6. **Gantt chart for real instances: Developed vs existing model. The selected solution is returned by Decision Maker result [61].**

B. COMPARISON OF DEVELOPED SPS MODEL WITH STATE-OF-THE-ART MODEL

In this section, the performance of developed model against state-of-the-art model [11] using a baseline algorithm (NSGA-II [14]) is presented. The main purpose of this comparison is to validate that our developed model impacts and reduces project duration and cost. To address this, we compare results of developed model by those achieved by existing approach in Table 4. The results show that our developed model significantly reduces project duration and cost for 6 real-world data instances. Since EAs provide a set of non-dominated solutions to solve any NP-hard problem, therefore, a non-dominated solution set is obtained for each problem instance. It also implies that the software project manager can choose from a set of solutions with a good balance between project cost and duration. We choose and present one solution using an automatic decision making method [11].

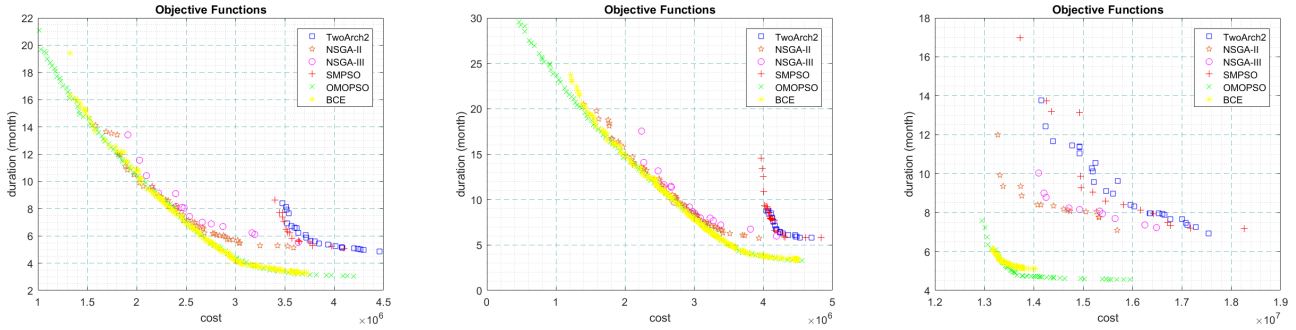
To compare the performance using quality indicator, Table 5 shows the HVR values for both models for benchmark and real-world data instances. These values are average of 30 independent runs. Algorithms returning higher HVR values perform better [54]. For most of the benchmark data instances, our model returns higher HVR values. Moreover, existing model returns zero HVR value for five out of six real data instances, it means that no solutions are found which dominate the reference point. That's why it is clear from results that our novel formulation is more effective as compared to the existing one.

To further elaborate the effectiveness of our developed model, we draw a Gantt chart for real-world instances obtained by the developed and existing models in Fig. 6. For

TABLE 6. Mean and standard deviation values of HVR indicator, best mean is highlighted in boldface.

Instances	BCE	NSGA-II	NSGA-III	Two_Arch2	OMOPSO	SMPSO
T10_E5_SK4-5	0.00E+00(0.00E+00)	**8.87E-01**(4.78E-01) †	2.69E-04(1.45E-3) †	0.00E+00(0.00E+00)	0.00E+00(0.00E+00)	0.00E+00(0.00E+00)
T10_E10_SK4-5	**8.81E-01**(1.32E-02)†	8.24E-01(2.12E-02) †	7.30E-01(2.57E-02)†	5.51E-01(7.20E-03)†	8.37E-01(3.61E-02) †	5.62E-01(5.13E-03) †
T10_E15_SK4-5	**9.16E-01**(4.19E-02) †	8.03E-01(2.32E-02) †	7.05E-01(3.48E-02) †	2.21E-01(7.32E-03) †	7.91E-01(4.83E-02) †	2.30E-01(6.62E-03) †
T10_E5_SK6-7	0.00E+00(0.00E+00)	**2.91E-01**(4.57E-01) †	2.30E-01(3.94E-01)†	0.00E+00(0.00E+00)	0.00E+00(0.00E+00)	0.00E+00(0.00E+00)
T10_E10_SK6-7	**8.91E-01**(2.05E-02) †	8.39E-01(3.50E-02) †	7.26E-01(3.58E-02) †	5.13E-01(9.72E-03) †	8.27E-01(3.86E-02) †	5.25E-01(5.93E-03) †
T10_E15_SK6-7	**9.48E-01**(2.32E-02)†	8.37E-01(2.25E-02) †	7.34E-01(3.00E-02) †	1.99E-01(8.11E-03) †	8.96E-01(3.65E-02) †	2.16E-01(8.64E-03) †
T20_E5_SK4-5	0.00E+00(0.00E+00)	**1.15E-01**(7.90E-01) †	0.00E+00(0.00E+00)	0.00E+00(0.00E+00)	0.00E+00(0.00E+00)	0.00E+00(0.00E+00)
T20_E10_SK4-5	**8.40E-01**(4.63E-02) †	4.59E-01(3.48E-02) †	2.98E-01(5.34E-02) †	2.41E-01(2.22E-02)†	8.14E-01(4.46E-02)†	2.41E-01(3.34E-02) †
T20_E15_SK4-5	6.44E-01(8.84E-02) †	5.55E-01(4.20E-02) †	3.08E-01(6.24E-02) †	1.01E-01(1.78E-02) †	**6.87E-01**(6.99E-02) †	8.33E-02(3.59E-02) †
T20_E5_SK6-7	**8.14E-01**(4.78E-02) †	1.47E-01(2.94E-02) †	6.22E-02(4.50E-02) †	6.02E-02(2.15E-02)†	7.05E-01(6.35E-02) †	5.60E-02(2.62E-02) †
T20_E10_SK6-7	**7.96E-01**(7.49E-02)†	4.69E-01(4.31E-02) †	2.41E-01(8.77E-02) †	2.26E-01(2.01E-02) †	7.84E-01(5.20E-02) †	2.11E-01(3.78E-02) †
T20_E15_SK6-7	7.35E-01(5.09E-02) †	7.36E-01(2.63E-02) †	5.99E-01(5.09E-02) †	4.46E-01(1.96E-02) †	**7.75E-01**(3.10E-02) †	4.57E-01(2.34E-02) †
T30_E5_SK4-5	9.86E-02(5.34E-02) †	9.94E-02(5.26E-01) †	9.77E-05(5.26E-04) †	**5.37E-01**(4.62E-02) †	9.78E-02(3.49E-02) †	5.09E-01(5.61E-02) †
T30_E10_SK4-5	2.47E-01(5.40E-02) †	1.27E-01(1.56E-02) †	7.42E-02(2.50E-02) †	**8.98E-01**(2.54E-02) †	2.73E-01(3.29E-02)†	8.56E-01(2.64E-02) †
T30_E15_SK4-5	**8.60E-01**(4.72E-02) †	4.75E-01(2.40E-02) †	3.67E-01(4.15E-02) †	2.69E-01(2.09E-02) †	8.16E-01(3.77E-02) †	2.63E-01(2.83E-02) †
T30_E5_SK6-7	**5.05E-01**(2.38E-01) †	0.00E+00(0.00E+00)	0.00E+00(0.00E+00)	0.00E+00(0.00E+00)	8.44E-02(1.11E-01) †	0.00E+00(0.00E+00)
T30_E10_SK6-7	**4.27E-01**(2.29E-01)†	0.00E+00(0.00E+00)	0.00E+00(0.00E+00)	0.00E+00(0.00E+00)	2.62E-01(.42E-01) †	0.00E+00(0.00E+00)
T30_E15_SK6-7	6.83E-01(1.24E-01)†	4.22E-01(3.76E-02) †	2.33E-01(5.92E-02) †	5.54E-02(2.28E-02) †	**6.85E-01**(7.59E-02)†	4.43E-02(3.07E-02) †
Real_1	**9.81E-01**(4.09E-03) †	9.09E-01(1.41E-02) †	8.58E-01(1.43E-02)†	0.00E+00(0.00E+00)	9.20E-01(2.17E-02) †	0.00E+00(0.00E+00)
Real_2	**9.67E-01**(3.13E-03)†	9.43E-01(6.91E-03) †	9.02E-01(6.56E-03) †	4.51E-01(1.54E-02) †	9.58E-01(4.71E-03) †	5.08E-01(1.59E-02) †
Real_3	7.44E-01(4.56E-03) †	**7.89E-01**(7.59E-02) †	7.23E-01(1.38E-02) †	0.00E+00(0.00E+00)	7.28E-01(7.19E-03) †	0.00E+00(0.00E+00)
Real_4	**9.73E-01**(9.74E-03) †	8.76E-01(1.29E-02) †	8.11E-01(1.88E-02) †	0.00E+00(0.00E+00)	9.26E-01(1.66E-02) †	0.00E+00(0.00E+00)
Real_5	**9.83E-01**(1.33E-03) †	8.65E-01(6.09E-03) †	8.48E-01(7.40E-03) †	1.04E-01(1.66E-02) †	9.73E-01(4.35E-03) †	1.69E-01(1.10E-02) †
Real_6	**8.73E-01**(3.71E-02) †	2.12E-01(3.70E-02) †	1.50E-01(5.24E-02) †	1.56E-01(2.44E-02) †	8.16E-01(3.22E-02) †	2.33E-01(2.27E-02) †

"†" indicates that each algorithm result is significantly different from each other at 0.05 significance level by the Wilcoxon's rank sum test.



(a) 'T10_E15_SK4-5' instance (b) 'T10_E15_SK6-7' instance (c) 'T30_E15_SK4-5' instance

FIGURE 7. Results comparison between state-of-the-art algorithms. The final solutions of State-of-the-art Algorithms are shown in the objective space f_1 and f_2.

developed model, project duration is reduced by 40% while being within budget. An experienced employee is able to finish a task earlier. It is obvious from the results that inclusion of an important human factor into SPS model returns a shorter project duration as desired.

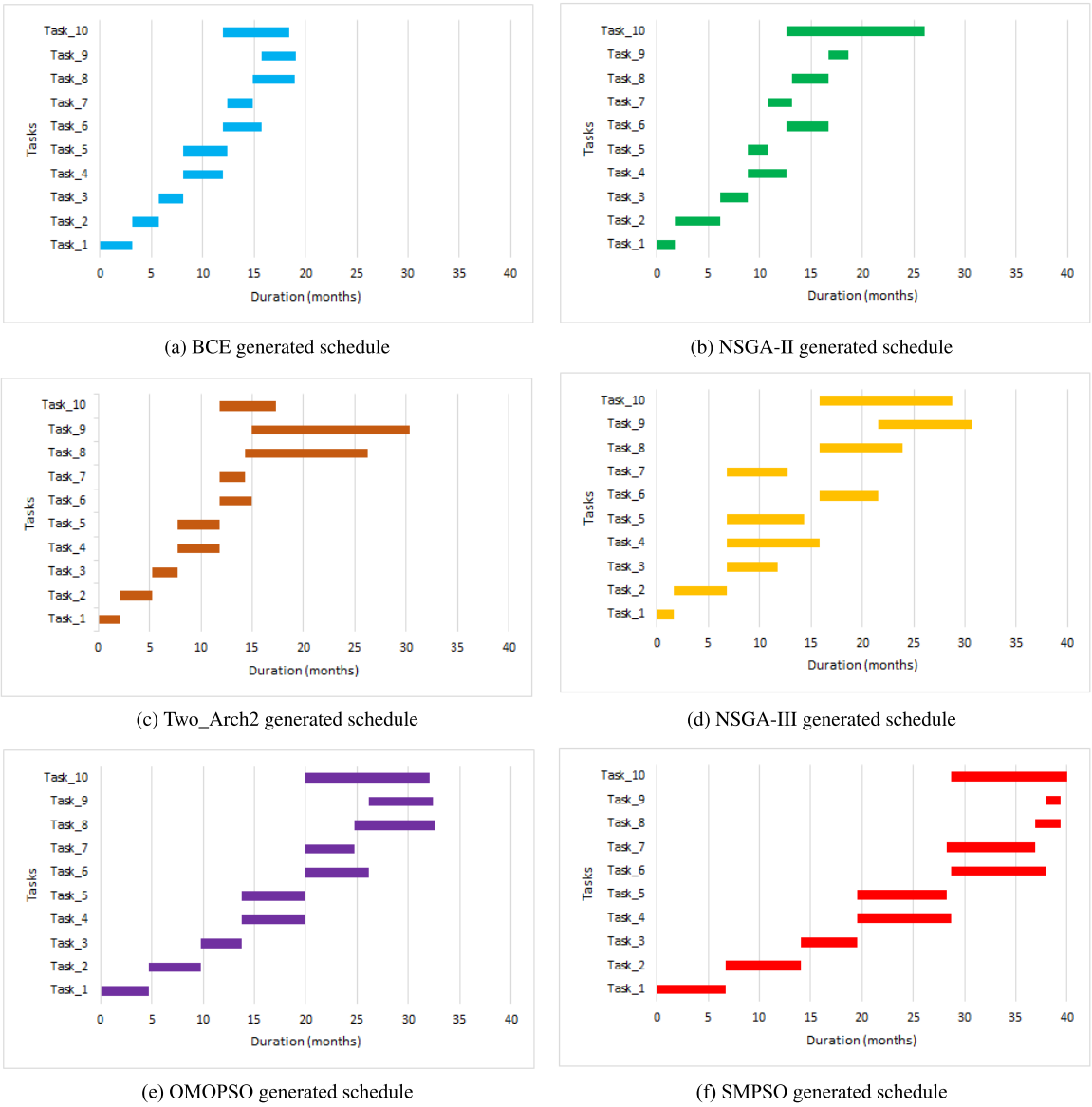
C. COMPARISON OF STATE-OF-THE-ART ALGORITHMS FOR DEVELOPED MODEL

In this section, we analyse which state-of-the-art algorithm is best fit for our developed model while considering employee's experience factor. As mentioned earlier, studies in [34]–[37] analyse the performance of classical metaheuristic on the same model as proposed in [4]. The comparison of six targeted state-of-the-art algorithms based on HVR values is to identify which algorithm provides the software project managers near-optimal project schedules.

HVR results in Table 6 are the mean and standard deviation (SD) values. For each problem instance, the best mean values among the algorithms are shown with **bold** font. Moreover, concerning the statistically sound conclusions, the Wilcoxon's rank sum test is applied [62] (using the significance level of 0.05) to determine that if the results obtained by the state-of-the-art algorithms are significantly different from each other.

It can be observed from Table 6 that for 63% data instances, BCE is giving better results based on HVR values. After BCE, NSGA-II, OMOPSO and Two_Arch2 are performing better for rest of the instances. The results also show that the state-of-the-art algorithms are significantly different from each other for most of the test data instances.

The final solutions of a single run of all state-of-the-art algorithms are plotted in Fig. 7 regarding two-dimensional objective space f_1 and f_2, for benchmark data instances 'T10_E15_SK4-5', 'T10_E15_SK6-7', and 'T30_E15_SK4-5' as chosen randomly. As shown, BCE returns a well-converged and well-distributed set of solutions whereas the five peer algorithms show a poor performance in comparison to it. After BCE, OMOPSO is performing better by



(a) BCE generated schedule

(b) NSGA-II generated schedule

(c) Two_Arch2 generated schedule

(d) NSGA-III generated schedule

(e) OMOPSO generated schedule

(f) SMPSO generated schedule

FIGURE 8. **Comparison of software project schedules based on EMO algorithms for 'T10_E5_SK4-5' data instance.**

providing the minimized values of project duration and cost to the software project manager. SMPSO and Two_Arch2 are performing least well and do not provide good schedules. For the developed model, BCE also shows, with statistical significance better performance against the other five state-of-the-art algorithms.

In Fig. 8, schedules have been drawn from the results generated by each algorithm. We have selected the first run's results for data instance 'T10_E5_SK4-5'. The software project schedule generated from the BCE algorithm result while considering evolving employee's experience, has the shortest project duration as desired. Another interesting observation is that after BCE, NSGA-II performs better than all other algorithms when drawing schedule. Two_Arch2 algorithm's results dominates NSGA-III in terms of project schedule. NSGA-III performs better than OMOPSO but worse than others. SMPSO is performing least and returns the highest project duration.

To sum up, the distinct superiority of BCE is due to overcoming the following limitations of its peers: i) NSGA-III has been designed for more than three objectives and may not give very good results for two objective problems [15] as in our case, ii) OMOPSO loses the extrema of the Pareto front due to the use of ε-dominance [17] which results in losing optimal solutions, iii) SMPSO is among the slowest techniques in yielding a HVR greater than zero, that represents the bad exploration of search space resulting in limited solutions [18], iv) Two_Arch2 does not keep extreme points of the Pareto fronts which indicates loss of potential solutions [16], and v) NSGA-II lacks the uniform diversity among obtained

non-dominated solutions while neglecting some potential areas in the search space [14]. We also observed that for data instances comprised of 42,43, and 91 tasks respectively, BCE performs better while other algorithms give trivial solutions.

V. THREATS TO VALIDITY

This study is to help software project managers in generating good software project plan. However, there are some limitations to this work. First, the duration objective has not considered the group learning behaviour. It implies that if multiple employees are working on a task, they can learn from each other's expertise which will also impact and reduce the project duration. In this study, we select and implement one single solution (project schedule) from a set of solutions based on decision maker choice. The weight of each objective function in decision maker may be subjective. The resiliency of SPS solutions is another open issue. It means to check if solution generated by our approach is identical as assigned by project manager (employees to tasks).

VI. CONCLUSION AND FUTURE DEVELOPMENTS

This paper investigates the SPS problem for the software industry where significant cost and time can be saved by appropriately assigning employees to tasks. The employees' experience along with their learning ability over time, has a direct impact on reducing project duration and cost. Based on this fact, this paper presents a new model for the SPS problem with the consideration of employee experience evolution. The developed model is evaluated on 24 problem instances, including six real-world instances, where results show that developed SPS model reduces 40% project duration while being within budget. This provides an evidence that taking into account experience evolution while tasks reallocation under dynamic events optimizes project schedules significantly. Moreover, the developed SPS model is evaluated with six state-of-the-art algorithms as bi-criterion evolution (BCE), NSGA-II, NSGA-III, Two_Arch2, OMOPSO, speed-constrained multi-objective particle swarm optimization (SMPSO). The BCE generates large hypervolume i.e. objective space in comparison to other algorithms which inherently provides trade off based on employees experience evolution for resources reallocation during project rescheduling while being within budget. The BCE has demonstrated distinct superiority for 63% data instances.

It is critical for the current competitive software industry to adopt inclusion of employees experience evolution to provide their business partners a smart competitive edge. Based on the results as discussed in the above paragraph, the developed SPS model including employees experience evolution will facilitate project managers and practitioners to improve project schedule while being within budget under dynamic and uncertain events where rescheduling either increases cost or affects the project schedule. The results also provide an evidence for researchers that hypervolume based decisions on algorithmic choice may be leveraged by other factors if modelled in SPS problem to provide optimal solution.

As future work, the propsoed SPS model will be generalized with more objectives. For example, project quality is one of the important project objectives [63] besides project cost and duration. In addition, during the development of software projects, many unforeseen events may trigger, e.g. a new employee may join the company in the middle of project, requirements cancellation, and employee resign etc. To adapt project schedules to these dynamic changes is important.

REFERENCES

[1] F. Bersin and M. Mazor. (2017). *The Employee Experience: Culture, Engagement, and Beyond*, [Online]. Available: https://www2.deloitte.com/us/en/insights/focus/human-capital-trends/2017/improving-the-employee-experience-culture-engagement.html

[2] J. Morgan and M. Goldsmith. *The Employee Experience Advantage: How to Win the War for Talent by Giving Employees the Workspaces They Want, the Tools They Need, and a Culture They Can Celebrate*. Hoboken, NJ, USA: Wiley, 2017. [Online]. Available: https://books.google.co.uk/books?id=qBUwvQEACAAJ

[3] 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.

[4] E. Alba and J. F. Chicano, "Software project management with GAs," *Inf. Sci.*, vol. 177, no. 11, pp. 2380–2401, Jun. 2007.

[5] 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. [Online]. Available: https://books.google.co.uk/books?id=Hn1HAAAAMAAJ

[6] F. Habibi, F. Barzinpour, and S. J. Sadjadi, "Resource-constrained project scheduling problem: Review of past and recent developments," *J. Project Manage.*, vol. 3, no. 2, pp. 55–88, 2018.

[7] L. L. Minku, D. Sudholt, and X. Yao, "Improved evolutionary algorithm design for the project scheduling problem based on runtime analysis," *IEEE Trans. Softw. Eng.*, vol. 40, no. 1, pp. 83–102, Jan. 2014.

[8] 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.

[9] M.-C. Wu and S.-H. Sun, "A project scheduling and staff assignment model considering learning effect," *Int. J. Adv. Manuf. Technol.*, vol. 28, nos. 11–12, pp. 1190–1195, May 2006.

[10] W.-N. Chen and J. Zhang, "Ant colony optimization for software project scheduling and staffing with an event-based scheduler," *IEEE Trans. Softw. Eng.*, vol. 39, no. 1, pp. 1–17, Jan. 2013.

[11] 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.

[12] Source. (2019). *Know Your Worth*. [Online]. Available: https://www.glassdoor.com/Salaries/know-your-worth.htm

[13] M. Li, S. Yang, and X. Liu, "Pareto or non-Pareto: Bi-criterion evolution in multiobjective optimization," *IEEE Trans. Evol. Comput.*, vol. 20, no. 5, pp. 645–665, Oct. 2016.

[14] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *IEEE Trans. Evol. Comput.*, vol. 6, no. 2, pp. 182–197, Apr. 2000.

[15] K. Deb and H. Jain, "An evolutionary many-objective optimization algorithm using reference-point-based nondominated sorting approach. Part I: Solving problems with box constraints," *IEEE Trans. Evol. Comput.*, vol. 18, no. 4, pp. 577–601, Aug. 2014.

[16] H. Wang, L. Jiao, and X. Yao, "Two_Arch2: An improved two-archive algorithm for many-objective optimization," *IEEE Trans. Evol. Comput.*, vol. 19, no. 4, pp. 524–541, Aug. 2014.

[17] M. R. Sierra and C. A. C. Coello, "Improving PSO-based multi-objective optimization using crowding, mutation and ϵ-dominance," in *Proc. Int. Conf. Evol. Multi-Criterion Optim.* Springer, 2005, pp. 505–519.

[18] A. J. Nebro, J. J. Durillo, J. Garcia-Nieto, C. A. C. Coello, F. Luna, and E. Alba, "SMPSO: A new PSO-based Metaheuristic for multi-objective optimization," in *Proc. IEEE Symp. Comput. Intell. Milti-Criteria Decision-Making (MCDM)*, Mar. 2009, pp. 66–73.

[19] R. Ireland, B. West, N. Smith, and D. I. Shepherd, *Project Management for IT-Related Projects*. London, U.K.: The Chartered Institute, 2012.

[20] 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.

[21] L. L. Minku and X. Yao, "An analysis of multi-objective evolutionary algorithms for training ensemble models based on different performance measures in software effort estimation," in *Proc. 9th Int. Conf. Predictive Models Softw. Eng.*, Oct. 2013, pp. 1–10.

[22] S.-J. Huang and N.-H. Chiu, "Optimization of analogy weights by genetic algorithm for software effort estimation," *Inf. Softw. Technol.*, vol. 48, no. 11, pp. 1034–1045, Nov. 2006.

[23] F. S. Gharehchopogh, I. Maleki, and S. R. Khaze, "A novel particle swarm optimization approach for software effort estimation," *Int. J. Academic Res.*, vol. 6, no. 2, pp. 69–76, Mar. 2014.

[24] R. S. Wahono and N. Suryana, "Combining particle swarm optimization based feature selection and bagging technique for software defect prediction," *Int. J. Softw. Eng. Appl.*, vol. 7, no. 5, pp. 153–166, Sep. 2013.

[25] R. S. Wahono, N. Suryana, and S. Ahmad, "Metaheuristic optimization based feature selection for software defect prediction," *J. Softw.*, vol. 9, no. 5, pp. 1324–1333, May 2014.

[26] H. Can, X. Jianchun, Z. Ruide, L. Juelong, Y. Qiliang, and X. Liqiang, "A new model for software defect prediction using particle swarm optimization and support vector machine," in *Proc. 25th Chin. Control Decis. Conf. (CCDC)*, May 2013, pp. 4106–4110.

[27] P. R. Srivastava and K. Baby, "Automated software testing using metahurestic technique based on an ant colony optimization," in *Proc. Int. Symp. Electron. Syst. Design*, Dec. 2010, pp. 235–240.

[28] H. Li and C. P. Lam, "An ant colony optimization approach to test sequence generation for state based software testing," in *Proc. 5th Int. Conf. Quality Softw. (QSIC)*, Sep. 2005, pp. 255–262.

[29] G. Zhang, Z. Su, M. Li, F. Yue, J. Jiang, and X. Yao, "Constraint handling in NSGA-II for solving optimal testing resource allocation problems," *IEEE Trans. Rel.*, vol. 66, no. 4, pp. 1193–1212, Dec. 2017.

[30] W. Zheng, R. M. Hierons, M. Li, X. H. Liu, and V. Vinciotti, "Multi-objective optimisation for regression testing," *Inf. Sci.*, vol. 334, pp. 1–16, Mar. 2016.

[31] K. Deb, *Multi-Objective Optimization Using Evolutionary Algorithms*, vol. 16. Hoboken, NJ, USA: Wiley, 2001.

[32] S. Kumar, R. Bahsoon, T. Chen, K. Li, and R. Buyya, "Multi-tenant cloud service composition using evolutionary optimization," in *Proc. IEEE 24th Int. Conf. Parallel Distrib. Syst. (ICPADS)*, Dec. 2018, pp. 972–979.

[33] C. A C. Coello, G. B. Lamont, D. A. Van Veldhuizen, *Evolutionary Algorithms for Solving Multi-Objective Problems*, vol. 5. New York, NY, USA: Springer, 2007.

[34] 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. Genet. Evol. Comput.*, 2011, pp. 1915–1922.

[35] B. Crawford, R. Soto, F. Johnson, S. Misra, and F. Paredes, "The use of metaheuristics to software project scheduling problem," in *Proc. Int. Conf. Comput. Sci. Appl.* Cham, Switzerland: Springer, 2014, pp. 215–226.

[36] F. Luna, F. Chicano, and E. Alba, "Robust solutions for the software project scheduling problem: A preliminary analysis," *Int. J. Metaheuristics*, vol. 2, no. 1, pp. 56–79, 2012.

[37] F. Luna, D. L. González-Álvarez, F. Chicano, and M. A. Vega-Rodríguez, "The software project scheduling problem: A scalability analysis of multi-objective metaheuristics," *Appl. Soft Comput.*, vol. 15, pp. 136–148, Feb. 2014.

[38] J. J. Durillo and A. J. Nebro, "jMetal: A Java framework for multi-objective optimization," *Adv. Eng. Softw.*, vol. 42, no. 10, pp. 760–771, 2011.

[39] C. A. C. Coello and M. S. Lechuga, "MOPSO: A proposal for multiple objective particle swarm optimization," in *Proc. Congr. Evol. Comput. (CEC)*, vol. 2, May 2002, pp. 1051–1056.

[40] C. K. Chang, M. J. Christensen, and T. Zhang, "Genetic algorithms for project management," *Ann. Softw. Eng.*, vol. 11, no. 1, pp. 107–139, Nov. 2001, doi: 10.1023/A:1012543203763.

[41] C. K. Chang, C. Chao, T. T. Nguyen, and M. Christensen, "Software project management net: A new methodology on software management," in *Proc. 22nd Annu. Int. Comput. Softw. Appl. Conf. (COMPSAC)*, 1998, pp. 534–539.

[42] T. Hegazy and K. Petzold, "Genetic optimization for dynamic project control," *J. Construct. Eng. Manage.*, vol. 129, no. 4, pp. 396–404, Aug. 2003.

[43] A. Delgoshaei, A. Aram, V. Mantegh, S. Hanjani, A. H. Nasiri, and F. Shirmohamadi, "A multi-objectives weighting genetic algorithm for scheduling resource-constraint project problem in the presence of resource uncertainty," *Int. J. Supply Oper. Manage.*, vol. 6, no. 3, pp. 213–230, 2019.

[44] E. B. Tirkolaee, A. Goli, M. Hematian, A. K. Sangaiah, and T. Han, "Multi-objective multi-mode resource constrained project scheduling problem using Pareto-based algorithms," *Computing*, vol. 101, no. 6, pp. 547–570, Jun. 2019.

[45] A. Delgoshaei, T. Rabczuk, A. Ali, and M. K. A. Ariffin, "An applicable method for modifying over-allocated multi-mode resource constraint schedules in the presence of preemptive resources," *Ann. Oper. Res.*, vol. 259, nos. 1–2, pp. 85–117, Dec. 2017.

[46] M. Ning, Z. He, T. Jia, and N. Wang, "Metaheuristics for multi-mode cash flow balanced project scheduling with stochastic duration of activities," *Autom. Construct.*, vol. 81, pp. 224–233, Sep. 2017.

[47] 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.

[48] 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.

[49] N. Nigar, "Model-based dynamic software project scheduling," in *Proc. 11th Joint Meeting Found. Softw. Eng.*, Aug. 2017, pp. 1042–1045.

[50] A. Mockus and J. D. Herbsleb, "Expertise browser: A quantitative approach to identifying expertise," in *Proc. 24th Int. Conf. Softw. Eng.*, 2002, pp. 503–512.

[51] J. Siegmund, C. Kästner, J. Liebig, S. Apel, and S. Hanenberg, "Measuring and modeling programming experience," *Empirical Softw. Eng.*, vol. 19, no. 5, pp. 1299–1334, Oct. 2014.

[52] S. Baltes and S. Diehl, "Towards a theory of software development expertise," in *Proc. 26th ACM Joint Meeting Eur. Softw. Eng. Conf. Symp. Found. Softw. Eng.*, Oct. 2018, pp. 187–200.

[53] 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.

[54] 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.

[55] T. Friedrich, C. Horoba, and F. Neumann, "Multiplicative approximations and the hypervolume indicator," in *Proc. 11th Annu. Conf. Genetic Evol. Comput. (GECCO)*, 2009, pp. 571–578.

[56] A. Auger, J. Bader, D. Brockhoff, and E. Zitzler, "Theory of the hypervolume indicator: Optimal μ-distributions and the choice of the reference point," in *Proc. 10th ACM SIGEVO Workshop Found. Genetic Algorithms*, 2009, pp. 87–102.

[57] T. Friedrich, F. Neumann, and C. Thyssen, "Multiplicative approximations, optimal hypervolume distributions, and the choice of the reference point," *Evol. Comput.*, vol. 23, no. 1, pp. 131–159, Mar. 2015.

[58] H. Ishibuchi, Y. Hitotsuyanagi, N. Tsukamoto, and Y. Nojima, "Many-objective test problems to visually examine the behavior of multiobjective evolution in a decision space," in *Proc. Int. Conf. Parallel Problem Solving Nature (PPSN)*, 2010, pp. 91–100.

[59] M. Garza-Fabre, G. Toscano-Pulido, C. A. C. Coello, and E. Rodriguez-Tello, "Effective ranking + speciation = many-objective optimization," in *Proc. IEEE Congr. Evol. Comput. (CEC)*, Jun. 2011, pp. 2115–2122.

[60] T. Wagner, N. Beume, and B. Naujoks, "Pareto-, aggregation-, and indicator-based methods in many-objective optimization," in *Evolutionary Multi-Criterion Optimization*. Berlin, Germany: Springer, 2007, pp. 742–756.

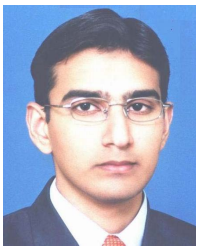
[61] X.-N. Shen and X. Yao, "Mathematical modeling and multi-objective evolutionary algorithms applied to dynamic flexible job shop scheduling problems," *Inf. Sci.*, vol. 298, no. 219, pp. 198–224, Mar. 2015.

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

[63] M. Tavana, A.-R. Abtahi, and K. Khalili-Damghani, "A new multi-objective multi-mode model for solving preemptive time–cost–quality trade-off project scheduling problems," *Expert Syst. Appl.*, vol. 41, no. 4, pp. 1830–1846, Mar. 2014.

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

MUHAMMAD KASHIF SHAHZAD received the bachelor's degree in engineering from 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 working as the Chief Technical Officer (CTO) at Power Information Technology Company (PITC), Ministry of Energy, Power Division, Government of Pakistan. His research interests include data models interoperability, advanced software engineering, technology, smart grid solutions, and engineering data management. He has vast experience of 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 deregulated market. He has 38 publications and two book chapters. Moreover, he has more than 20 years of professional experience designing, business process re-engineering, and managing large scale software development projects.



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 working as an Assistant Professor with the Rachna College, University of Engineering and Technology, Lahore. His research interests include cloud computing, machine learning, semantic web, m-learning, and intelligent agent applications.

SATISH KUMAR received the Ph.D. degree from the School of Computer Science, University of Birmingham, U.K., in 2021. He is currently a Research Fellow with the School of Computing, University of Leeds, U.K. His research is mainly on distributed systems, where he focuses on engineering service-based systems, resource management in cloud-edge computing, Internet of Things systems, and self-adaptive software systems. His research work has received the Best Student Paper Award at ACM/IEEE SEAMS Conference. He has published several research articles in the top ACM/IEEE conferences, such as ICWS, SEAMS, and ICPADS.

ABDUL JALEEL received the B.S., M.S., and Ph.D. degrees in computer science and engineering from the University of Engineering and Technology, Lahore, Pakistan, in 2006, 2010, and 2019, respectively. He is currently working as an Associate Professor with the Rachna College, University of Engineering and Technology, Lahore. His research interests include the development of self-managing software applications, health-IoT, autonomic computing, and software quality measurement metrics.