IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 17 December 2023, accepted 2 January 2024, date of publication 10 January 2024, date of current version 29 January 2024.

Digital Object Identifier 10.1109/ACCESS.2024.3352438

RESEARCH ARTICLE

A Practical Approach for Resource-Constrained Project Scheduling

KONSTANTINOS MANOUSAKIS[1,2], (Senior Member, IEEE), GIANNIS SAVVA[1,2], (Member, IEEE), NICOS PAPADOURI[1,2], MICHALIS MAVROVOUNIOTIS[3], ATHANASIOS CHRISTOFIDES[4], NEDI KOLOKOTRONI[4], AND GEORGIOS ELLINAS[1,2], (Senior Member, IEEE)

[1]Department of Electrical and Computer Engineering, University of Cyprus, 1678 Nicosia, Cyprus
[2]KIOS Research and Innovation Center of Excellence, University of Cyprus, 1678 Nicosia, Cyprus
[3]Eratosthenes Centre of Excellence, 3012 Limassol, Cyprus
[4]Cyta, Strovolos, 1396 Nicosia, Cyprus

Corresponding author: Konstantinos Manousakis (manousakis.konstantinos@ucy.ac.cy)

This work was supported by the European Union's Horizon 2020 research and innovation programme under grant agreement No 739551 (KIOS CoE - TEAMING) and from the Republic of Cyprus through the Deputy Ministry of Research, Innovation and Digital Policy. This work was also supported under the Cyta-KIOS Collaboration Framework.

ABSTRACT This work considers project scheduling and planning for the decision support of organizations, that continuously require the implementation of many projects for their successful operation. The efficient scheduling and planning of these projects is essential for the timely completion of all projects, utilizing the appropriate resources. To this end, this work presents a novel integer linear program (ILP) formulation, that takes into account the project requirements, the involved teams, their interdependencies, as well as other constraints, so as to provide an optimal scheduling plan. Moreover, additional constraints are considered to address practical challenges such as complexity and uncertainties. Finally, techniques are introduced in this work to address scalability issues, as well as dynamic changes that may occur when the obtained schedule is currently being implemented. All aforementioned techniques present a number of advantages for an organization, as they reduce considerably the person-hours required by the management team to perform the scheduling, they produce scheduling plans that span large planning horizons, and they decrease the project completion times, thus reducing the cost that the organization incurs for implementing the projects. Realistic scenarios are considered, where real data on projects and teams are taken into account. From the results obtained, it is evident that the proposed ILP can obtain the optimal solution in terms of minimizing the duration for the completion of all projects, while the proposed practical (heuristic) approaches, can obtain solutions close to the optimal in terms of planning horizon and objective score with significant reduction in computation time, from hours to seconds/minutes. Moreover, it is shown that the scheduling plan can be adapted in the event of miscalculations related to the effort required for implementing the projects or new projects can be added within the existing scheduling plan.

INDEX TERMS Project scheduling and planning, decision support systems, integer linear programming.

I. INTRODUCTION

Organizations often struggle to deliver projects on time, within budget, utilizing the right teams of personnel, and with the required quality. The number one cause for this problem is related to the complexity of effective project management and, in particular, sub-optimal project scheduling and poor team staffing. Creating a schedule for a set of projects is an essential activity in organizations, that is important for their efficient operation. When scheduling a plan for the organization's projects, the allocation of resources to project tasks, as well as issues related to time, skills, project complexity, and budget, amongst others, must be addressed. Several organizations are creating their project scheduling

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

in an ad-hoc manner, a process which requires a significant number of person-hours from the management team. Further, with such a process, and due to the problem's complexity, the resulting scheduling plan will most likely be far from optimal, with detrimental results for the organization, as it will have higher project completion times and thus higher cost for the implementation of the projects. In addition, the planning horizon with the manual process can only be limited to a small time span and hence, a number of projects remain unplanned creating issues (and possible complaints) to their stakeholders. Therefore, optimal or near-optimal techniques are required for the problem of project scheduling. The advantages of such techniques include a reduction of the effort/cost required by the management team to perform the scheduling, the planning of schedules for projects spanning very large planning horizons, and the reduction of the projects' completion times, that translates directly to the reduction of the organization's cost for implementing the projects, as well as the increase of the number of projects that can be implemented within a specified planning horizon.

Research on project scheduling problems has received considerable attention over the years, mainly due to the practical interest of several organizations and the increased complexity of these problems, that now have to account for a large number of company-specific constraints and a diverse pool of employees with different expertise and skill sets [1], [2], [3], [4].

There are several problem variants, objectives, and methods that have been considered over the years for the project scheduling problem, which is mainly known as resource-constrained project scheduling (RCPS). The problem variants deal with different specifications that arise on the application under study and require different constraints to be implemented. In addition, several objectives are considered in the literature to solve the RCPS problem related to completion times and the cost of the projects [5]. In terms of constraints, as discussed later in Sections III and IV, there are a host of constraints that must be satisfied, stemming from the general scheduling problem, as well as custom requirements specified by the organization in regards to staffing, idle times, etc.

In general, the goal of this work is to create and apply innovative intelligent algorithms for making informed decisions on project scheduling and task planning, considering current project requirements in order to achieve on-time project delivery. Providing the earliest possible completion times to scheduled projects, allows the organization to increase its revenue stream, as well as to implement additional projects, thus facilitating even higher earnings for the organization.

Specifically, this work aims to address the projects' scheduling problem of any organization, by developing innovative techniques for the scheduling and planning of projects to: (a) enable efficient (optimal) project and task scheduling with regards to the user's requirements, the involved teams, their interdependencies, as well as other parameters; (b) enable the addition of new projects to the scheduling plan without affecting the previous plan; (c) reschedule previous projects that could not be implemented on time due to unpredictable factors, so as to deal with uncertainties; and (d) provide the capability to divide the problem into groups of projects and solve the problem sequentially, so as to significantly reduce the running time.

Overall, the contribution of the work can be summarized as follows:

- Implementation of a novel integer linear program (ILP) formulation to solve a real-world scheduling problem, where numerous constraints are imposed. An optimal solution is provided for this problem, which includes objective terms considering both project-, as well as person-based metrics.
- An incremental method for solving the scheduling problem is also proposed, where previous solutions can be incorporated (i.e., previous scheduling plans, projects under implementation, etc.) into a new scheduling plan without any effect on the current schedule. The proposed incremental approach, in combination with the designed model, also allows for manual changes to the schedule that could potentially violate several previously-defined rules. Such manual changes could be posed by the organization during the practical implementation of the plan, mainly to meet unforeseen needs.
- A sequential method is further proposed for scalability purposes, allowing the implementation of practical problems with a larger number of projects, and increased interdependencies. The sequential approach works as a hybrid ILP-based heuristic approach, where a set of projects are divided into smaller groups to solve the problem sequentially. This is done in an effort to reduce the overall computational complexity of the problem, while providing an efficient scheduling solution for each sub-group.
- An extensive simulation campaign is performed, utilizing real-world data, in order to validate the applicability and feasibility of all developed techniques.

The rest of the paper is organized as follows. Related work on project scheduling is presented in Section II, followed by the problem description in Section III. Then, the proposed ILP formulation is presented in Section IV. Section V describes the proposed incremental and sequential methods that deal with the solution of the real-world problem, while performance evaluation results and discussion are included in Section VI. Finally, Section VII offers some concluding remarks, as well as future research directions.

II. RELATED WORK

Comprehensive surveys of constraint-based scheduling are provided in [1], [3], and [6], while more recent advances on the topic regarding different requirements, methodologies and objectives are presented in [4] and [5]. The basic resource-constrained project scheduling problem (RCPSP) is comprised of a single project, consisting of a set of tasks, that must be scheduled to the available resources for execution. There are three main extensions of the classical RCPSP: (a) multi-project RCPSP in which more than one projects are to be scheduled independently of each other concerning their tasks [5], (b) multi-skill RCPSP in which resources are typically employees that have different skills suitable to execute particular tasks [7], and (c) multi-mode RCPSP in which some tasks can be associated with more than one resources [8]. The reader should note that the model presented in this work is in essence a combination of the first two extensions. A feasible solution to the RCPSP is defined by assigning starting times to each task in each project in order to minimize the projec's completion time, while at the same time satisfying the project, tasks, and resource constraints [9].

In general, the RCPSP can be classified into several categories based on the criteria under study. These criteria can be the (i) variant/model of the problem, (ii) the objective function, and (iii) the method used to solve the problem. Each variant of the problem is defined by the different requirements that must be taken into account depending on the specific application and, for each variant, a set of constraints have to be satisfied in order to implement the specific requirement. For example, when scheduling workflows in a multi-processor embedded system, the objective is to keep the schedule length of workflows as short as possible, while satisfying a number of constraints such as energy consumption, deadlines, and data dependencies [10]. In general, various constraints that are taken into consideration in each variant of the problem include the following: no task can be left unassigned, the employees assigned to a particular task must satisfy the skills required by the task, and the employees must not exceed their maximum dedication time [11]. In addition, other variations of the problem may require that each task cannot be interrupted until it is finished (non-preemptive scheduling) [12], or some tasks may be interrupted (preemptive scheduling) without any restriction [13], or at most one interruption may be allowed [14], or multiple interruptions may be allowed taking into account the additional time required to resume the task [15]. In this work, the requirement is that each task cannot be interrupted until it is finished and furthermore the idle time between the tasks of the same project is minimized.

Regarding variations related to resource constraints, some models classify employees into different skills and skill levels (e.g., beginner, junior, senior, expert), requiring a particular employee skill level or more than one skills for each task [16], [17]. Other models characterize employees with a productivity attribute that corresponds to the speed with which they complete their assigned tasks [18], [19]. In this case, constraints can also be applied on which tasks must be assigned to an employee whose productivity level is high enough to complete the corresponding tasks within a predefined deadline. Further, other constraints can also be added to the model, including a period of availability (planned unavailability) for each employee [20]. Finally, in other models, the projects are defined as work packages, instead of tasks, with no interdependencies between them, and the employees are grouped in teams according to their skill set [21]. It should be noted that in this work, in regards to the resource-constrained variations, our model considers a set of teams that have different skills to perform a task, allows for planned unavailability for each employee (defined for specific periods), and also allows for task interdependencies and (partial or full) task overlap. Depending on the problem variant, other constraints may also require that tasks cannot finish after the established deadline [22], [23], some tasks must be assigned to at least one employee that possesses the skill level required by the task [17], there is a limit on the resources that can be assigned to the tasks [24], no employee can be assigned simultaneously to more than one tasks at any given time [25], and so on. Furthermore, in [26] the projects to be scheduled are given a weight of importance providing priority to the projects with higher weight. All these constraints are also taken into account by our model considering also the fact that the tasks are implemented as close to the deadline as possible (i.e., it is not desirable to implement a project prior to or after its deadline).

Moreover, due to uncertainties related to the task duration, the model used in [8] considers a varying, instead of a constant, effort for an employee assigned to a particular task, while the models in [27] consider the scheduling problem with requests that vary over time. In addition, authors in [28] propose a real-time reactive scheduling approach to minimize the project completion times in order to deal with uncertainties and task disruptions. On the contrary, authors in [29] focus on the proactive resource-constrained project scheduling problem in which each activity can be split at discrete time instants, under the constraints of a maximum number of activity splitting and a minimum period of continuous execution, demonstrating that activity splitting improves robustness. In this work, in the event of changes in the scheduling plan, the affected tasks can be rescheduled without however affecting the overall scheduling plan.

In regards to the objectives of the RCPSP, one of the main objectives in the literature [5] is to minimize the total time (makespan) for the completion of all projects and/or to minimize the cost associated with the completion of all projects. Another objective is to minimize the project implementation delay compared to the project due date. The difference between the start and end dates of the project implementation is also a known objective, called ‘‘flow time objective’’. In addition, an objective related to project implementation is project splitting (idle time between tasks), which is required to be minimized so as to minimize the project overhead. Based on the goals of an organization, other objectives can also include: (a) the minimization of the overtime level (e.g., when the employees exceed their maximum designated work time), and (b) the minimization of the amount of time that a team can remain inactive as a consequence of the sequence in which projects/tasks are implemented. There are models that consider one, two, or more objectives to minimize (i.e., single-, multi- or many-objective, respectively) [30]. Recently, other models aim to optimize the robustness of the schedule, that is, to minimize the difference in the completion times when new tasks are added to the project or when some tasks require additional time compared to what was originally planned [29]. In fact, each objective of the problem is also closely related to the problem variant and specific constraints are required to define each one of the objectives. For example, five different objectives are considered in our model, related to the project completion times, the deadlines, and the teams working on each project, and each one requires a set of constraints to be implemented.

Apart from the variants proposed and the objective functions considered, several algorithms and techniques have also been developed to solve the scheduling problem, ranging from exact (optimal) solutions using ILPs [8], [12] to sub-optimal solutions utilizing heuristic algorithms (heuristics, metaheuristics, and optimization/heuristic hybrid approaches) [5], [12]. This is the case, since the problem is difficult to solve (proven to be NP-hard [31]), thus heuristic algorithms are required to solve real-world instances that are required by the organizations. Metaheuristic algorithms such as ant-colony optimization (ACO), particle swarm optimization (PSO), and genetic algorithms (GAs) have also been used to address the complexity of the problem [14], [29], [30], [32], [33], [34], [35], [36]. For instance, the authors in [30] and [32] solve the software project scheduling problem using an ACO metaheuristic, obtaining better results in terms of running time or problem objective for some instances compared to other algorithms using other methods such as genetic algorithms. In [14], the authors solve the resource-constrained project scheduling problem, in which one interruption per task is allowed, using the PSO metaheuristic algorithm. In a similar vein, in [29], the authors provide a solution to the same problem (when now the tasks are allowed to be interrupted at discrete times) using a GA, while in [33] the same problem is addressed using a quantum-inspired GA that differs from the classical GA in the way the initial and the updated populations are implemented. In [34] authors investigate the resource-constrained project scheduling problem with resource transfer times under uncertain environment using GA. Hybrid approaches [37] combine the power of different kind of algorithms (ILPs, metaheuristics, heuristics) to solve large scale problems. For example in [37], authors use a combination of tabu search and simulated annealing to solve the RCPS problem. In addition, the authors in [35] and [36] solve variations of the RCPS problem using a combination of several heuristic and metaheuristic algorithms to address large scale instances. Finally, machine learning techniques (e.g., such as reinforcement learning) can be employed to solve scheduling problems in dynamic environments where agents cooperate to achieve group missions [38]. In our work, an ILP with typical and novel constraints (Section IV) is formulated to optimally solve the problem under consideration. In addition, two techniques called ‘‘incremental’’ and ‘‘sequential’’ (hybrid approach - ILP-based heuristic) are proposed to solve the challenges of a real-world practical problem (e.g., take into account previous solutions, solve problems with many projects, large size groups, and a large number of interdependencies).

The reader should note that the solution developed in this work differs from other state-of-the-art solutions, as it is a uniquely considered variant of the RCSPS problem, with a combination of objectives and constraints that cannot be found in the literature, and with a set of techniques for utilizing the proposed solution in real-world problems. Thus, it is not practical to perform a quantitative comparison between the developed approach and approaches in the state of the art. Nevertheless, a qualitative comparison is performed below and Table 1 below presents a number of established RCPSP approaches presented in the literature and how they compare with the proposed solution.

More specifically, in terms of the objective functions found in the literature, the most common is to minimize the project completion time, while in some works a relative weight factor for each project is also considered. Other approaches consider the project deadlines as optimization objective and one approach considers the earliest release of resources (ERR) which means that more resources are assigned to earlier time-slots so as to implement a task. Our approach considers all the above objectives and also two additional ones that are not found in the literature that consider minimizing both the idle time between tasks, and the time duration of each project.

In terms of the features found in the literature, besides the mandatory ones to implement a scheduling plan, all the works presented in Table 1 consider the precedence feature which defines the ordering to implement each task and the unavailability which specifies the available resources at each time-slot. Additionally, some works consider only one project while other consider many projects. In some cases, the non-preemptive constraint is relaxed. Another feature that is present in our work, is the consideration of previous project scheduling (PPS) solutions, where PPS can be added to a new scheduling plan, and can be found in works that consider robust optimization. Two other characteristics that are not found in the literature is the overlapping constraint that allows some tasks to overlap at a certain percentage and the idle slot constraint that do not allow to teams that worked at a project to work at the first time slot after the project completion. This is mainly used in practical problems, since organizations require the teams that have worked on a project which was just completed, to address any possible requirement that may arise.

It should also be noted that there are several commercial project management tools (e.g., Oracle’s Primavera [39] and Microsoft’s MS-Project [40]), that can be used for project management, taking as input the scheduling plan. In addition,

TABLE 1. Qualitative comparison with state-of-the-art approaches.

Papers	Objective Function					Features						
	PCT	ITT	D	ERR	LS	MP	Pre	Over	IdleS	NP	UN	PPS
Our Work	x	x	x	x	x	x	x	x	x	x	x	x
[7], [13], [14], [17], [27], [28]	x	-	-	-	-	-	x	-	-	x	x	-
[8], [16]	x	-	-	-	-	x	x	-	-	x	x	-
[12]	x	-	-	x	-	x	x	-	-	x	x	x
[15]	x	-	-	-	-	-	x	-	-	-	x	-
[20], [26]	-	-	x	-	-	x	x	-	-	x	x	x
[22]	x	-	-	-	-	x	x	-	-	-	x	-
PCT: Project completion time and cost; ITT:Idle time between tasks; D: Deadline; ERR:Earliest release of resources												
LS: Less time slots to implement a task; MP: Many projects; Pre: Precedence; NP: Non-Preemptive												
IdleS: Idle slot after completion of a project; UN: Unavailability; PPS: Previous project scheduling												

there are also commercial software tools available for project scheduling (e.g., MJC2 [41] and MangoGem [42]). These tools, however, offer solutions for specific applications and cannot be used by an organization without customization.

Table 1 demonstrates that the proposed technique outperforms other state-of-the-art approaches as it accounts for more objectives and additional features, making it more flexible and better applicable to practical scenarios. In addition, the proposed scheme is dynamic and can be easily adjusted to address any additional requirements; new projects can be embedded to previous solutions and old solutions can remain either unchanged or can be changed according to the new requirements. Moreover, solution optimality can be traded with computational complexity, i.e., fast solutions can be obtained that are nevertheless sub-optimal. Subsequently, each solution obtained can be evaluated and compared to the optimal solution.

Clearly, there are a number of challenges in solving the problem considered in this work, with the main being the large number of requirements and team interdependencies that must be considered (translating in a large number of constraints) when devising the ILP formulation, as well as the design of practical techniques that can provide "good" solutions (in terms of objective score and planning horizon) in short time in order to address real-world requirements imposed by the organizations. All these issues are addressed in the sections that follow below.

III. PROBLEM DESCRIPTION

In general, the project scheduling approach involves the following steps: (i) each project is categorized based on its importance and it is divided into several tasks; (ii) the teams/skills that are required to implement each project are identified; and (iii) project scheduling and planning is performed based on the aforementioned data, as well as additional parameters such as project cost, project time bounds, specific order of project completion, interdependencies between projects/tasks, etc. However, these steps are usually performed manually by several organizations, which in turn leads to poor scheduling performance.

The main goal of this work is to develop an automated solution to the scheduling problem through the usage of an intelligent approach that obtains optimal (or close to optimal) solutions in a time- and cost-efficient manner. In this case, it is important to consider the problem's requirements that are tightly coupled to any organization's processes, needs, and interdependencies. In the following, the system's requirements as well as the problem inputs and objective are described.

A. SYSTEM REQUIREMENTS

The rules (requirements) that describe the scheduling problem under consideration are the following:

- Each project is divided into tasks;
- Each task can be implemented by only one team based on the skills of the personnel comprising the team;
- A certain amount of time is required for the implementation of each task;
- No task can be left unassigned;
- Each task cannot be implemented with gaps in time (i.e., when a task commences, it cannot be interrupted);
- Some tasks have specific ordering – a "high" ordering (i.e., high priority) task must be implemented first and then the rest (with a "lower" ordering). Task ordering is used as team ordering as well, since each task can be implemented by only one team;
- A specific number of people is associated to each team;
- The efficient time for the person within a team working on a task is $et\%$ of their time;
- Each team cannot work on more than P_{max} projects at the same time;
- For tasks with specific ordering, a certain percentage of overlapping is allowed;
- After the implementation of a project, the teams that worked on that project cannot work on another project for a specified amount of time (e.g., the next time slot) due to any maintenance requirements that may arise for the project that was just completed;
- A number of persons that belong to a team may not be available to work at specific time slots;
- Each project has a weight (score) based on its importance;
- A previous project scheduling solution can be given as an input to be incorporated within the current scheduling

problem. The state of active projects (i.e., projects under implementation) is given as input to the problem by specifying the percentage of completion of each task. Active tasks that are not completed must commence at the start of the next scheduling period;

- A project with a specific deadline must be implemented as specified by the deadline (whenever possible), or as close to the deadline as possible;
- The final schedule must fit within the actual planning horizon start/end dates.

B. PROBLEM OBJECTIVE

The objective of the current work is to provide a project scheduling solution, that minimizes the project completion times, taking into account the score of each project (projects with higher score are implemented with higher priority), while following all rules outlined above. Further, the proposed solution aims to meet all deadlines (if possible), as well as minimize the offset of any project implementation (i.e., minimize the time required to finish a project as specified by the deadline), if any of the deadlines cannot be met. Finally, the objective aims to minimize the number of time slots required to implement each task and at the same time to maximize the number of persons of each team to be involved at the implementation of the task as soon as possible.

C. PROBLEM INPUTS

The basic inputs to the proposed algorithm, that are required to define the problem, include: (i) the teams with the number of employees comprising each team, (ii) the interdependencies between teams, (iii) the priority of the teams, (iv) the list of projects, (v) the importance of each project (i.e., its score), based on several criteria defined by the organization, (vi) the task breakdown for each project (with one team working on each task), (vii) the full time equivalent (FTE) that is required by a team to work on a project (with the corresponding time units, e.g., weeks, months), (viii) the unavailability of team members for specific dates, (ix) the required deadlines for some of the projects, and (x) an overlapping parameter, that refers to a task's percentage that is allowed to be completed prior to start overlapping with other interdependent tasks.

D. AN ILLUSTRATIVE EXAMPLE

Prior to describing the project scheduling formulation, a toy example is provided below to illustrate what constitutes the best scheduling solution. Specifically, in the example provided below, 4 projects are considered, each comprised of $2-4$ tasks. As shown in Table 2, to schedule project p_1, two tasks must be performed that require 2, and 4 person-weeks by teams f_1, and f_2, respectively. Further, the number of persons in each team is shown in Table 3, along with the team priority. Regarding team priority, if for example teams f_2 and f_3 must both work on project p_3, team f_3 must finish an amount of its task (based on the overlapping percentage) in order for f_2 to start working on its own task.

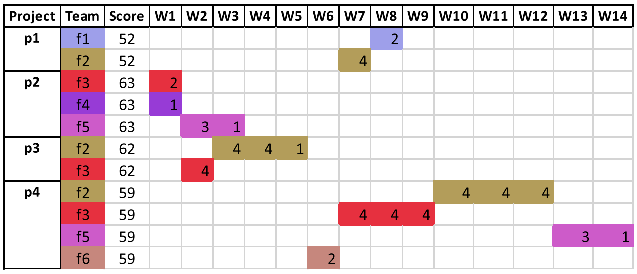
TABLE 2. Example with 4 projects to be scheduled.

Project Id	Tasks required per project Team - FTEs)			
p_1	f_1-2	f_2-4		
p_2	f_3-2	f_4-1	f_5-4	
p_3	f_2-9	f_3-4	$-$	
p_4	f_2-12	f_3-12	f_5-4	f_6-2

TABLE 3. Teams working on the 4-project scenario of Table 2.

Team Id	Number of persons in team	Priority
f_1	2	4
f_2	4	3
f_3	4	2
f_4	1	2
f_5	3	4
f_6	2	1

One possible solution is depicted in Fig. 1, where the projects are scheduled one by one, starting with the project with the highest score (higher importance). This solution satisfies all the rules (requirements) presented in Section III-A. $W_1 - W_{14}$ are the weeks that are required to implement all the projects. The number under each week declares the number of people from each team that have to work that week in each of the projects. It is assumed that each team can work to at most 2 projects during each specific week. As can be seen from Fig. 1, the priorities among the teams are satisfied. Additionally, in this example, project p_1 cannot be implemented earlier (start at week W_6 instead of W_7) since at week W_6, team f_2 is required to work at project p_3 (finished at week W_5) due to maintenance requirements. Finally, according to the objective, the number of persons from each team that are working for the implementation of each task are involved as soon as possible (i.e., more people are assigned to earlier time-slots). For example, the allocation of team f_2 to project p_3 is $4-4-1$ on weeks $W_3 - W_4 - W_5$, respectively, and not $3-3-3$ or $1-4-4$.

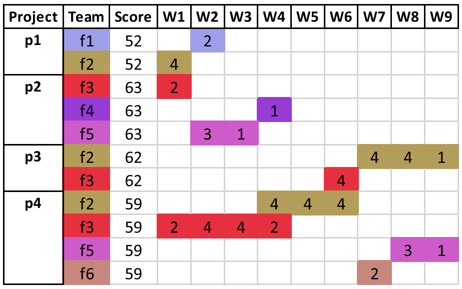


Project	Team	Score	W1	W2	W3	W4	W5	W6	W7	W8	W9	W10	W11	W12	W13	W14
p1	f1	52								2						
	f2	52							4							
p2	f3	63	2													
	f4	63	1													
	f5	63		3	1											
p3	f2	62			4	4	1									
	f3	62		4												
p4	f2	59										4	4	4		
	f3	59							4	4	4					
	f5	59													3	1
	f6	59						2								

FIGURE 1. Toy example with the 4 projects not optimally scheduled.

While the solution presented in Fig. 1 satisfies all problem constraints, it is not optimal. Another solution to this problem that constitutes the optimal one (based on the objective presented in Section III-B) is depicted in Fig. 2, where all the rules of the problem are again satisfied. As can be seen, even for this small problem, the required scheduling weeks are decreased from 14 to 9. In this example, teams f_2 and

f_3 are working concurrently on project p_4 during week W_4, since team f_3 already implemented 75% of its work on the project (assuming an overlapping percentage equal to 25%). Therefore, efficient scheduling techniques are required to address the problem, which is the goal of sections that follow.



Project	Team	Score	W1	W2	W3	W4	W5	W6	W7	W8	W9
p1	f1	52		2							
	f2	52	4								
p2	f3	63	2								
	f4	63				1					
	f5	63		3	1						
p3	f2	62							4	4	1
	f3	62						4			
p4	f2	59				4	4	4			
	f3	59	2	4	4	2					
	f5	59								3	1
	f6	59							2		

FIGURE 2. Toy example with 4 projects scheduled optimally.

IV. PROJECT SCHEDULING FORMULATION

This section describes the proposed ILP formulation for the efficient scheduling and planning of a set of projects.

A. ILP FORMULATION

The parameters and variables of the ILP formulation as well as other symbols utilized in this work are included in Table 4.

ILP Objective:

$$\textit{Minimize}: \mathrm{w}_1 \cdot \sum_{p\notin P_d} c_p \cdot S^e_p + \mathrm{w}_2 \cdot \sum_{p\in P}\sum_{w\in W} y_{pw} + \mathrm{w}_3 \cdot \sum_{p\in P_d} A_p +$$

$$+\mathrm{w}_4 \cdot \sum_{f\in TM}\sum_{t\in T}\sum_{w\in W} w \cdot z^f_{tw} + \mathrm{w}_5 \cdot \sum_{p\in P}\sum_{t\in T_p}(S^e_{pt} - \sum_{w\in W} v_{tw} \cdot w)$$

The objective aims at minimizing the project completion times taking into account the score of each project (without a given deadline) as defined by the first term of the objective, $\sum_{p\notin P_d} c_p S^e_p$. The second term, $\sum_{p\in P}\sum_{w\in W} y_{pw}$, aims at minimizing the idle time between the implementation of different tasks of the same project and the third term $\sum_{p\in P_d} A_p$ aims to achieve all the given deadlines, (i.e., minimize the time-slots required to schedule a project prior to or after its given deadline). Further, the fourth term $\sum_{f\in TM}\sum_{t\in T}\sum_{w\in W} w \cdot z^f_{tw}$ aims to assign more persons at the initial time-slots during the implementation of each task. Finally, the fifth terms, $\sum_{p\in P}\sum_{t\in T_p}(S^e_{pt} - \sum_{w\in W} v_{tw} \cdot w)$, minimize the required time-slots to implement each task. It is noted that weights $\mathrm{w_i}$ are used to define the relevance importance of each term in the objective.

Subject to the following constraints:

Every task must be allocated to a team and must be implemented for a certain number of time slots:

$$\sum_{f\in TM}\sum_{w\in W} z^f_{tw} = PS_t, \quad \forall t \in T \tag{1}$$

TABLE 4. Variables and Constraints of the ILP formulation.

Symbol	Parameters
P	set of projects
p	a project, $p \in P$
T	set of tasks
t	a task, $t \in T$
T_p	set of tasks in project p
PS_t	required person-slots for task t (i.e., number of time slots required to implement the task, assuming one person is working on that task)
W	set of time slots
w	a time slot (e.g., week), $w \in W$
TM	set of teams
f	a team, $f \in TM$
TM_f	number of persons in team f
T_{ft}	equal to 1 if task t cannot be accommodated by team f; equal to 0 otherwise
TO_p	set of tasks in project p that must be implemented in a specific order, $TO_p \subseteq T_p$
TD_{pt}	order in which task t ($t \in TO_p$) must be implemented in project p. Lower order value signifies that a task must be performed first
TU_{fw}	number of persons of team f that are unavailable at time slot w
B	a big constant
c_p	cost that represents priority of project p (i.e., the greater the cost the higher the priority in terms of project ending time)
TMP_{pf}	equal to 1 if team f must work on project p, and 0 otherwise
E_p	deadline of project p (given as time slot index)
Z_{fw}	equal to the number of tasks that team f has been assigned to at time slot w. This parameter is used only if a previous solution exists
P_d	set of projects that have a deadline, $P_d \subseteq P$
$\mathrm{w_i}$	weights used in the objective function, $i \in \{1, 2, 3, 4, 5\}$
α	completion percentage of a task prior to start overlapping with other interdependent tasks
	Variables
x_{tw}	Boolean variable equal to 1 if task t is being implemented at time slot w; equal to 0 otherwise
z^f_{tw}	Integer variable equal to the number of persons of team f assigned to task t at time slot w
u_{pw}	Boolean variable equal to 1 if project p is active at time slot w; equal to 0 otherwise
y_{pw}	Boolean variable equal to 1 if time slot w is the starting time slot to implement project p or part of the project (some tasks of the project may have a gap in time); equal to 0 otherwise
v_{tw}	Boolean variable equal to 1 if time slot w is the starting time slot to implement task t; equal to 0 otherwise
S^e_{pt}	Integer variable equal to the ending time slot of task t that is in project p
S^e_p	Integer variable equal to the ending time slot of project p
b_t	Boolean variable, used as control variable to correctly select the ending time for a project
q_{tw}	Continuous variable equal to the completion percentage of task t at time slot w
r_{tw}	Boolean variable equal to 1 if task t can be performed at time slot w; equal to 0 otherwise
o^f_{tw}	Boolean variable equal to 1 if team f works on task t at time slot w; equal to 0 otherwise
G_{pw}	Boolean variable equal to 1 if project p is completed at time slot w; equal to 0 otherwise
F_{pw}	Boolean variable equal to 1 if w is the time slot following the completion date of project p; equal to 0 otherwise
A_p	Integer variable equal to the offset (number of time slots) from project's p completion time (based on given deadline)

Maximum number of persons that can work per time slot for each team:

$$\sum_{t \in T} z_{tw}^{f} \leq TM_f - TU_{fw}, \quad \forall f \in TM,\ w \in W \tag{2}$$

Tasks cannot be allocated to teams that do not have the required skills:

$$\sum_{\substack{f \in TM \\ s.t.\ T_{ft}=1}} \sum_{w \in W} z_{tw}^{f} = 0, \quad \forall t \in T \tag{3}$$

If a task is assigned to a time slot then a team must be active:

$$\sum_{f \in TM} z_{tw}^{f} \leq B \cdot x_{tw}, \quad \forall t \in T,\ w \in W \tag{4}$$

$$x_{tw} \leq \sum_{f \in TM} z_{tw}^{f}, \quad \forall t \in T,\ w \in W \tag{5}$$

If a task is assigned to a time slot then the project must be active:

$$x_{tw} \leq u_{pw}, \qquad \forall t \in T_p, p \in P,\ w \in W \tag{6}$$

If a project is active then at least one task must be active:

$$u_{pw} \leq \sum_{t \in T_p} x_{tw}, \quad \forall p \in P,\ w \in W \tag{7}$$

Idle time between the tasks of a project:

$$u_{pw} - u_{p(w-1)} \leq y_{pw}, \qquad \forall p \in P,\ w \in W \tag{8}$$

$$\text{Case } w = 1, \text{ then } u_{p(w-1)} = 0$$

Tasks must be implemented without gaps:

$$x_{tw} - x_{t(w-1)} \leq v_{tw}, \quad \forall t \in T,\ w \in W \tag{9}$$

$$\text{Case } w = 1, \text{ then } x_{p(w-1)} = 0$$

$$\sum_{w \in W} v_{tw} \leq 1, \quad \forall t \in T \tag{10}$$

Ending time-slot of a task:

$$w \cdot x_{tw} \leq S_{pt}^{e}, \quad \forall t \in T_p,\ p \in P,\ w \in W \tag{11}$$

Upper bound for task ending time:

$$S_{pt}^{e} \leq \sum_{w \in W} (v_{tw} \cdot w + x_{tw}) - 1, \quad \forall t \in T_p,\ p \in P \tag{12}$$

Task ordering:

$$S_{pt}^{e} < S_{pt'}^{e}, \quad \forall p \in P, t, t' \in TO_p,\ s.t.\ TD_{pt} < TD_{pt'} \tag{13}$$

Ending time slot of a project:

$$S_{p}^{e} \geq S_{pt}^{e}, \quad \forall t \in T_p,\ p \in P \tag{14}$$

$$S_{p}^{e} \leq S_{pt}^{e} + (1 - b_t) \cdot B, \quad \forall t \in T_p,\ p \in P \tag{15}$$

$$\sum_{t \in T_p} b_t = 1, \quad \forall p \in P \tag{16}$$

Time slots at which each task completion percentage is greater than α:

$$q_{tw} \geq \alpha - B \cdot (1 - r_{tw}), \quad \forall t \in T_p,\ p \in P,\ w \in W \tag{17}$$

$$\alpha \geq q_{tw} + 1 - B \cdot r_{tw}, \quad \forall t \in T_p,\ p \in P,\ w \in W \tag{18}$$

Completion percentage of a task:

$$q_{tw} = \sum_{f \in TM} \frac{z_{tw}^{f}}{PS_t} + q_{t(w-1)}, \quad \forall t \in T_p,\ p \in P,\ w \in W \tag{19}$$

$$\text{Case } w = 1, \text{ then } q_{t(w-1)} = 0$$

Allocation of a task to allowed time slots (based on completion percentage):

$$x_{tw} \leq r_{t'w}, \quad \forall p \in P, t, t' \in TO_p,\ s.t.\ TD_{pt'} < TD_{pt}, w \in \mathrm{W} \tag{20}$$

Calculate if team f works on task t at time slot w:

$$B \cdot o_{tw}^{f} \leq z_{tw}^{f} + B - 1, \quad \forall f \in TM,\ t \in T,\ w \in W \tag{21}$$

$$B \cdot o_{tw}^{f} \geq z_{tw}^{f}, \quad \forall f \in TM,\ t \in T,\ w \in W \tag{22}$$

A team must work at most on P_{max} projects during a time slot:

$$\sum_{t \in T} o_{tw}^{f} \leq P_{max}, \quad \forall f \in TM,\ w \in W \tag{23}$$

Calculate when the ending time is reached for each project:

$$S_{p}^{e} + B \cdot G_{pw} \geq w \quad \forall w \in W,\ p \in P \tag{24}$$

$$S_{p}^{e} + B \cdot G_{pw} \leq -0.001 + w + B \quad \forall w \in W,\ p \in P \tag{25}$$

Find the next time slot after each project is completed:

$$F_{pw} \geq G_{pw} - G_{p(w-1)} \quad \forall p \in P,\ w \in W \tag{26}$$

$$\text{Case } w = 1, \text{ then } G_{p(w-1)} = 0$$

Teams that worked on a project cannot work on any other task at the first time slot after a project has been completed:

$$TM_f \cdot F_{pw} + z_{tw}^{f} \leq TM_f - TU_{fw}, \\ \forall t \in T,\ w \in W,\ p \in P,\ f \in TM,\ s.t.\ TMP_{pf} \neq 0 \tag{27}$$

Calculate any offset in the project completion time based on its deadline:

$$A_p \geq S_{p}^{e} - E_p, \quad \forall p \in P_d \tag{28}$$

$$A_p \geq E_p - S_{p}^{e}, \quad \forall p \in P_d \tag{29}$$

Specifically, Constraint (1) ensures that the person-slots required to implement a task are assigned to specific time slots, while Constraint (2) is used to ensure that the number of persons from a team assigned to a time slot cannot exceed the total number of persons of that team or the available number of persons of that team at the specific time slot w. Constraint (3) defines the specific teams that can implement specific tasks and Constraint (4) ensures that each task must be allocated to specific time slots. Constraints (5) –

(7) correlate different variables in regards to active teams, projects, and tasks. More specifically, Constraints (5) and (6) ensure that a team and a project are active, respectively, in the case that a task is assigned to a time slot, while Constraint (7) ensures that in the case that a project is active, then at least one task of this project must be active as well. Constraint (8) is used to define the number of idle time slots between tasks of the same project (as previously mentioned, this number is minimized in the objective function). Specifically, this constraint (inspired by constraints in [43]) counts the transitions of a project p from inactive state (value of y_{pw} equal to zero) to active condition (value of y_{pw} equal to one) at time slot w. Constraints (9) – (10) are used to prevent gaps in the implementation of a task [the logic of Constraint (9) is similar to Constraint (8)]. Further, Constraints (11) and (12) are used to define the ending time of each task and to calculate the upper bound for each task's ending time, respectively.

In addition, Constraint (13) defines a project's task ordering, where necessary. Constraints (14) - (16) define the ending time of each project, and Constraints (17) - (20) ensure that a task can only be assigned to specific time slots, based on the completion percentage of other tasks within the same project. Specifically, Constraints (17) and (18) calculate the time slots at which each task's completion percentage is greater than α, Constraint (19) calculates the completion percentage of a task i, and Constraint (20) ensures that the allocation of task j that is interdependent with task i, can be only assigned to the time slots after the completion of α of task i. Constraints (21) - (22) specify whether team f works on task t at time slot w, and Constraint (23) defines the maximum number of projects that a team can work on concurrently. Constraints (24) - (27) are used to specify that the teams that worked on a project cannot work on any other task on the first time slot (i.e., the first week in this implementation) after a project is completed. More specifically, Constraints (24) and (25) calculate the ending time of each project, and Constraints (26) and (27) calculate the next time slot (e.g., week) after the ending time of a project, and ensure that all persons of a team that worked on that project are available, and cannot work on any other project on that specific time slot, respectively. Finally, Constraints (28) - (29) are used to calculate any offset (before or after) in the project's ending time, based on its deadline.

B. NUMBER OF VARIABLES AND CONSTRAINTS

The number of constraints and variables of the problem is mainly affected by four parameters: number of projects ($|P|$), number of tasks ($|T|$), number of teams ($|TM|$), and number of time slots ($|W|$). Other parameters (e.g., interdependencies between tasks) can also have an impact on the number of constraints created. Table 5 presents the number of variables and constraints required in the ILP formulation.

As expected, as the problem size increases, the number of variables and constraints increases, which will have an effect on the execution times and the memory usage. Table 6 illustrates the number of variables and constraints based on 4 input parameters for 5 specific instances. For example, for a problem of 60 projects and 127 tasks, with 14 teams and 220 weeks as a scheduling horizon, the total number of variables is equal to 947, 254 (up from approximately 90, 000 for 20 projects), while the total number of constraints (upper bound) is on the order of 5.5 million (up from approximately 300, 000 for 20 projects).

TABLE 5. Variables and constraints of the ILP formulation.

Variables	Constraints
$(\lvert P\rvert + \lvert T\rvert)\cdot(4\cdot\lvert W\rvert + 2)+$ $+2\cdot\lvert T\rvert\cdot\lvert TM\rvert\cdot\lvert W\rvert$	$5\cdot\lvert T\rvert + 3\cdot\lvert P\rvert + \lvert TM\rvert\cdot\lvert W\rvert+$ $+\lvert T\rvert\cdot\lvert TM\rvert + 8\cdot\lvert T\rvert\cdot\lvert W\rvert+$ $+6\cdot\lvert P\rvert\cdot\lvert W\rvert + 2\cdot\lvert P\rvert\cdot\lvert T\rvert+$ $+\lvert P\rvert\cdot\lvert T\rvert^2 + \lvert W\rvert\cdot\lvert T\rvert^2+$ $+\lvert TM\rvert\cdot\lvert W\rvert\cdot(2\cdot\lvert T\rvert + 1)$

TABLE 6. Scenarios for variables and constraints.

Instance Size				Variables	Constraints
$\lvert P\rvert$	$\lvert T\rvert$	$\lvert TM\rvert$	$\lvert W\rvert$		
3	7	6	20	2500	4718
10	28	7	80	43596	126742
20	42	9	90	90484	307114
30	64	11	100	178588	749670
60	127	14	220	947254	5625159

In terms of the memory required, as the number of variables and constraints increase, the ILP solvers require more memory to solve the problems. However, solvers like Gurobi [44] can handle the problems with sizes depicted in Table 6 efficiently in terms of memory usage (the problem can be solved in a PC even for the larger instances). Clearly, based on the discussion above, scalability becomes a crucial issue when the number of projects and planning horizon scale to large numbers [since in general RCPS problems (the problem studied in this work falls under that category) are NP-complete [1]]; thus, more practical techniques are required to address the scalability issue. This is precisely the focus of the section that follows, which considers ILP-based heuristics in order to accommodate the execution of larger instances of the problem in "reasonable" time scales (i.e., within a few minutes) as can be seen in Section VI-F.

V. TECHNIQUES TO ADDRESS PRACTICAL IMPLEMENTATION ISSUES

This section describes the two proposed methods for dealing with the solution of the ILP in real-world practical scenarios. More specifically, this section deals with the incorporation of previous solutions (previous scheduling plans, active projects) into a new planning period, as well as a decomposition technique in which a set of projects are divided into smaller groups to solve the problem sequentially, aiming to reduce computational complexity and scale the problem in terms of the number of projects, group sizes, etc.

A. BUILDING ON PREVIOUS SOLUTIONS

Project scheduling for any organization usually builds on previous scheduling solutions, since new projects are often requested to be scheduled in the middle of the scheduling plan that is currently being implemented. To address the problem of building a new scheduling plan on previous scheduling solutions, the projects under consideration fall into three different categories: (i) scheduled projects, (ii) new projects, and (iii) active projects. Scheduled projects are the projects of the previous solution that are scheduled at specific time slots with specific resources, but are not yet active, and can be further distinguished in projects that must remain unchanged in the new scheduling plan, and projects that are required to be rescheduled (i.e., the resources allocated for these projects are freed and these projects must be scheduled in the new plan). In addition to these projects, new projects that must be scheduled taking into account the previous solution are also considered. Finally, to keep up with the needs of any organization, if any project is currently implemented by the organization (either as part of the previous scheduling solution or provided as a new project), it is considered as an active project, and must be scheduled at the starting time slot of the planning period. It is noted that all scheduled and active projects undergo a proposed incremental approach as is described below, while any new projects are scheduled according to the already presented formulation.

1) INCREMENTAL APPROACH: SCHEDULED PROJECTS

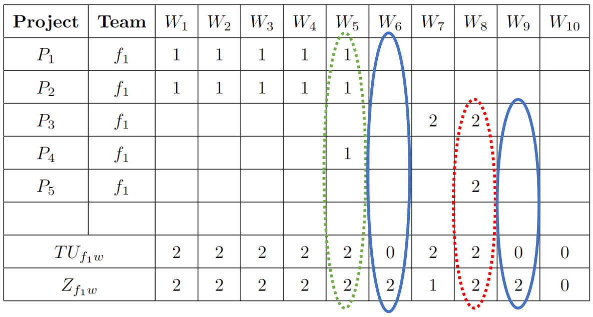
The concept of the proposed incremental approach is to build on a previous solution (scheduled projects) and create a new plan. To achieve that, the incremental approach uses the concept of team unavailability. Specifically, based on the previous solution, the teams (and the number of persons in each team) that have already been assigned to tasks and slots (during the previous scheduling process) are considered as unavailable during the new scheduling period, so as to find the new scheduling plan (i.e., allocation of the unscheduled projects). This is achieved through Constraint (2) and Constraint (27) using parameter TU_{fw} that accounts for the team unavailability of the new scheduling plan. Also, since a team is not allowed to work on more than P_{max} projects concurrently during a time slot, parameter Z_{fw} is utilized based on the number of tasks and projects that the team is working at each time slot in the previous solution, and Constraint (23) is modified as follows:

$$\sum_{t \in T} o^{f}_{tw} \leq P_{max} - Z_{fw}, \quad \forall f \in TM, \ w \in W \tag{30}$$

Moreover, to satisfy the rule which specifies that when a project p finishes at time slot $w-1$, all teams involved in that project must not work at time slot w, a preprocessing step is also performed. Specifically, Z_{fw} is equal to P_{max} in Constraint (30), to ensure that no other task can be performed during that time slot.

In addition, it is important to account for the dynamic and unexpected changes that may occur in a real environment and are not known a-priori. Under these conditions, an organization is very likely to proceed with manual changes and modifications to the scheduled plan provided by the ILP. These modifications may include allocations of teams and tasks that can violate some constraints of the formulation (i.e., higher number of employees working on a team for specific time slots, teams working on more than P_{max} projects, etc.). In general, this functionality is very important towards utilizing this approach in practical scenarios. To address this issue, the incremental approach in combination with some processing steps are used to ensure that in case any manual modifications are performed by the organization, any violated rules are allowed just for that previous solution. This is achieved through setting parameters TU_{fw} and Z_{fw} to their maximum values for these cases.

Figure 3 presents a toy example illustrating the projects planned for team f_1, that consists of 2 persons. In this scheduling example, projects p_1, p_2, and p_3 are part of the scheduling plan, whereas projects p_4 and p_5 are added manually by the organization. As shown in Fig. 3, to account for this scheduling plan, variables TU_{f_1w} and Z_{f_1w} are equal to the number of assigned persons of team f_1 and number of projects where team f_1 works at time slot w, respectively. Now, at time slot W_5, project p_4, that was added manually, violates several constraints (i.e., the number of persons working and the number of simultaneous projects are greater than 2). Nevertheless, by assigning parameters TU_{fw} and Z_{fw} to their maximum allowed values, the ILP allows this instance (i.e., all constraints are satisfied for that time slot), while at the same time it will not allocate any other (additional) resources of that team at that time slot. This is also the case at time slot W_8, where more persons than allowed are working at the same time. Also, it is important to note that to account for the time slot after the project ends (in which the teams working at each project must not be allocated to any other project), Z_{f_1w} is equal to its maximum value during time slots W_6 and W_9.



Project	Team	W_1	W_2	W_3	W_4	W_5	W_6	W_7	W_8	W_9	W_{10}
P_1	f_1	1	1	1	1	1					
P_2	f_1	1	1	1	1	1					
P_3	f_1							2	2		
P_4	f_1					1					
P_5	f_1								2		
TU_{f_1w}		2	2	2	2	2	0	2	2	0	0
Z_{f_1w}		2	2	2	2	2	2	1	2	2	0

FIGURE 3. Toy example with 5 projects that includes manual changes and modifications to the scheduled plan.

Overall, following the aforementioned approach, the algorithm can consider any previous solution when planning a new set of projects.

2) INCREMENTAL APPROACH: ACTIVE PROJECTS

As previously mentioned, active projects are projects that have started to be implemented by the organization and, therefore, these projects must not be interrupted. A project is characterized as active, if at least one task of this project is active. For this reason, when considering a new scheduling plan, the ongoing tasks of these projects (T_A) have to be scheduled by the first time slot of the new planning period, a condition that is ensured by the active project assignments constraint [i.e., Constraint (31)]

$$x_{tw} = 1, \quad \forall t \in T_A, \ w = 1 \tag{31}$$

Also, it is noted that the number of time slots during which a person is required to work in order to implement the active tasks depends on the percentage of completion of the tasks (i.e., at_t for task t), that is provided as input to the algorithm. Specifically, Eq. (32) is used to calculate the new remaining time slots (PS'_t) required as

$$PS'_t = \left\lceil \frac{(100 - at_t)}{100} \right\rceil \cdot PS_t \tag{32}$$

with PS_t being the number of time slots required when 0% of the task has been completed (provided as input to the algorithm).

B. SEQUENTIAL APPROACH - GROUP OF PROJECTS

Considering that in an organization some scheduling plans may have to be planned on-the-fly in order to support decisions or to implement "what-if" scenarios, the scheduling technique must be able to provide a solution in a relatively short period of time. Also, in these cases, obtaining the optimum solution may not be the main objective of the algorithm; rather a (sub-optimal) solution is sought that tries to balance solution quality and running time. Hence, to address the increase of computational complexity when the problem scales to a large number of projects and tasks, the projects can be separated into groups, based on each project's characteristics, priority score, etc. Clearly, in such a case there is a trade-off between optimality and computational complexity. In general, this will be decided by the organization, based on the groups in which the problem is divided to. Moreover, it is important to note that (by default) in an organization several projects can be categorized in groups of projects, depending on the importance of implementing each project.

To achieve that, the concept of the incremental approach using the team unavailability constraint can be used as an intermediate step to solve large problems in a sequential manner. For instance, consider the case where some projects must be implemented with high priority, without considering any other projects (with low priority) within their implementation. In that case, the projects that have to be scheduled are separated into groups based on their priority score and initially the first (high priority) group is scheduled. Then, this solution is considered as the previous solution and the second group of (low priority) projects is now solved by the ILP, considering the solution of the first group as the unavailability of the teams. This process is repeated for all groups until the final scheduling plan is obtained, that consist of all planned projects.

VI. PERFORMANCE EVALUATION

For solving the ILP-based formulations, the Gurobi library is used [44] on a PC with Core $i5-8400$ CPU @2.80 GHz and 16 GB RAM. It is important to note that the related input in this section is based on realistic data provided by a large organization. The results presented below are performed based on these data to provide a proof of concept and also provide some illustrative examples for the better understanding of the features taken into account of our approach. The related input is presented in the corresponding Tables of the current section. The same results were also observed when performing simulations with similar characteristics. The scheduling technique developed takes the required inputs and provides an optimized assignment and schedule for a period of time ahead. Specifically, the inputs to proposed ILP are as follows: (i) all projects to be scheduled including the required deadline for each project, (ii) the current status of projects in progress, (iii) the teams' capacities, (iv) personnel skills, (v) tasks/projects at hand, (vi) interdependencies between teams, (vii) work efficiency percentage parameter (the percentage of time for a staff member dedicated to a project), (viii) overlapping parameter α (i.e., the completion percentage of a task prior to start overlapping with other interdependent tasks), (ix) start and end dates of the schedule to be created, as well as (x) time limit parameter (optional - i.e., the maximum time that the solver can run in order to find a solution to the problem (per group)); if the solver cannot find an optimal solution within this time, then the best solution found by that time is returned.

A. SIMPLE SCHEDULING SCENARIOS

Given the problem's inputs as previously described, the ILP provides a schedule as the output, specifying the time slots for the implementation of all projects, as well as the team that is responsible for the implementation of each task. Figure 4 shows a simple example for scheduling the 7 projects presented in Table 7 comprising of 14 tasks, when the available teams of the organization are the ones shown in Table 8. For example, as shown in Table 7, to schedule project p_1, three tasks must be performed that require 10, 4, and 7 person-weeks by teams f_1, f_2, and f_3, respectively. Further, the number of persons in each team is shown in Table 8, along with the team priority. Regarding team priority, if for example teams f_1 and f_2 must both work on a project, team f_2 must finish at most α of its task in order for f_1 to start working on its own designated task. Also, team f_6 (that does not have a priority value) can be implemented without any task interdependencies.

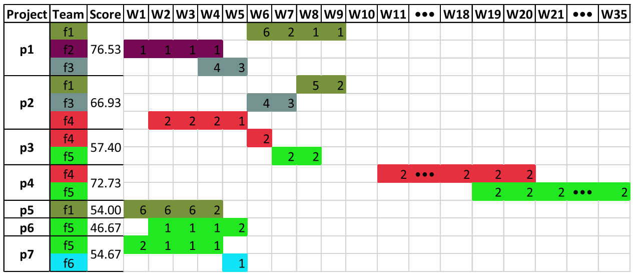
As illustrated in Fig. 4, a total of 35 weeks are required to schedule all projects, while the ILP required 320 seconds to solve the problem. The reader should note that the presented

TABLE 7. Scenario with 7 projects to be scheduled.

Project Id	Tasks required per project (Team - FTEs)		
p_1	$f_1 - 10$	$f_2 - 4$	$f_3 - 7$
p_2	$f_1 - 7$	$f_3 - 7$	$f_4 - 7$
p_3	$f_4 - 2$	$f_5 - 4$	–
p_4	$f_4 - 20$	$f_5 - 34$	–
p_5	$f_1 - 20$	–	–
p_6	$f_5 - 5$	–	–
p_7	$f_5 - 5$	$f_6 - 1$	–

TABLE 8. Teams working on the 7-project scenario of Table 7.

Team Id	Number of persons in team ($\|TM_f\|$)	Priority
f_1	6	5
f_2	1	3
f_3	4	4
f_4	2	1
f_5	2	2
f_6	2	–



Project	Team	Score	W1	W2	W3	W4	W5	W6	W7	W8	W9	W10	W11	•••	W18	W19	W20	W21	•••	W35
p1	f1	76.53						6	2	1	1									
	f2		1	1	1	1														
	f3					4	3													
p2	f1	66.93								5	2									
	f3							4	3											
	f4			2	2	2	1													
p3	f4	57.40						2												
	f5								2	2										
p4	f4	72.73											2	•••	2	2	2			
	f5															2	2	2	•••	2
p5	f1	54.00	6	6	6	2														
p6	f5	46.67		1	1	1	2													
p7	f5	54.67	2	1	1	1														
	f6						1													

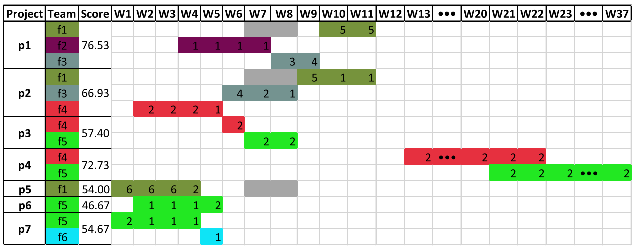
FIGURE 4. Scheduling plan of 7 projects comprising of 14 tasks.

schedule follows all the organization's rules, as presented in Section III. Further, it is noted that these 6 teams and these 7 projects along with the scheduling solution provided in Fig. 4, serve as a basis for the rest of the planning scenarios presented in this section.

In addition, Fig. 5 presents the results for the same scheduling scenario, but for the case where team f_1 is unavailable at time slots $7 - 8$ (illustrated as gray empty cells in the scheduling solution). Due to this unavailability period, it is shown that in the final schedule the required number of weeks to implement all projects now increases from 35 to 37, with the ILP now requiring 190 seconds to solve the problem. The reader should note that the solution does not just shift the tasks/projects after the team unavailability period for the projects p_1 and p_2 that team f_1 must work on after weeks W_7 and W_8, as such an approach would create gaps between the tasks of these projects. Rather, since it is also considered in the ILP's objective, it provides a solution in which tasks are scheduled in such a way that gaps between tasks are minimized.

B. BUILDING ON PREVIOUS SCHEDULING SOLUTIONS

As previously mentioned, the proposed approach also works for dynamic practical scenarios, since it can take the proposed schedule as input and incorporate additional projects to the already planned schedule. In addition, some already planned projects can be rescheduled. To evaluate this functionality, the following scenario with 9 projects is applied: assume that in the scheduling solution of Fig. 4, two weeks have passed and the organization has followed the implementation according to plan. Now, the first 7 projects of the previous scenario must remain unchanged, and 2 additional projects must be scheduled, that are shown in Table 9.



Project	Team	Score	W1	W2	W3	W4	W5	W6	W7	W8	W9	W10	W11	W12	W13	•••	W20	W21	W22	W23	•••	W37
p1	f1	76.53										5	5									
	f2					1	1	1	1													
	f3									3	4											
p2	f1	66.93									5	1	1									
	f3							4	2	1												
	f4			2	2	2	1															
p3	f4	57.40						2														
	f5								2	2												
p4	f4	72.73													2	•••	2	2	2			
	f5																	2	2	2	•••	2
p5	f1	54.00	6	6	6	2																
p6	f5	46.67		1	1	1	2															
p7	f5	54.67	2	1	1	1																
	f6						1															

FIGURE 5. Scheduling plan of 7 projects, with team f_1 unavailable during weeks $7 - 8$.

TABLE 9. Two additional projects to be scheduled in the 7-project scenario.

Project Id	Tasks required per project (Team - FTEs)	
p_8	$f_2 - 2$	$f_4 - 4$
p_9	$f_4 - 7$	$f_5 - 10$

The new scheduling plan is shown in Fig. 6. It is noted that as 2 weeks have passed from the previous scheduling plan, week W_1 of the solution presented in Fig. 6 corresponds to week W_3 of the scheduling plan presented in Fig. 4. In this scenario, the ILP required 13 seconds to provide a solution, with the new scheduling plan requiring a total of 42 weeks.

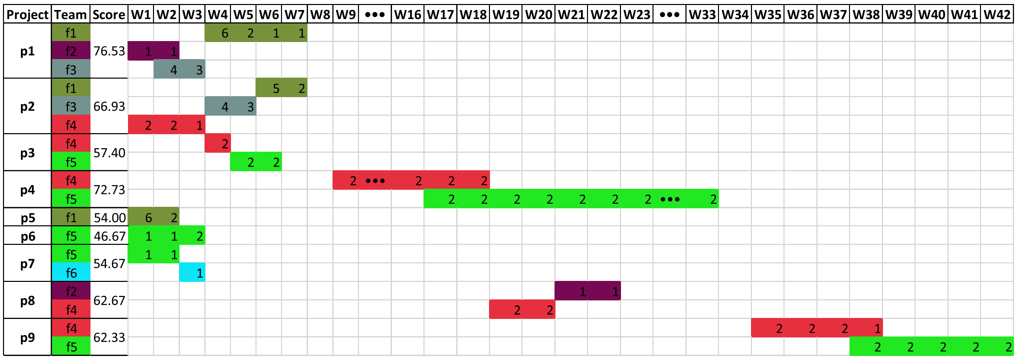
C. MANUAL SCHEDULING CHANGES

Considering again the scheduling plan of Fig. 4, in this scenario the organization decides to reschedule manually two of the projects, as seen in Fig. 7.

This change provides a scheduling plan that now violates the constraints of the formulation, since more employees than the team's available persons are required to work on a task during weeks $3 - 5$ [Constraint (2) violation], as shown in Fig. 7. In addition, during week 4, team f_5 works on three different projects [Constraint (23) violation]. Taking into account the modified schedule, the two new projects presented in Table 9 must be scheduled. Figure 8 illustrates the scheduling plan when using the proposed incremental approach so as to bypass the constraint violations. The ILP solved the problem in 13 seconds, and a total of 44 weeks are now required for the scheduling plan.

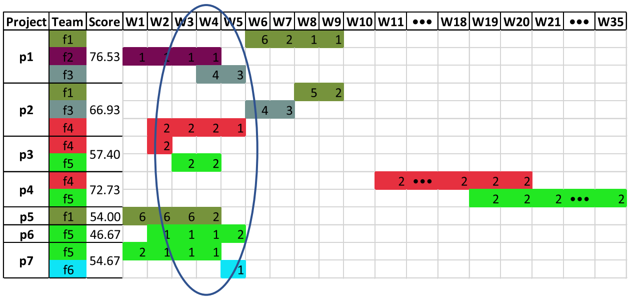
D. PROJECT STATUS

To account for the projects currently implemented (i.e., active projects) during a new scheduling process, the initial set of



Project	Team	Score	W1	W2	W3	W4	W5	W6	W7	W8	W9	•••	W16	W17	W18	W19	W20	W21	W22	W23	•••	W33	W34	W35	W36	W37	W38	W39	W40	W41	W42
p1	f1					6	2	1	1																						
	f2	76.53	1	1																											
	f3			4	3																										
p2	f1							5	2																						
	f3	66.93				4	3																								
	f4		2	2	1																										
p3	f4	57.40				2																									
	f5						2	2																							
p4	f4	72.73									2	•••	2	2	2																
	f5													2	2	2	2	2	2	2	•••	2									
p5	f1	54.00	6	2																											
p6	f5	46.67	1	1	2																										
p7	f5	54.67	1	1																											
	f6				1																										
p8	f2	62.67																1	1												
	f4															2	2														
p9	f4	62.33																						2	2	2	1				
	f5																										2	2	2	2	2

FIGURE 6. Scheduling plan of 9 projects, building on the 7-project scenario.



Project	Team	Score	W1	W2	W3	W4	W5	W6	W7	W8	W9	W10	W11	•••	W18	W19	W20	W21	•••	W35
p1	f1							6	2	1	1									
	f2	76.53	1	1	1	1														
	f3					4	3													
p2	f1									5	2									
	f3	66.93						4	3											
	f4			2	2	2	1													
p3	f4	57.40		2																
	f5				2	2														
p4	f4	72.73											2	•••	2	2	2			
	f5															2	2	2	•••	2
p5	f1	54.00	6	6	6	2														
p6	f5	46.67		1	1	1	2													
p7	f5	54.67	2	1	1	1														
	f6						1													

FIGURE 7. Manual change of the scheduling plan for the 7-project scenario.

7 projects presented in Table 7 is used, but in this scenario the organization sets the status of project p_4 as active and also signifies that 20% of the task performed by team f_4 has been already completed. As can be seen in the new scheduling plan obtained (Fig. 9), since project p_4 is considered active, it is planned to start on the first week of the scheduling plan. To solve the problem and obtain the optimal solution, the ILP required 125 seconds.

E. PROJECT STATUS CONSIDERING DEADLINES

Finally, to account for deadlines in a scheduling plan, the projects presented in Table 7 must be scheduled while in this scenario the organization has set a deadline (week W_8) for project p_7. Also, project p_4 is again considered as active with 20% of the task performed by team f_4 already implemented, while the organization has also provided a deadline for this project (week W_{29}). The results for this scenario can be seen in Fig. 10, where project p_7 finishes at week W_8 instead of week W_5. Also, compared to Fig. 9, project p_4 finishes at week W_{29} instead of week W_{23}. These results are a direct consequence of Constraints (28) - (29), since the objective of the ILP is not just to minimize the ending week of the scheduling plan, but to schedule the projects as close to their deadlines as possible. Clearly, this approach will also have an effect on the other projects that must be scheduled, as can be seen by comparing project p_3 in Figs. 9 and 10. The ILP required 85 seconds to address this scenario.

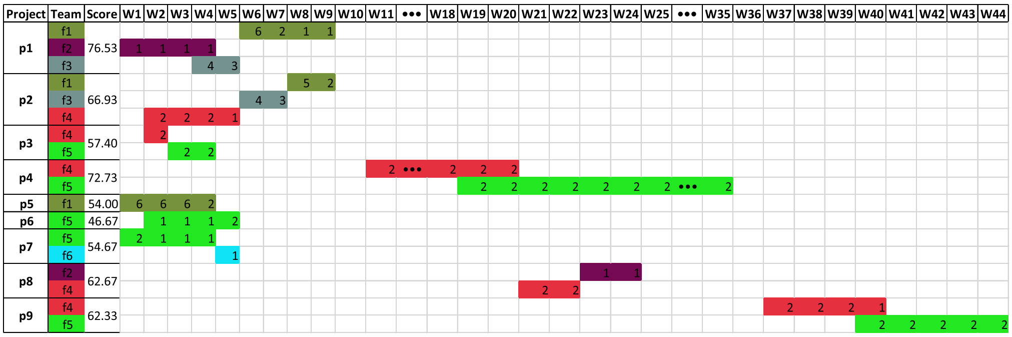
F. EXECUTION TIME - GROUPS

This section presents the performance of the technique that splits the projects into groups (i.e., sequential approach) to evaluate the trade-off between optimality and computational complexity. Three different scenarios are considered, with 10, 20, and 30 projects. Table 10 summarizes the inputs provided for these simulations regarding also the number of tasks, overall number of FTEs required (when the FTEs of each task are summed), and the number of different teams required to implement the projects.

TABLE 10. Summarized inputs for 10, 20, and 30 projects.

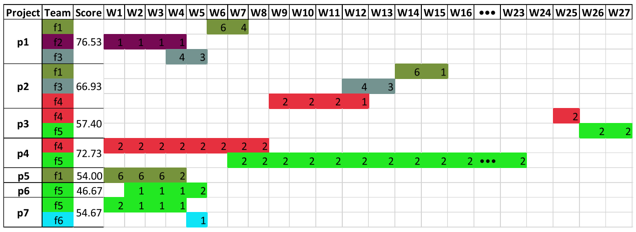
Scenario Id	Number of Projects	Number of Tasks	Overall FTEs	Number of Teams
1	10	19	43	6
2	20	35	82	9
3	30	48	138	10

As for these scenarios there are not any factors affecting the objective, such as active projects, team unavailabilities, or deadlines, the quality of the solution (in terms of scheduling) will be evaluated based on the ending time and score of each project (i.e., $\sum_p c_p \cdot S_p^e$, denoted as the *scheduling score*). Further, the running time of the algorithm is considered as a performance metric, as well as the planning horizon [last time slot (i.e., week) to schedule all projects]. Therefore, only the first and second terms of the objective are taken into account (i.e., the other terms are not considered) when comparing the group approach with the ILP. Specifically, using the first term, the aim is to minimize the score and the planning horizon of the scheduling solution, while using the second term (with its weight set to dominate over the first term), any solution aims to schedule the projects without gaps between the tasks of each project. The optimal solution (in terms of score) is the output of the algorithm when considering only one group (i.e., the problem is solved while considering all projects at the same time). The results shown in Table 11 below present how different group decompositions perform in terms of the objective function and the required execution time. It is noted that these results are performed based on one



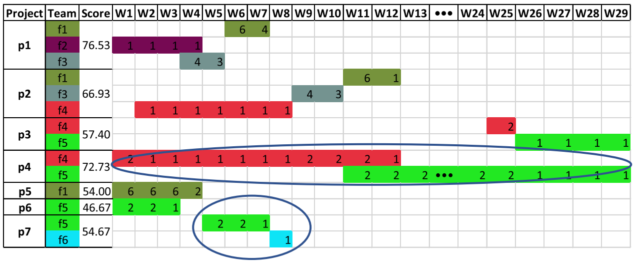
Project	Team	Score	W1	W2	W3	W4	W5	W6	W7	W8	W9	W10	W11	•••	W18	W19	W20	W21	W22	W23	W24	W25	•••	W35	W36	W37	W38	W39	W40	W41	W42	W43	W44
p1	f1	76.53						6	2	1	1																						
	f2		1	1	1	1																											
	f3					4	3																										
p2	f1	66.93								5	2																						
	f3							4	3																								
	f4			2	2	2	1																										
p3	f4	57.40		2																													
	f5				2	2																											
p4	f4	72.73											2	•••	2	2	2																
	f5															2	2	2	2	2	2	2	•••	2									
p5	f1	54.00	6	6	6	2																											
p6	f5	46.67		1	1	1	2																										
p7	f5	54.67	2	1	1	1																											
	f6						1																										
p8	f2	62.67																		1	1												
	f4																	2	2														
p9	f4	62.33																									2	2	2	1				
	f5																													2	2	2	2	2

FIGURE 8. Manual change of the scheduling plan with 2 additional projects (7-project scenario).



Project	Team	Score	W1	W2	W3	W4	W5	W6	W7	W8	W9	W10	W11	W12	W13	W14	W15	W16	•••	W23	W24	W25	W26	W27
p1	f1	76.53						6	4															
	f2		1	1	1	1																		
	f3					4	3																	
p2	f1	66.93														6	1							
	f3													4	3									
	f4										2	2	2	1										
p3	f4	57.40																				2		
	f5																						2	2
p4	f4	72.73	2	2	2	2	2	2	2	2														
	f5								2	2	2	2	2	2	2	2	2	2	•••	2				
p5	f1	54.00	6	6	6	2																		
p6	f5	46.67		1	1	1	2																	
p7	f5	54.67	2	1	1	1																		
	f6						1																	

FIGURE 9. Scheduling plan considering project p_4 as active (7-project scenario).



Project	Team	Score	W1	W2	W3	W4	W5	W6	W7	W8	W9	W10	W11	W12	W13	•••	W24	W25	W26	W27	W28	W29
p1	f1	76.53						6	4													
	f2		1	1	1	1																
	f3					4	3															
p2	f1	66.93											6	1								
	f3										4	3										
	f4			1	1	1	1	1	1	1												
p3	f4	57.40																2				
	f5																		1	1	1	1
p4	f4	72.73	2	1	1	1	1	1	1	1	2	2	2	1								
	f5												2	2	2	•••	2	2	1	1	1	1
p5	f1	54.00	6	6	6	2																
p6	f5	46.67	2	2	1																	
p7	f5	54.67					2	2	1													
	f6								1													

FIGURE 10. Scheduling projects with deadlines for projects p_4 and p_7 (7-project scenario).

instance of realistic data of the organization. The same results were also observed when performing simulations with similar characteristics and group decompositions.

From the results, it is clear that there is a trade-off between the running time and the optimality of the solution. For example, in the case of 10 projects the planning horizon is 6 weeks with a score of 1581 and it requires 10 sec to obtain the optimal solution. On the other hand, considering 2 groups for the same problem, the planning horizon is now equal to 9 weeks with a score of 2020. This solution can be achieved in only 3.97 seconds.

For the case of 20 groups, and when using only one group, the execution time is equal to 495 sec with a score 4324 and planning horizon equal to 13 weeks. On the other hand, considering 2 groups for the same problem, the execution time is reduced to 15.67 sec, while the planning horizon remains the same and the score increases to 5063. In addition, considering 3 groups, the execution time is further decreased but the score and the planning horizon are increased. A similar trend is followed when considering the case of 30 projects for 1, 2, and 3 groups, as can be seen in Table 11, where the execution time is reduced from 22.298 to 35 and to 12 sec, respectively. The planning horizon and the score however, with 2 and 3 groups are increased.

Clearly, when considering several groups, much lower running times can be achieved at the expense of sub-optimal solutions. Therefore, the organization can choose between running time and optimality, based on the available time to produce a scheduling plan. In either case, an estimation on the scheduling plan can be produced in small time scales when considering the concept of groups. It is also clear that the running time of the algorithm in order to find the optimal solution (considering only one group) is increased exponentially as the input size increases.

TABLE 11. Performance metrics: Groups of projects vs optimal solution.

Projects	Groups	Execution time (sec.)	Score	Planning horizon
10	1	10.05	1581	6
	2	3.97	2020	9
20	1	495.65	4324	13
	2	15.67	5063	13
	3	6.79	5960	15
30	1	22, 289.41	7002	14
	2	35.7	8291	15
	3	11.91	8896	17

VII. CONCLUSION

Project scheduling techniques are developed in this work to achieve on-time project delivery. An ILP is initially developed to provide an optimal solution for the problem, as well as other techniques to address scalability issues, incorporate any dynamic changes, and provide flexibility when performing project scheduling. Performance results utilizing real-world data demonstrate the feasibility of the proposed techniques and their capabilities to address a number of different scenarios in terms of input size, project requirements, and respond to any unforeseen changes and uncertainties by rescheduling projects without affecting projects under implementation.

Clearly, based on the aforementioned results, the utilization of the developed ILP formulations can provide significant gains in terms of operational expenditures (OPEX) and revenue stream that an organization incurs by potentially utilizing less personnel for specific tasks/projects, by not allocating personnel for computing these solutions in a non-automated manner, as well as by achieving earlier project completion times.

Future work considers the development of heuristic and metaheuristic algorithms to address scalability considerations, that will be compared to the optimal solution obtained by the ILP. In addition, the model will be updated to consider finer personnel characteristics (i.e., scheduling at the skills rather than at the teams level).

ACKNOWLEDGMENT

(Konstantinos Manousakis and Giannis Savva are co-first authors.) The authors would like to thank Philippos Isaia from the KIOS Research and Innovation Center of Excellence and Christos Siskos and Savvas Charalambides from Cyta for their assistance towards the testing of the proposed techniques. The work of Michalis Mavrovouniotis was done while he was at the KIOS Research and Innovation Center of Excellence, University of Cyprus.

REFERENCES

[1] P. Brucker, A. Drexl, R. Möhring, K. Neumann, and E. Pesch, "Resource-constrained project scheduling: Notation, classification, models, and methods," *Eur. J. Oper. Res.*, vol. 112, no. 1, pp. 3–41, Jan. 1999.

[2] 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.

[3] S. Hartmann and D. Briskorn, "A survey of variants and extensions of the resource-constrained project scheduling problem," *Eur. J. Oper. Res.*, vol. 207, no. 1, pp. 1–14, Nov. 2010.

[4] S. Hartmann and D. Briskorn, "An updated survey of variants and extensions of the resource-constrained project scheduling problem," *Eur. J. Oper. Res.*, vol. 297, no. 1, pp. 1–14, Feb. 2022.

[5] M. G. Sánchez, E. Lalla-Ruiz, A. F. Gil, C. Castro, and S. Voß, "Resource-constrained multi-project scheduling problem: A survey," *Eur. J. Oper. Res.*, vol. 309, no. 3, pp. 958–976, Sep. 2023.

[6] A. Allahverdi, C. T. Ng, T. C. E. Cheng, and M. Y. Kovalyov, "A survey of scheduling problems with setup times or costs," *Eur. J. Oper. Res.*, vol. 187, no. 3, pp. 985–1032, Jun. 2008.

[7] Y. Tian, T. Xiong, Z. Liu, Y. Mei, and L. Wan, "Multi-objective multi-skill resource-constrained project scheduling problem with skill switches: Model and evolutionary approaches," *Comput. Ind. Eng.*, vol. 167, May 2022, Art. no. 107897.

[8] H. D. Ardakani and A. Dehghani, "Multi-objective optimization of multi-mode resource-constrained project selection and scheduling problem considering resource leveling and time-varying resource usage," *Int. J. Supply Oper. Manage.*, vol. 9, no. 1, pp. 34–55, Feb. 2022.

[9] R. van Eynde and M. Vanhoucke, "Resource-constrained multi-project scheduling: Benchmark datasets and decoupled scheduling," *J. Scheduling*, vol. 23, no. 3, pp. 301–325, Jun. 2020.

[10] J. Chen, P. Han, Y. Zhang, T. You, and P. Zheng, "Scheduling energy consumption-constrained workflows in heterogeneous multi-processor embedded systems," *J. Syst. Archit.*, vol. 142, Sep. 2023, Art. no. 102938.

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

[12] S. Ben Issa, R. A. Patterson, and Y. Tu, "Solving resource-constrained multi-project environment under different activity assumptions," *Int. J. Prod. Econ.*, vol. 232, Feb. 2021, Art. no. 107936.

[13] A. Moukrim, A. Quilliot, and H. Toussaint, "An effective branch-and-price algorithm for the preemptive resource constrained project scheduling problem based on minimal interval order enumeration," *Eur. J. Oper. Res.*, vol. 244, no. 2, pp. 360–368, Jul. 2015.

[14] Y. Shou, Y. Li, and C. Lai, "Hybrid particle swarm optimization for preemptive resource-constrained project scheduling," *Neurocomputing*, vol. 148, pp. 122–128, Jan. 2015.

[15] M. Vanhoucke and J. Coelho, "Resource-constrained project scheduling with activity splitting and setup times," *Comput. Oper. Res.*, vol. 109, pp. 230–249, Sep. 2019.

[16] L. Cui, X. Liu, S. Lu, and Z. Jia, "A variable neighborhood search approach for the resource-constrained multi-project collaborative scheduling problem," *Appl. Soft Comput.*, vol. 107, Aug. 2021, Art. no. 107480.

[17] J. Lin, L. Zhu, and K. Gao, "A genetic programming hyper-heuristic approach for the multi-skill resource constrained project scheduling problem," *Expert Syst. Appl.*, vol. 140, Feb. 2020, Art. no. 112915.

[18] 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.

[19] A. Ngo-The and G. Ruhe, "Optimized resource allocation for software release planning," *IEEE Trans. Softw. Eng.*, vol. 35, no. 1, pp. 109–123, Jan. 2009.

[20] X. Wang, Q. Chen, N. Mao, X. Chen, and Z. Li, "Proactive approach for stochastic RCMPSP based on multi-priority rule combinations," *Int. J. Prod. Res.*, vol. 53, no. 4, pp. 1098–1110, Feb. 2015.

[21] 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.

[22] H. Moradi and S. Shadrokh, "A robust scheduling for the multi-mode project scheduling problem with a given deadline under uncertainty of activity duration," *Int. J. Prod. Res.*, vol. 57, no. 10, pp. 3138–3167, May 2019.

[23] F. Kong and D. Dou, "Resource-constrained project scheduling problem under multiple time constraints," *J. Construct. Eng. Manage.*, vol. 147, no. 2, Feb. 2021, Art. no. 04020170.

[24] F. Li, Z. Xu, and H. Li, "A multi-agent based cooperative approach to decentralized multi-project scheduling and resource allocation," *Comput. Ind. Eng.*, vol. 151, Jan. 2021, Art. no. 106961.

[25] J. A. S. Araujo, H. G. Santos, B. Gendron, S. D. Jena, S. S. Brito, and D. S. Souza, "Strong bounds for resource constrained project scheduling: Preprocessing and cutting planes," *Comput. Oper. Res.*, vol. 113, Jan. 2020, Art. no. 104782.

[26] E. N. Afruzi, A. Aghaie, and A. A. Najafi, "Robust optimization for the resource-constrained multi-project scheduling problem with uncertain activity durations," *Scientia Iranica*, vol. 27, no. 1, pp. 361–376, Feb. 2020.

[27] F. Zaman, S. Elsayed, R. Sarker, D. Essam, and C. A. C. Coello, "An evolutionary approach for resource constrained project scheduling with uncertain changes," *Comput. Oper. Res.*, vol. 125, Jan. 2021, Art. no. 105104.

[28] M. H. F. Rahman, R. K. Chakrabortty, and M. J. Ryan, "Managing uncertainty and disruptions in resource constrained project scheduling problems: A real-time reactive approach," *IEEE Access*, vol. 9, pp. 45562–45586, 2021.

[29] Z. Ma, Z. He, N. Wang, Z. Yang, and E. Demeulemeester, "A genetic algorithm for the proactive resource-constrained project scheduling problem with activity splitting," *IEEE Trans. Eng. Manag.*, vol. 66, no. 3, pp. 459–474, Aug. 2019.

[30] J. Xiao, M.-L. Gao, and M.-M. Huang, "Empirical study of multi-objective ant colony optimization to software project scheduling problems," in *Proc. Annu. Conf. Genetic Evol. Comput.*, Jul. 2015, pp. 759–766.

[31] J. K. Lenstra and A. H. G. R. Kan, "Complexity of scheduling under precedence constraints," *Operations Res.*, vol. 26, no. 1, pp. 22–35, Feb. 1978.

[32] B. Crawford, R. Soto, F. Johnson, E. Monfroy, and F. Paredes, "A max–min ant system algorithm to solve the software project scheduling problem," *Expert Syst. Appl.*, vol. 41, no. 15, pp. 6634–6645, Nov. 2014.

[33] H. M. H. Saad, R. K. Chakrabortty, S. Elsayed, and M. J. Ryan, "Quantum-inspired genetic algorithm for resource-constrained project-scheduling," *IEEE Access*, vol. 9, pp. 38488–38502, 2021.

[34] Z. Ma, W. Zheng, Z. He, N. Wang, and X. Hu, "A genetic algorithm for proactive project scheduling with resource transfer times," *Comput. Ind. Eng.*, vol. 174, Dec. 2022, Art. no. 108754.

[35] Y. Liu, R. Li, and H. Liu, "Heuristic optimization for robust resource-constrained flexible project scheduling problem," *IEEE Access*, vol. 8, pp. 142269–142281, 2020.

[36] F. Mahmud, F. Zaman, A. Ahrari, R. Sarker, and D. Essam, "Genetic algorithm for singular resource constrained project scheduling problems," *IEEE Access*, vol. 9, pp. 131767–131779, 2021.
[37] Y. He, Z. He, and N. Wang, "Tabu search and simulated annealing for resource-constrained multi-project scheduling to minimize maximal cash flow gap," *J. Ind. Manage. Optim.*, vol. 17, no. 5, p. 2451, 2021.
[38] J. Chen, T. Li, Y. Zhang, T. You, Y. Lu, P. Tiwari, and N. Kumar, "Global-and-local attention-based reinforcement learning for cooperative behaviour control of multiple UAVs," *IEEE Trans. Veh. Technol.*, early access, 2024, doi: 10.1109/TVT.2023.3327571.
[39] *PrimaVera: The Standard for Planning and Scheduling*. Accessed: Sep. 4, 2023. [Online]. Available: https://www.oracle.com/industries/construction-engineering/primavera-p6/
[40] *Microsoft Project*. Accessed: Sep. 4, 2023. [Online]. Available: https://www.microsoft.com/en-us/microsoft-365/project/project-management-software
[41] *MJC2: Employee Scheduling Software*. Accessed: Sep. 4, 2023. [Online]. Available: https://www.mjc2.com/
[42] *MangoGem APS Optimizer*. Accessed: Sep. 4, 2023. [Online]. Available: https://www.mangogem.com/
[43] K. Manousakis and G. Ellinas, "Crosstalk-aware routing spectrum assignment and WSS placement in flexible grid optical networks," *J. Lightw. Technol.*, vol. 35, no. 9, pp. 1477–1489, Mar. 13, 2017.
[44] *Gurobi Optimizer Reference Manual*. Accessed: Sep. 4, 2023. [Online]. Available: https://www.gurobi.com

KONSTANTINOS MANOUSAKIS (Senior Member, IEEE) received the Diploma, M.Sc., and Ph.D. degrees in computer engineering and informatics from the University of Patras, Greece, in 2004, 2007, and 2011, respectively. He is currently a Research Fellow with the KIOS Research and Innovation Center of Excellence, University of Cyprus. His research work has been published in more than 65 top-tier journals and telecommunications conferences. His research interests include optimization algorithms, heuristics and metaheuristics, resource allocation, protection and restoration techniques, and techno-economic aspects of communication networks.

GIANNIS SAVVA (Member, IEEE) received the B.Sc. and Ph.D. degrees in electrical engineering from the Department of Electrical and Computer Engineering, University of Cyprus, Nicosia, Cyprus, in 2017 and 2022, respectively. He is currently a Research Associate with the KIOS Research and Innovation Center of Excellence, University of Cyprus. His research interests include telecommunications, optimization algorithms, heuristic and metaheuristic approaches, resource allocation techniques, security and protection, network planning, network coding, and physical layer security in optical networks.

NICOS PAPADOURI received the B.Sc. degree in computer science with a minor in biomedical engineering from the University of Cyprus, in 2020. His undergraduate thesis project focused on developing an Android application to provide cognitive assistance for elderly users. Currently, he is a Software Engineer with the KIOS Research and Innovation Center of Excellence, specializing in web and mobile application development and leveraging technologies, such as angular, react native, and android development using Java and Django. With a passion for enhancing user experiences, he strives to bridge the gap between technology and user needs.

MICHALIS MAVROVOUNIOTIS received the B.Sc. degree in computer science from the University of Leicester, U.K., in 2008, the M.Sc. degree in natural computation from the University of Birmingham, U.K., in 2009, and the Ph.D. degree in computer science from the University of Leicester, in 2013. He is currently a Researcher with the Eratosthenes Centre of Excellence, Limassol, Cyprus. He has more than 55 refered publications. His research interests include evolutionary computation, swarm intelligence, memetic computing, combinatorial optimization problems, artificial intelligence in dynamic and uncertain environments, and relevant real-world applications. He is the Chair of the IEEE Task Force on Evolutionary Computation in Dynamic and Uncertain Environments, under the Evolutionary Computation Technical Committee of the IEEE Computational Intelligence Society.

ATHANASIOS CHRISTOFIDES received the Diploma degree in electrical engineering from the National Technical University of Athens, Greece, the M.Sc. degree in electrical engineering from The University of Texas at Austin, the Diploma (master's) degree in management from the Mediterranean Institute of Management, Cyprus. Furthermore, he was a Ph.D. candidate with the Department of Electrical and Computer Engineering, University of Cyprus, from 2006 to 2014. From 2006 to 2014, he was a member of the KIOS Research and Innovation Center, carrying out research on the application of complex networks theory in transparent mesh optical networks. He has been with Cyta, the incumbent telecommunications operator in Cyprus, for the last 26 years, serving in various positions in networks, strategy, and IT. During the last eight years, he has been a member of the IT Planning and Architecture Team, Cyta.

NEDI KOLOKOTRONI received the B.Sc. degree in computer science from Iowa State University, in 1986, and the M.B.A. degree from the University of New Haven, in 2000. She has been with Cyta, the telecom incumbent operator in Cyprus, for the last 33 years, where she is currently the Head of the IT Business and Operations Support Systems (BSS/OSS). She also served as the Head of the IT Planning and Architecture, for the previous eight years.

GEORGIOS ELLINAS (Senior Member, IEEE) received the B.S., M.Sc., M.Phil., and Ph.D. degrees in electrical engineering from Columbia University. He was the Past Chair of the Department of Electrical and Computer Engineering, from 2014 to 2020. He is currently a Professor with the Department of Electrical and Computer Engineering and a Founding Member of the KIOS Research and Innovation Center of Excellence, University of Cyprus. Previously, he served as an Associate Professor of electrical engineering with The City College of New York, the Senior Network Architect with Tellium Inc., and a Research Scientist with Bell Communications Research (Bellcore). His research interests include telecommunication networks, intelligent transportation systems, the IoT, and unmanned aerial systems.