A Hybrid Heuristic for a Two-Agent Multi-Skill Resource-Constrained Scheduling Problem

Meya Haroune[1]
Université de Tours
Laboratoire d'Informatique
Fondamentale et Appliquée
de Tours (LIFAT) ROOT ERL-CNRS 7002
Université de Nouakchott Al-Aasriya

Cheikh Dhib[2]
Institut Supérieur du Numérique
Nouakchott, Mauritanie

Emmanuel Néron[3]
Université de Tours
Laboratoire d'Informatique
Fondamentale et Appliquée
de Tours (LIFAT) ROOT ERL-CNRS 7002
France, Tours

Ameur Soukhal[4]
Université de Tours
Laboratoire d'Informatique
Fondamentale et Appliquée
de Tours (LIFAT) ROOT ERL-CNRS 7002
France, Tours

Hafed Mohamed Babou[5]
École Supérieure de Polytechnique
Unité de Recherche Intelligence Artificielle
Nouakchott, Mauritanie

Farouk Mohamedade Nanne[6]
Université de Nouakchott Al-Aasriya
Unité de Recherche Calcul Scientifique,
Informatique et Science de Données
Nouakchott, Mauritanie

***Abstract*—This paper addresses an industrial case of the two-agent scheduling problem with a global objective function. Each agent manages one or several projects and competes with another agent for the use of common multi-skilled employees. There is a pool of employees, each of which can perform a set of skills with heterogeneous performance levels. The objectives of the two agents are both to minimize the total weighted tardiness of its tasks. Furthermore, We assume that some constraints (soft constraints) can be violated when there is no feasible schedule for the problem. Thus, the global objective function minimizes the constraint violations by reducing the undesirable deviations in the soft constraints from their respective goals. The overall objective is to find a schedule that minimizes both agents objective functions (local objectives) and the global objective function. We provide a mixed-integer goal programming (MIGP) formulation for the problem. In addition, we present a hybrid algorithm combining an exact procedure, a greedy heuristic, and a genetic algorithm to find an approximate Pareto solution set. We compare the performance of the hybrid algorithm against the corresponding MIGP formulation with simulated instances derived from real-world instances.**

***Keywords*—Two agents; multi-skilled employees; multi-project scheduling; hybrid genetic algorithm; MIGP**

I. Introduction

In the last decade, multi-project scheduling studies have introduced a new environment where different agents (local decision makers) are involved in the scheduling process. This is very common in some real-life situations, where tasks to be processed belong to different subsets and are subject to different performance measures. Such a situation can be encountered in an organization that deals with multiple projects for which the customers do not have the same requirements. For instance, some customers may be more demanding on delay, some on cost, and others on both at the same time, etc. To deal with these different requirements, new extensions of multi-project scheduling problems have been introduced, in which at least one performance measure is applied on some tasks and not on the whole set. In addition, subsets of tasks compete for the use of common processing resources, which can create conflicts. These kinds of problems are called multi-agent scheduling problems [1].

The authors in [2] and [1] were pioneers who introduce the multi-agent concept into scheduling problems. Particularly in the two-agent scheduling model, two agents want to perform their respective tasks on common processing resources. Each agent has its own subset of tasks, which is entirely distinct from the subset of the other agent, and wants to optimize some scheduling criterion that depends on its tasks only. The goal is to determine the best compromise solutions that satisfy the agents' criteria.

This paper studies, to our knowledge for the first time, a model integrating the concept of the two-agent scheduling and multi-skill project scheduling. The two agents compete on the usage of common multi-skilled employees. Each agent manages one or more software projects that must be carried out simultaneously and completed within a fixed horizon (consecutive weeks). Each project consists of a set of independent, preemptive tasks; and each task belongs to one of the agents.

Each task is associated with a release date, a due date, and a penalty value must be paid for each week of delay. These release and due dates are negotiated with the final client and are contractually fixed. Thus, the non-respect of one of these due dates may lead to the payment of penalties. Our aim is to reduce these penalties. We consider that each task needs exactly one skill and must be performed by one employee who possesses the corresponding skill with an efficiency level. Furthermore, each task has a nominal load which corresponds to a theoretical time needed to perform this task. The nominal load for each task may be compressed according to the efficiency level of the employee in charge of that task.

There is a pool of multi-skilled employees with known weekly availability. Each employee may be involved in more than one project at the same time with a maximum quota (percentage of time) allotted to each project. These quotas are considered here as variables and need to be calculated by the procedure of scheduling.

Furthermore, we consider that some constraints (soft constraints) can be violated when there is no feasible schedule for the problem. The global objective function seeks to minimize these constraint violations by reducing the undesirable deviations in the soft constraints from their respective goals. The objectives of the two agents are both to minimize the total weighted tardiness of its tasks.

The problem under study is a general instance of the uniform parallel machines scheduling problem with preemptions and release dates $Qm \mid r_j, pmtn \mid \sum w_j T_j$, which is known to be $\mathcal{NP}$-hard ([3]). However, we focus here on the case with more than one agent with different objective functions and resources with heterogeneous skills, which obviously increases the difficulty of solving an instance considerably. Our goal in this paper is to design effective heuristics that are able to generate a good approximation of the Pareto set.

The rest of the paper is organized as follows. The next section reviews the relevant literature on the two-agent scheduling problem. Section III describes the addressed problem in more detail. Sections IV and V present a mixed-integer goal programming (MIGP) formulation and heuristic approaches, respectively. Afterward, section VI-B5 presents the results of experiments conducted to analyze the performance of the proposed methods, and then Section VII concludes and presents future works.

II. Literature Review

This section presents the literature related to two scheduling topics that have been addressed so far separately: the multi-skill resource-constrained project scheduling problem (MS-RCPSP) and the multi-agent scheduling problem. In Section II-A, we discuss multi-skill project scheduling studies that focused on the multi-project environment. In Section II-B, we discuss multi-agent scheduling studies that specifically interested in the case of two agents competing on parallel machines. A synthesis of the reviewed literature is given in Section II-C.

A. Multi-Project Multi-Skill Resources-Constrained Project Scheduling Problems

The multi-skill Resources-Constrained Project Scheduling problem (MS-RCPSP) is an extension of the well-known Resource-Constrained Project Scheduling Problem (RCPSP), whereby multi-skilled resources (human resources or multipurpose machines) are involved. This multi-skill RCPSP extension focuses more on the particularities of human resources, such as the skills they master and sometimes the level of effectiveness in exercising those skills.

The author in [4] were the pioneers who introduced multi-skill resources into the project-scheduling field. Since then, many researchers have focused their studies on this problem considering many properties and optimizing various objectives. Particularly, most of the studies focused on MS-RCPSP merely assume that all tasks to be scheduled belong to the same project. In this review, we focus on a multi-project setting, the reader interested in the mono-project case is referred to the papers by [5], [6].

The MS-RCPSP has been considered in a multi-project environment. The author in [7] consider a multi-project setting with heterogeneous skill resources and learning effect. The concept of learning effect means that the efficiency of resources will increase by doing more. The objective function in their study minimizes outsourcing costs. The authors in [8] and [9] considered resources with heterogeneous skills that influence the speed of work of resources. The author in [8] subdivided projects into work packages, with earliest and latest start periods associated with projects. However, [9] considered earliest and latest start periods for tasks. Thus, each task has exactly one predecessor task linked with it by maximum and minimum start-to-start time lags. The objective functions in both studies minimize the costs associated with internal and external resource usage. The author in [10] extended the same model by considering a stochastic setting. The author in [11] considered heterogeneous skills and assumed that the efficiency levels of skills may increase or decrease task duration. The objective is to assign to each project a subset of resources (team), with each team member can be assigned to several projects at a time. The author in [12] focused on a multi-objective version for project selection and scheduling problem, that includes heterogeneous skills, variable capacities over time, learning and forgetting effects. The joint problem of project selection and scheduling consists in selecting then scheduling an optimal portfolio of projects among several available projects. The objectives of the authors are the maximization of the economic gains of the selected projects and the maximization of the efficiency increase of the resources due to learning effects. The author in [13] developed a similar model for multi-project scheduling and multi-skilled staff assignment for IT product development. The objective functions they considered consist of maximizing the efficiency gain and minimizing the product development cycle time and costs. The author in [14] investigated a roughly similar model with uncertainty and learning effect. They assumed that each task requires several skills with a minimum level per skill and its processing time is related to resource efficiency. The book of [15] covers three versions for multi-project scheduling, namely project selection and scheduling, workforce assignment, and resource leveling. In the recent paper of [16], an integrated model of multi-mode and multi-skill project scheduling problem is considered. The multi-skilled resources have different skill levels, resulting in different processing times for the same task. The author focused on the minimization of the total makespan of projects.

B. Two-Agent Scheduling Problems

To better position our work more clearly in the multi-agent scheduling literature, we decide to limit our review to the related literature, which topics may be classified into (1) two-agent single-machine scheduling problems and (2) two-agent scheduling problems in a parallel machine environment.

1) Two-Agent Single-Machine Scheduling Problems: There is an enormous amount of literature investigating two-agent single-machine scheduling problems. Because of the large amount of literature, we only discuss in this section those including due date-based objective functions. Agnetis et al ([1]) studied several scenarios for different combinations of objective functions of the two agents involving a single machine. The problems addressed consist in minimizing the value of one agent, while maintaining the objective value of the other agent below or at a fixed level. The objective functions they considered include the maximum of regular functions (f_{max}), number of late jobs ($\sum U_j$), and total weighted completion times ($\sum w_j C_j$). The author in [17] deal with a similar model with the goal to minimize the total completion time ($\sum C_j$) of one agent with the restriction that the number of tardy jobs ($\sum U_j$) of the other agent cannot exceed a given number. The author in [18] addressed several two-agent single-machine problems consisting in minimizing the total weighted completion time of one agent, subject to an upper bound on the value of the other agent, which may be: total weighted completion time, maximum lateness, and maximum completion time. More recently, [19] considered a similar model with an objective to minimize the weighted number of tardy tasks ($\sum w_j U_j$) of the first agent, subject to an upper bound on the weighted number of tardy jobs of the second agent. The author in [20] proposed a model similar to the models above with the objective to minimize the total weighted late tasks of one agent, while keeping the value of the total completion time of the other agent lower than or equal to a given value. The authors of [21] addressed a two-agent single-machine scheduling problem with learning effects where the objective is to minimize the total tardiness ($\sum T_j$) of the first agent, subject to an upper bound on the maximum tardiness (T_{max}) of the second agent. The author in [22] extended the model of [17] to the case with learning effect. The objective was the minimization of the total weighted completion time of the first agent with the restriction that no tardy job is allowed for the second agent. The author in [23] considered a two-agent single-machine scheduling problem with assignable due dates. The goal is to assign a due date from a given set of due dates and a position in the sequence to each task so that the weighted sum of the objectives of both agents is minimized. The authors considered several combinations of the objectives, which include the maximum lateness, total (weighted) tardiness, and total (weighted) number of tardy tasks. In [24], the author extended the same model by minimizing the objective of the first agent with an upper bound on the value of the objective of the second agent. The author in [25] considered unit processing time tasks and a common due date (see Section II-B2). The author in [26] tackled a two-agent single-machine scheduling model that considers setup times between agent tasks. The authors considered several combinations of the objectives: the maximum lateness, the total (weighted) completion time, and the (weighted) number of tardy tasks. The author in [27] considered the same setting with the objective to minimize the total weighted completion time of the first agent subject to an upper bound on the makespan of the second agent. The author in [28] assumed that tardy a task incurs a tardiness penalty cost which can be avoided by compressing the processing time of some tasks, which includes an additional cost. The objective of each agent is to minimize the total tardiness penalty cost plus the total compression cost. The authors considered two single-machine scheduling problems. The first problem is to minimize the weighted sum of the objectives of the two agents. The second problem is to minimize the objective of one agent with a constraint on the value of the objective of the other agent.

2) Two-Agent Scheduling on Several Machines: Considering the above literature, it can be seen that studies including a parallel-machine environment are relatively limited. More precisely, most of the studies on the two-agent parallel-machine scheduling problem focused on identical parallel machine environment. The author in [29] were the first to consider this setting, where the objective of one agent is to minimize the makespan and that of the other is to minimize the total completion time. The author in [30] studied two models of two-agent scheduling on identical machines where the goal is to minimize the makespan and the total completion time of one agent respectively, subject to an upper bound on the makespan of the other agent. The author in [31] interested in a similar model with the goal to minimize the total weighted completion time of the first agent, subject to an upper bound on the value of the makespan of the second agent. The author in [32] tackled also a similar model with the goal to minimize the makespan of the first agent, subject to an upper bound on the makespan of the second agent. The author in [33] studied several two-agent scheduling problems for identical parallel machines with preemption and release dates for either one set or both sets of tasks. The objective functions they considered are the total (weighted) completion time, the number of tardy tasks, the total tardiness, the maximum lateness, and a regular function of type f_{max}. The author in [34] considered a two-agent setting with a single machine or two identical machines in parallel. The processing times of the tasks of one agent are compressible at an additional cost. The authors considered several different objective functions: the regular function f_{max}, the total completion time plus compression cost, the maximum tardiness plus compression cost, the maximum lateness plus compression cost, and the total compression cost subject to deadline constraints. The author in [35]) studied a two-agent parallel-machine scheduling model with the assumption that a task can be rejected, which incurs a penalty. The objective is to minimize the sum of the scheduling cost of the accepted tasks and the total rejection penalty of the rejected tasks. The authors considered several combinations of objectives: the makespan, the total completion time, the maximum lateness, and the weighted number of tardy tasks. The author in [36] studied a two agent scheduling problem with deteriorating effect on bounded parallel batching machines. The objective is to minimize the makespan of one agent with the constraint that the makespan of the other agent is no more than a given threshold. The author in [37] study a scheduling problem for concurrent jobs on identical parallel machines. It deals with an interfering multi-agent scheduling problem. New complexity results have been developed when the jobs are of identical durations. Some problems are shown to be polynomial where exact solution algorithms are developed and others are shown to be $\mathcal{NP}$-hard.

To the best of our knowledge, very few studies focused on a setting of two agents competing on uniform parallel machines. The author in [38] considered identical processing time tasks where the goal is to minimize at the same time a general cost function associated with the first agent and the makespan of the other agent. In [39], the author addressed the same model with the goal to minimize two maximum functions associated with the two agents. The author in [25] developed a single machine, and parallel (both identical and uniform) machine settings. They discussed the case where the tasks have identical processing times and a common due date. They focused on minimizing the total weighted earliness–tardiness of the first agent, subject to an upper bound on the maximum weighted deviation from the common due date of the tasks of the second agent.

There exist at least two studies that tackled the case of two competing agents on unrelated parallel machines. The author in [40] considered a Just-in-Time setting where the objective of the first agent is to maximize the weighted number of its just-in-time tasks, while the objective of the second agent is either to maximize its maximum gain from its just-in-time jobs or to maximize the weighted number of its just-in-time jobs. The author in [41] focused on the objective of minimizing the total completion time of the tasks of one agent, while keeping the weighted number of tardy tasks of the other agent within a given limit.

C. Synthesis

In this paper, we study an integrated multi-agent scheduling and multi-skill project scheduling problem with many particularities. To the best of our knowledge, this problem has never been studied in the literature. The novelty of our model is related also to several particularities stem from the preferences of the managers. For instance, we consider minimum and maximum loads associated with each task. Also, we consider that each employee, should not exceed a fixed number of different tasks over a given week. We consider a preemptive MS-RCPSP problem with a multi-project setting and heterogeneous skill levels. In light of the existing literature on multi-project multi-skill RCPSP, it is noticed that the studies including those two features are very limited.

Table I presents a synthesis of the studies reviewed above. The first column indicates the paper. The second column provides the characteristics of mutli-skilling: "#SK" indicates the number of skills required by the task, "HS" for heterogeneous skills and "ML" indicates if a minimum level of skill is required to perform the task. The third column indicates some multi-agent features presented according to several sub-columns. Sub-column "M" indicates the number of machines. Sub-column 'E" describes the parallel machine environment ("Pm" for identical machines (Qm" for uniform machines "Rm" for unrelated machines). Sub-column "O" indicates that the objective value of one agent is constrained under an upper bound. Sub-column "C" means that the objective function is a combination of the agent's objectives functions. Sub-column "P" means that the goal is to enumerate the entire Pareto frontier. The fourth column "Objective" shows the objective function. Followed by the last column, which indicates if the paper considers other specific characteristics.

For each paper in Table I, we use the following abbreviations to indicate the objective considered. OC: outsourcing costs; EC: External cost; ATS: average team size; SEG: skill efficiency gain; PDCT: product development cycle time and costs; w_jE : weighted numbers of just-in-time tasks, G: gain from just-in-time tasks; w_jC_j: weighted total completion time; C_{max}: project duration (makespan); f_{max}: maximum of regular functions; L_{max}: maximum lateness; w_jU_j : weighted number of tardy tasks.

III. Problem Definition and Notations

There is a set $K = \{k_1, \dots, k_L\}$ of L projects that must be completed before a common due date (horizon). There are two competing project managers (agents), called A and B, each has a disjoint subset of projects. Each project k_l is broken down into a set of independent, preemptive tasks; and each task belongs to one agent. As explained earlier, we do not consider precedence constraints either between projects or between tasks. As the tasks are independent, we suppose that all the tasks are numbered from 0 to $J + 1$, where the 0th and the $n + 1$th tasks are dummy indicating the start and end of projects, respectively. The subset of tasks of agents A and B are denoted by $N_A = \{n_1, \dots, n_{J_A}\}$ and $N_B = \{n_{J_A+1}, ..., n_J\}$, respectively. The time unit is the half-day.

For each task n_j, there is a nominal load c_j (expressed in man-days), a release date r_j (given in weeks), and a minimum load denoted $c_j^{\min}$ (expressed in half-days) to quantify the minimum degree of realization of this task per week. The minimum load of tasks per week allows modeling some tasks that cannot be interrupted during more than one week. In the case where the employee assigned to a task n_j works on that task during a given week, he or she should perform at least its minimum load $c_j^{\min}$. Furthermore, In each week, the employee assigned to task n_j must not exceed its maximum load $c_j^{\max}$. We want to avoid loss of time due to changing context of employees, which is required when changing from one task to another. Thus, during each week, the number of different tasks on which an employee is working is less than a given value b (fixed for all the projects and all the employees).

Let $E = \{e_1, \dots, e_I\}$ be a set of I multi-skilled employees working in the company. Every employee has an availability per week (a working time known in advance) ranging from 0 to 10 half-days. We refer by $D_{i,s}$ the availability of employee e_i during week h_s, where $h_s \in H$. Employees are allocated to different projects with maximum percentages of time (quotas). As mentioned earlier, these quotas must be determined by the scheduling procedure. Once determined, they must be respected during each week of the horizon. In other words, during each week h_s, any employee e_i assigned to project k_l cannot spend on this project more than $D_{i,s} \times Q_{i,l}$, where $Q_{i,l}$ is the quota of employee e_i on project k_l. Each employee can work on only one task at a given time frame.

In our model, once a task is assigned to an employee with the required skill, it remains so until its accomplishment. The capabilities of performing tasks by resources are represented by a binary skill matrix denoted by m, where $m_{j,i} = 1$ if employee e_i masters task n_j, and $m_{j,i} = 0$ otherwise. It means not every employee can be assigned to a task. Furthermore, we assume that several employees may have different levels of efficiency for the same skill. Since each task requires only one skill, the skill level of the employee is directly associated with the task. The manager estimates the employees' efficiency level according to the standard classification of expertise level: junior, middle and senior. Based on these estimations, we assign an efficiency coefficient equal to $0, 0.5$, and 1 to a junior, middle and senior, respectively. This coefficient is a ratio of an employee's actual processing time to perform the task against the theoretical processing time (nominal load) needed to complete the corresponding task. Thus, to consider the employee's efficiency level in the processing time of task calculation, a simple linear formula is assumed between the task nominal load and the employee assigned to it. We apply this formula to convert the task nominal load (c_j) to duration (processing time, $p_{j,i}$) according to the efficiency level ($v_{j,i}$) of the employee: $p_{j,i} = (2 - v_{j,i})c_j$. Since the nominal load of the task is given in number of days, we multiply it by 2 to convert it into half-days. For example, a task that requires a java developer and 2 days to be performed, it can be done by a senior developer in 2 half-days (i.e. half of the time), and by a junior developer in 4 half-days (the actual time).

The constraints on the minimum load of tasks per week and on the number of different tasks on which an employee is working are soft constraints imposed by the manager to increase the productivity of the employees.

For an effective schedule, these soft constraints should be taken into account. However, the manager allows the soft constraints to be violated when there is no feasible schedule for the problem. The solution approach must minimize these constraint violations by reducing the undesirable deviations in the soft constraints from their respective goals. We introduce a global objective function to penalize these constraint violations. All the other constraints (also called hard constraints) must be respected by the proposed solution.

The objective functions considered here are as follows. Let f^A and f^B be the objective functions of agents A and B, respectively. Each of the agents wants to minimize the total weighted tardiness of its tasks denoted by $f^X = \sum_{j \in N_X} w_j T_j$, where $X \in \{A, B\}$, T_j is the number of weeks of task n_j tardiness and w_j is the penalty cost for this task. Note that if task n_j takes at least one half-day of week ($d_j + 1$) before its completion time, it is late by one week ($T_j = 1$). Furthermore, the soft constraints are addressed as goals to be reached, and the global objective is to get as close as possible to these goals. We consider a global objective function that consists to minimize the violations of these constraints. This objective function is defined by $f^G = \sum_{j=1}^{J} \sum_{s=r_j}^{H} \alpha u_{j,s}^{-} + \sum_{i=1}^{I} \sum_{s=1}^{H} \beta o_{i,s}^{+}$, where $u_{j,s}^{-}$ is the deviation below $c_j^{\min}$, $o_{i,s}^{+}$ is the deviation above b, and α and β are problem parameters stem from the preferences of project managers on soft constraints. Between the two soft constraints, the minimum load constraint is slightly more important than the other soft constraint. Hence, α is slightly higher than β. The problem is to find a schedule that minimizes at the same time the objective functions of both agents as well as the global objective function.

Following the conventional three-field notation introduced by [42] and extended by [43], this problem may be denoted by: $Qm \mid CO-GA, r_j, pmtn \mid f^G, f^A, f^B$, where $CO-GA$ denotes a problem of disjoint subsets competing with a global objective.

A solution consists of two parts: the first is to specify a suitable allocation of employees to projects, and the second is to determine a schedule of tasks for each employee to complete within the planning horizon H. Note that a schedule is defining by a suitable assignment of employees to task and the load (i.e, the number of half-days) that each employee has to perform of each of its tasks during each week. We are interested in determining a good approximation of the Pareto frontier.

TABLE I. SYNTHESIS OF THE REVIEWED LITERATURE

Auteurs	Multi-skill features			Multi-agent features					Objective	Other
	# Sk	HS	ML	# M	E	O	C	P		
This paper	1	●	●	> 1	Q_m		●		$\sum w_jTj$, GF	Multi-project, preemption, minimum and maximum loads, soft constraints
[16]	1								C_{max}	Multi-project, release dates for projects, multi-mode
[19]				1		●			$\sum w_jU_j$	
[14]	≥ 1	●	●						C_{max}, cost	Mutil-project, multi-objective, uncertainty, learning effect
[28]				1		●	●		C	Controllable processing times
[36]				> 1	Pm	●			C_{max}	Deteriorating effect
[10]	≥ 1								EC	Multi-project, sctochastic concept, internal and external resources
[27]				1		●			$\sum w_jC_j, C_{max}$	Setup time
[13]	1								SEG,PDCT	Multi-project, Multi-objective,Learning and forgetting effects
[35])				> 1	Pm	●			$\sum C_j, C_{max}, \sum C_j, L_{max}, \sum w_jU_j$	Task rejection
[31]				> 1	Pm	●			$\sum w_jC_j, C_{max}$	
[26]				1		●			$\sum L_{max}, \sum C_j, \sum w_jC_j, \sum U_j, \sum w_jU_j$	Setup times
[11]	1	●							ATS	Multi-project
[32]				> 1	Pm	●			C_{max}	
[21]				1		●			$T_{max}, \sum T_j$	Learning effects
[34]				< 2	Pm	●			$\sum C_j, f_{max}, L_{max}$	Controllable processing times
[24]				1		●			$\sum T_j, \sum w_jT_j, L_{max}, \sum U_j, \sum w_jT_j, \sum w_j(E_j + T_j)$	Assignable due dates
[25]				1 and > 1	$Qm, Pm, P1$	●			$E_j + T_j, max(E_j + T_j)$	Unit processing time tasks, common due date
[30]				> 1	Pm	●			C_j, C_{max}	
[9]	≥ 1	●							C	Internal and external resources, overtime
[23]				1			●		$\sum T_j, \sum w_jT_j, L_{max}, \sum U_j, \sum w_jT_j$	Assignable due dates
[22]				1		●			$\sum w_jC_j, \sum U_j$	Learning effect
[12]	≥ 1	●	●						EC	Multi-project,Dynamic competencies, time-dependent capacities
[8]	≥ 1	●							C	Internal and external resources, overtime
[33]				> 1	Pm	●			$\sum C_j, \sum w_jC_j, \sum U_j, \sum T_j, L_{max}, f_{max}$	machines,Preemption
[18]				1		●			$\sum w_jC_j, C_{max}, L_{max}$	
[29]				> 1	Pm		●		$C_{max}, \sum C_j$	
[7]	≥ 1	●							OC	Multi-project,Learning effects, external resources
[17]				1		●			$\sum C_j^A, \sum U_j^B$	
[1]				1		●			$\sum U_j, \sum w_jC_j, f_{max}$	
[38]				> 1	Qm		●		C_{max}, f_{max}	Identical processing time
[39]				> 1	Qm		●		f_{max}	Identical processing time
[40]				> 1	R_m			●	w_jE,G	Just-in-time

Table II presents the notation of the problem parameters used throughout this paper.

TABLE II. NOTATIONS AND PARAMETERS OF THE PROBLEM

General data	
H	project planning horizon
X	index of agent X, $X \in A, B$
K	set of projects, $K = \{k_1, \dots, k_L\}$, $\|K\|=L$
E	set of employees, $E = \{e_1, \dots, e_i\}$, $\|E\|=I$
N	set of tasks, $N = \{n_1, \dots, n_J\}$, $\|N\|=J$
N_X	set of agent X's tasks, $\|N_X\| = J_X$ and $J = J_A + J_B$
N_l	set of tasks of project k_l
WL_l^k	nominal load of project k_l in skill
$Z_k^l = \{z_k \| k \in \{1, \dots, K\}\}$	required skills to complete k_l project
Tasks data	
c_j	nominal load of task n_j (measured in man-day)
r_j	release date of task n_j (given in number of weeks)
d_j	due date of task n_j (given in number of weeks)
$c_j^{\max}$	maximum load of task n_j (given in half-day)
$c_j^{\min}$	minimum load of task n_j (given in half-day)
w_j	penalty cost of task n_j per week
F_j	completion date of task n_j (given in number of weeks)
T_j	tardiness of task n_j (given in number of weeks)
E_k	set of employees mastering skill z_k
Employees data	
$D_{i,s}$	availability of employee e_i during week h_s (given in half-days)
b	number of different tasks on which every employee can work during each week
$M_{i,k}$	1 if employee e_i masters skill z_k required by project k_l, and 0 otherwise
Data on tasks and employees	
$v_{j,i}$	employee e_i's efficiency level for task n_j
$p_{j,i}$	processing time of task n_j according to the efficiency level of employee e_i (in half-days)
$m_{j,i}$	equal to 1 if employee e_i masters task n_j, and 0 otherwise
$\overline{v_j}$	average efficiency of employees mastering task n_j

IV. MIXED-INTEGER GOAL PROGRAMMING FORMULATION

A. Variables

1) $x_{j,i}$ –a binary variable equals 1 if employee e_i is assigned to task n_j and equals 0 otherwise.
2) $y_{j,i,s}$ –an integer variable (ranging from 0 to 10) equal to the number of half-days performed of task n_j by employee e_i during week h_s.
3) $z_{j,i,s}$ –a binary variable equal to 1 if $y_{j,i,s}$ is greater than 0, and equal to 0 otherwise.
4) F_j –the completion time of task n_j.
5) T_j –the tardiness of task n_j.
6) $u_{j,s}^+$ –deviation variable above $c_j^{\min}$ associated to task n_j and week h_s.
7) $u_{j,s}^-$ –deviation variable below $c_i^{\min}$ associated to task n_j and week h_s.
8) $o_{i,s}^+$ –deviation variable above b associated to employee e_i and week h_s.
9) $o_{i,s}^-$ –deviation variable below b associated to employee e_i and week h_s.
10) $Q_{i,l}$ –maximum quota of employee e_i on project k_l (percentage of time).

[]

B. Constraints

$$\sum_{i=1}^{I} x_{j,i} = 1, \quad j = 1, \dots, J \tag{1}$$

$$x_{j,i} \le m_{j,i}, \quad j = 1, \dots, J; \quad i =, \dots, I \tag{2}$$

$$\sum_{s=r_j}^{H} y_{j,i,s} = p_{j,i} \cdot x_{j,i} \quad j = 1, \dots, J; \quad i = 1, \dots, I \tag{3}$$

$$\sum_{i=1}^{I} y_{j,i,s} \le \sum_{i=1}^{I} c_j^{\max} \cdot x_{j,i}, \quad j = 1, \dots, J; \quad s = r_j, \dots, H \tag{4}$$

$$\sum_{i=1}^{I} z_{j,i,s} \le \sum_{i=1}^{I} x_{j,i} \quad j = 1, \dots, J; s = r_j, \dots, H \tag{5}$$

$$\sum_{i=1}^{I} 10 \cdot z_{j,i,s} \ge \sum_{i=1}^{I} y_{j,i,s}, \quad j = 1, \dots, J; \quad s = r_j, \dots, H \tag{6}$$

$$\sum_{j=1}^{J} y_{j,i,s} \le D_{i,s}, \quad i = 1, \dots, I; \quad s = 1, \dots, H \tag{7}$$

$$\sum_{j \in N_l}^{J} y_{j,i,s} \le Q_{i,l} \cdot D_{i,s}, i = 1, \dots, I; \quad s = 1, \dots, H; \quad l = 1, \dots, L \tag{8}$$

$$\sum_{i=1}^{I} y_{j,i,s} \ge \sum_{i=1}^{I} c_j^{\min} * z_{j,i,s}, \quad j = 1, \dots, J; \quad s = r_j, \dots, H \tag{9}$$

$$\sum_{j=1}^{J} z_{j,i,s} \le b, \quad i = 1, \dots, I; \quad s = 1, \dots, H \tag{10}$$

$$F_j \ge \sum_{i=1}^{I} s \cdot z_{j,i,s}, \quad j = 1, \dots, J; \quad s = r_j, \dots, H; \tag{11}$$

$$F_j - T_j \le d_j, i \quad F_j, T_j \ge 0 \tag{12}$$

C. Soft Constraints

According to the problem description, constraints (9) and (10) are soft constraints that can be violated when it is not possible to obtain a feasible schedule. Therefore, we incorporate the possibility of relaxing these soft constraints by adding deviation variables in the formulation. Meanwhile, these deviation variables are calculated and added to the objective function as penalties. After adding the deviation variables, the soft constraints in equations (9) and (10)) became, respectively:

$$\sum_{i=1}^{I} y_{j,i,s} - u_{j,s}^+ + u_{j,s}^- = \sum_{i=1}^{I} c_j^{\min} * z_{j,i,s}, \quad j = 1, \dots, J; \quad s = r_j, \dots, H \tag{13}$$

$$\sum_{j=1}^{J} z_{j,i,s} - o_{i,s}^{+} + o_{i,s}^{-} = b, \quad i = 1, \dots, I; \quad s = 1, \dots, H \tag{14}$$

D. Objective Functions

The three different objective functions are listed as follows.

Global Objective: the goal associated to constraint (13) is to avoid as much as possible that, on a given week, the employee assigned to a task perform of this task less than its minimum load. Because of that, only the negative deviation from this goal is minimized in the following equation.

$$\text{Min} \sum_{j=1}^{J} \sum_{s=r_j}^{H} u_{j,s}^{-} \tag{15}$$

The goal associated to constraint (14) is to limit the loss of employee time due to switching between tasks. To this end, no employee should work on more than the maximum number of tasks per week. Therefore, only the positive deviation from this goal is minimized in the following objective function.

$$\text{Min} \sum_{i=1}^{I} \sum_{s=1}^{H} o_{i,s}^{+} \tag{16}$$

After incorporating these goals, the achieving global objective function can be written as follows.

$$\text{Min } \sum_{j=1}^{J} \sum_{s=r_j}^{H} \alpha u_{j,s}^{-} + \sum_{j=1}^{I} \sum_{s=1}^{H} \beta o_{i,s}^{+} \tag{17}$$

Local Objectives the agents' objective functions in equations 18 and 19 seek to minimize the total weighted tardiness of their tasks.

$$\text{Minimize} \sum_{j=1}^{J_A} w_j T_j \tag{18}$$

$$\text{Minimize} \sum_{j=J_A+1}^{J} w_j T_j \tag{19}$$

V. Heuristic Algorithms

Due to its complexity, an exact resolution of the problem is very difficult within a reasonable computation time. Therefore, we propose a hybrid algorithm combining an exact procedure, a greedy heuristic, and a genetic algorithm to find an approximate Pareto solution set. The main steps of this hybrid algorithm are as follows:

- Use a mixed integer-linear program (MILP)to set maximum quotas of employees' time on projects.
- Use a greedy heuristic to determine initial solutions.
- Apply a genetic algorithm of type NSGA-II to determine a good approximation of the Pareto frontier.

In the next sections, we detail each of the steps of the hybrid algorithm.

A. Generating Maximum Quotas

This procedure, denoted by PGQ_{exact}, is used to allocate to each project the set of employees with the necessary skills for its realization. Furthermore, it specifies the working time that each employee must not exceed per week on each of its projects. We use a simplified MILP model, which considers only the constraints on the weekly availability of employees and the workload of projects to be carried out. This MILP model uses time-indexed decision variables.

1) Variables: We define the integer variable $Y_{i,l,s,k}$ (ranging from 0 to 10) equal to the effective working time of employee e_i on project k_l during week h_s on skill z_k. The integer variable $Q_{i,l}$ (ranging from 0 to 10) equal to the maximum quota of employee e_i assigned to the project k_l. The integer variable $d_{i,s}^{-}$ (ranging from 0 to $D_{i,s}$) equal to the unused availability of employee e_i during week h_s.

2) General Formulation: The general formulation is given in the following.

$$\text{Minimize} \sum_{i=1}^{I} \sum_{s=1}^{H} d_{i,s}^{-} \tag{20}$$

s.c.

$$Y_{i,l,s,k} \leq 10 \cdot M_{i,k} \quad i = 1, \dots, I; \quad s = 1, \dots, H; \\ k = 1, \dots, K; \quad = l, \dots, L; \tag{21}$$

$$\sum_{k=1}^{K} Y_{i,l,s,k} \leq D_{i,s} \cdot Q_{i,l}, \quad i = 1, \dots, I; \quad s = 1, \dots, H; \\ l = 1, \dots, L \tag{22}$$

$$\sum_{l=1}^{L} \sum_{k=1}^{K} Y_{i,l,s,k} + d_{i,s}^{-} = D_{i,s}, \quad i = 1, \dots, I; \quad s = 1, \dots, H \tag{23}$$

$$\sum_{i=1}^{I} \sum_{s=1}^{H} Y_{i,l,s,k} \ = WL_l^k, \quad l = 1, \dots, L \tag{24}$$

The objective function (20) minimizes the sum of the unused working time (idle time), by avoiding that employees work, during each week, less than their availability. Constraint (21) guarantees that an employee must not work on a project that he or she does not master any of the skills required by that project. Constraint (22) ensures that, on any given week, no employee must exceed his or her quota on each project. Constraint (23) guarantees that an employee must not exceed his availability each week. Constraint (24) imposes that the nominal load of each project in each skill must be executed until completeness.

B. Greedy Heuristic

This algorithm employs simple priority rules and a simple heuristic to construct good initial solutions. This step is very important because a suitable assignment may help the hybrid algorithm to find an approximate Pareto solution set rapidly and effectively. This greedy heuristic returns all solutions obtained after a computation time-limited to AG_{max}.

Each initial solution is generated through two steps. The first step defines the order in which the tasks will be selected, the next step chooses, for each task, the employee who will be in charge of it among the employees mastering that task. We detail below the three steps of the greedy heuristic.

The procedure of the first step returns the list of tasks ordered in non-decreasing order of the number of employees mastering each task. For example, a task mastered by one employee should be assigned before another task mastered by two employees. This ensures that the most critical employees are not overloaded by other tasks that have multiple assignment options. In the case where two tasks are mastered by the same number of employees, the first task to be assigned is chosen according to the weighted earliest due date (WEDD) priority rule.

The procedure of the second step assigns tasks to employees. To perform this step, we proceed as follows. First, for each task n_j in the order of the list of tasks, we get the list of employees E_j mastering that task and allocated to project k_l (task n_j is part of project k_l). For each employee $e_i \in E_j$, we get $\tau_{j,l}^{tot}$ which corresponds to the total availability of employee e_i on project k_l. Then, the employee with the highest total availability will be assigned to task n_j. We note that before selecting this employee we subtract from his total availability the processing time ($p_{j,i}$) of task n_j, which ensure that he has sufficient availability to perform this task. Otherwise, if the employee does not have the required availability, the second employee in the list will be selected and so on. Finally, we update the availability of the selected employee. This means that the choice of the employee for the next task is influenced by the workload that has already been assigned.

The first and second steps of the GA generate a single solution. The remaining solutions are generated iteratively by performing simple mutation operations on the list of tasks, then repeating the second step. Algorithm (1) describes the procedure of the greedy heuristic.

Algorithm 1 Overview of the Greedy Algorithm

```
1: Inputs: $N$ the set of all tasks ordered in ascending order of the number of employees mastering each task; $\tau^{tot}$ the total availability of employees on projects
2: Output: $\sigma$ the assignment of employees to tasks
3: for each Task $n_j \in N$ do
4:     $k_l$ : the project to which $n_j$ belongs
5:     $E_j$ : the list of employees mastering $n_j$
6:     $BinomiaList = \phi$
7:     for each Employee $e_i \in E_j$ do
8:         Add $(e_i, \tau_{i,l}^{tot})$ to $BinomiaList$
9:     end for
10:    Short $BinomiaList$ in decreasing order of $\tau^{tot}$
11:    $NotAffected = true$ , $count = 0$
12:    while $NotAffected$ do
13:        if $BinomiaList[cout][1] - p_{j,i} \geq 0$ then
14:            Assign employee $BinomiaList[count][0]$ to task $n_j$
15:            $NotAffected = false$
16:        end if
17:        $cout++$
18:    end while
19:    update the availability of the employee
20:    Update list $\sigma$
21: end for
22: return $\sigma$
```

C. Adaptation of the NSGA-II

NSGA-II (for Non-dominated Sorting Genetic Algorithm II) is a genetic algorithm well known as one of the most efficient and popular algorithms for solving multi-criteria optimization problems. NSGA-II method is originally proposed by [44] on the basis of NSGA proposed in [45]. We recall in the following the main ideas of this method. For more details, the reader can refer to [44]. First, individuals (solutions) are classified into a number of dominance ranks at each generation using a fast non-dominated sorting method with low computational complexity. Second, a parameter-independent partitioning method was defined by evaluating the crowding distance of individuals in the same dominance rank. Third, a selection operator was designed based on the values of dominance rank and crowding distance of individuals. Finally, an elitism strategy was used to improve the convergence performance of the algorithm.

Our implementation of NSGA-II is based on the following elements. (i) the coding scheme to represent an individual, (ii) the genetic operators to generate and modify new individuals, (iii) the parameters of the algorithm (i.e. population size P_{max}, number of crossover points P_c and mutation points P_m, termination criterion G_{max}).

1) Coding Scheme: A chromosome (or an individual or a solution) contains a given number of genes and is divided into one or more segments. In our NSGA-II, an individual is represented by an assignment of employees to tasks, a gene corresponds to a task, a segment corresponds to the set of tasks assigned to an employee, and the length of the segment corresponds to the number of tasks in this segment. Thus, a separator (0) is used in the coding scheme of an individual to indicate the beginning and the end of each segment.

In order to understand the coding scheme, let's consider the case of two agents A and B, each of which is in charge of a single project. The projects consist of 8 tasks (n_1,...,n_8) to be scheduled over a two-week horizon. Three employees (e_1, e_2, e_3) can be assigned to different tasks. Each agent wants to minimize the sum of the weighted delays of its tasks. Table III shows for each task, the employee with the skill required to perform it. The gray columns represent the tasks of agent B, while the white columns represent the tasks of agent A.

TABLE III. SUITABLE EMPLOYEE ASSIGNMENTS TO TASKS

	n_1	n_2	n_3	n_4	n_5	n_6	n_7	n_8
c_j	3	5	2	6	4	4	5	3
$m_{j,1}$	1	0	0	1	1	0	0	1
$m_{j,2}$	1	1	1	1	0	0	1	1
$m_{j,3}$	0	0	1	0	1	1	1	1

Fig. 1 shows a solution coding for the previous example. In this solution, tasks n_4, n_5 are assigned to employee e_1; tasks n_1, n_2, n_7 are assigned to employee e_2; and tasks n_3, n_6, n_8 are assigned to employee e_3. The 0 is used to separate two assignments.

2) Initial Solutions: Initial solutions are generated using the greedy heuristic described in Section V-B. Note that the

n_4	n_5	0	n_1	n_2	n_7	0	n_3	n_6	n_8

Fig. 1. Representation of a Solution.

number of generated solutions can be smaller than the initial population size P_{max} (P_{max} is a parameter of the NSGA-II method). In this case, we apply simple mutation operations on the elitist individuals to complete the initial population.

3) Genetic Operators: These operators comprise several key elements: selection, crossover, mutation, and population sorting.

Selection: it chooses among the solutions of the parent population, those that will reproduce and generate the new population (offspring) for the following iteration. To this end, we use the binary tournament method, which is one of the most common selection operators. This method randomly selects a pair of parents from the population and compares their fitness functions. If both parents are on the same front, then we compare their crowding distances, and the one with the largest crowding distance is kept. Otherwise, if the two parents are on different fronts, the solution with the best Pareto front (i.e. the one with the best value of the objective function) is kept. Both solutions of a couple are unique, but a solution can be part of several couples.

Crossover: once the section process is done, the entire selected parents move on to the breeding stage. This is where all the parents recombine in some way to create a new population that will be used in the next genetic step (mutation). The process of combining two parents is what is often called crossover.

We adopt the two crossover operators PBX (position-based crossover) and OBX (order-based crossover) proposed by [46]. The choice of these crossover operators is made based on the study published in [47]. The authors compared the performances of 11 crossover operators with the goal to minimize the total weighted tardiness on a single machine. Their experimental results has shown the efficiency of OBX and PBX crossover operators, compared to other operators, to solve this type of scheduling problem. We detail below the different steps of these two crossover operators. Also, two corresponding examples are shown in Fig. 2 and 3.

- OBX Operator
  - Select randomly several genes (tasks) from a parent ($P1$ for example).
  - Place the selected tasks in the new solution, respecting the exact positions that they occupy in the other parent ($P2$).
  - Delete the tasks that are already selected in the other parent ($P2$) to avoid repeating these tasks in the offspring ($O1$).
  - Insert the remaining tasks into in the offspring, from left to right, in the order that they appear in the parent (P2).
  - Place the remaining tasks in the offspring, from left to right, in the order that they appear in the parent (P2).
  - By changing the roles of the parents, the same procedure can be applied to generate the offspring $O2$.
- PBX operator
  - Select randomly a set of tasks from a parent ($P1$ for example).
  - Place the selected tasks in the offspring ($O1$, respecting their exact positions in the parent ($P1$).
  - Delete the tasks that are already selected in the second parent $P2$. The sequence of remaining tasks in $P2$ contains only those tasks that the offspring ($O1$) needs.
  - Place the remaining tasks in $O1$, from left to right, in the order they appear in the parent $P2$.
  - By changing the parent roles, the same procedure can be applied to generate the second offspring ($O2$).

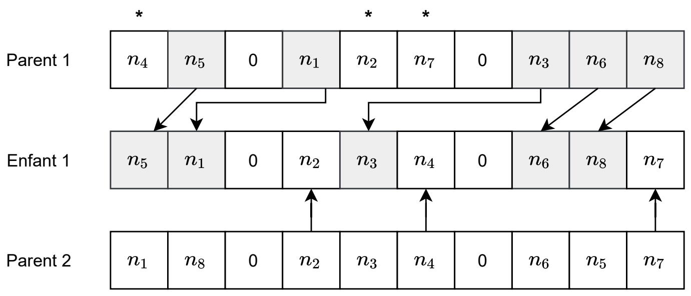
*
*
*
Parent 1
n_4 n_5 0 n_1 n_2 n_7 0 n_3 n_6 n_8
Enfant 1
n_5 n_1 0 n_2 n_3 n_4 0 n_6 n_8 n_7
Parent 2
n_1 n_8 0 n_2 n_3 n_4 0 n_6 n_5 n_7

Fig. 2. OBX Crossover Example

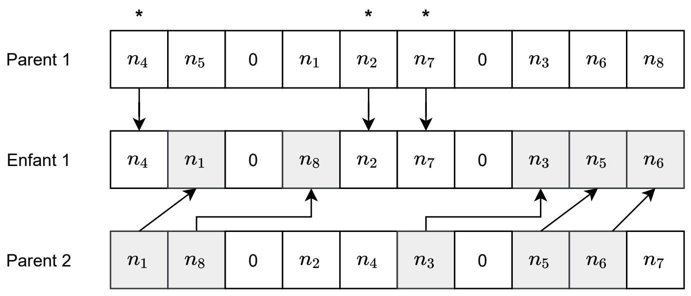
*
*
*
Parent 1
n_4 n_5 0 n_1 n_2 n_7 0 n_3 n_6 n_8
Enfant 1
n_4 n_1 0 n_8 n_2 n_7 0 n_3 n_5 n_6
Parent 2
n_1 n_8 0 n_2 n_4 n_3 0 n_5 n_6 n_7

Fig. 3. PBX Crossover Example

The similarity between parent and offspring populations depends on the number of crossover points. This number (denoted by P_c) represents the number of tasks selected to put into the offspring. Preliminary tests show that a large value of P_c gives a higher probability of building solutions similar to their parents. However, a small value of P_c allows the selection of more distant solutions.

Mutation: We apply this operator at the last stage of the population generation procedure. It maintains the diversity between individuals, and therefore, avoids a premature convergence to a local optimum. The mutation operator is applied to each new solution after the crossover stage. It consists in randomly selecting two employees (e_1 and e_2, for example) who have at least one common skill. Then, each task initially assigned to e_1 and demanding a skill mastered by e_2 will be assigned to the latter. The opposite is also applied to tasks initially assigned to e_2. These two operations are performed P_m times, where P_m (ranging between 2 and 10.) is a parameter of the NSGA-II method.

Population Sorting: This procedure follows the same dominance sorting procedure proposed by [44]. The crowding distance is also applied.

VI. Computational Experiments

This section outlines the characteristics of the instances and the computational results of the proposed methods. Our experiments consist of three parts. In the first part, we seek to determine the size of problem instances that can be solved by the MIGP within a reasonable computation time. In the second part, we want to determine the best parameterization for the NSGA-II method. Finally, in the last part, we evaluate the performance of the heuristic methods.

Our experiments were performed on an Intel® core™ i7-1.9 GHz with 16 GB of RAM under Windows 10. All algorithms were written in Python. The CPLEX 12.8.0 solver associated with Python API was used for solving the MIGP model, using the default parameters except for the time parameter.

A. Instance Characteristics

The methods proposed in this section are experimentally validated on instances derived from real instances. AS mentioned in the introductory section, our study comes from a real scheduling case raised in an IT company. However, to perform our experiments, we could not get enough real data for confidentiality reasons. So, based on the description of the problem and the few real data given by our partner company, we generated 8 sets of instances ($T1, ..., T8$) with 40 instances per set.

Table IV describes the characteristics of some general parameters per instance set. From left to right, the columns indicate the instance set, the project planning horizon, the number of projects, the number of employees, and the last column is the number of tasks. For each group of instances, the number of tasks per agent is chosen between 0.4xJ; 0.5xJ; 0.6xJ, with J being the number of tasks. The total number of skills required to perform the tasks is ranging between 3 and 10; and each employee masters 2 to 5 skills. Each task requires one skill with a nominal load of up to 15. The release dates of 30% of tasks (randomly chosen) range between 0.5xH and 0.75xH, equal to 0 for the other tasks. The due dates of 50% of tasks (also randomly chosen) are between 0.5xH and 0.75xH, equal to H for the other tasks. For 30% of tasks (randomly chosen), the values of the maximum loads (resp. the minimum loads) are between 2 and the nominal loads (resp. 2 and the maximum loads). For 80% of employees, we set the availability at 10 during each week of the planning horizon. The availability of the remaining employees is between 2 and 7.

B. NSGA-II Algorithm Evaluation

We present in this section the computational results of the NSGA-II method. First, we conduct some experiments to adjust the NSGA-II parameters. Then, we perform a second test campaign to compare the performances of the two crossover operators (OBX and PBX). Finally, we conduct our last experiments to measure how much the genetic algorithm improved the results of the greedy heuristic.

TABLE IV. General Parameter Values per Instance Set

Instance set	H	L	I	J
T1	4	2	3	15
T2	4	2	3	20
T3	6	2	5	25
T4	8	2	10	50
T5	10	4	15	100
T6	14	4	20	150
T7	18	4	25	200
T8	24	4	30	250

To evaluate the quality of the returned solutions, we apply three performance metrics widely used for multi-criteria optimization problems. These metrics are the hypervolume (HV)(originally proposed by [48]), the generational distance (GD) (originally proposed by [49]), and the Pareto front size (PFS). Note that in this part of the experiments, we used 10 instances on each set of the dataset. The maximum quotas of employees on projects are initially computed by the MILP presented in Section V-A.

1) Parameters Setting: The NSGA-II has the following parameters to define: the population size P_{max}, number of iterations G_{max}, number of crossover points P_c, and number of mutation points P_m. The NSGA-II algorithm was run 5 times with each combination of parameters and the best values obtained are presented in Table V. To maintain test consistency, for each run, we start the genetic algorithm from the same initial solutions generated by the GH. In order to fix the number of crossover points, we performed the tests with both crossover operators. We noticed reassuringly that the best values obtained are often the same with the two operators.

TABLE V. The Values of Parameters used by the NSGA-II

Parameter	Range	Value
P_{max}	[100,300]	200
G_{max}	[200,3000]	1000
P_c	[1,10]	8
P_m	[1,10]	7

TABLE VI. MILP Results for a Computation Time Limited to 5 min

Instance set	Number of feasible instances	Number of instances solved to optimality	Avg. time to the optimality	Avg. GAP from the optimal
T1	40	40	72.8	0%
T2	40	40	122.2	0%
T3	40	40	150.7	0
T4	40	40	194.6	0%
T5	40	38	256.8	2.5%
T6	40	35	299.2	4.5%
T7	40	32	316.5	76.7%
T8	40	29	396.9	7.6%

2) Comparison of Crossover Operators: Using the parameter values presented in Table V, and from the same initial solutions, we ran the NSGA-II algorithm 5 times using the OBX and PBX crossover operators. The average values of the results were calculated.

Fig. 4 shows the computed generational distances between the exact Pareto and the approximated Pareto fronts obtained with each operator. Note that, we obtain the exact Pareto fronts

TABLE VII. Comparison of Methods NSGA-II and GH

Instance set	GPNE		NSGA-II				AG		
	PFS^*	$CPU^*(s)$	PFS	HV	GD	CPU(s)	PFS	HV	GD
T1	3.42	746.22	2.82	0.82	3.51	177.12	1.81	0.47	8.91
T2	2.91	961.86	2.13	0.76	4.43	182.72	1.54	0.43	9.32
T3	2.52	1021.98	1.89	0.73	3.87	202.98	1.95	0.38	8.12
T4	2.23	1746.78	1.63	0.68	4.43	256.10	1.42	0.35	7.25
T5	1.85	3189.21	1.51	0.64	5.14	301.45	0.26	0.28	7.89
T6	0.78	6976.81	1.51	0.62	5.14	389.95	0.42	0.18	6.91
T7	0.12	8819.11	1.41	0.68	5.14	412.34	0	0.11	6.89
T8	0	-	1.12	0.59	5.14	671.73	0	0.18	7.12

by running the MIGP model without a time limit until the exact Pareto front was returned.

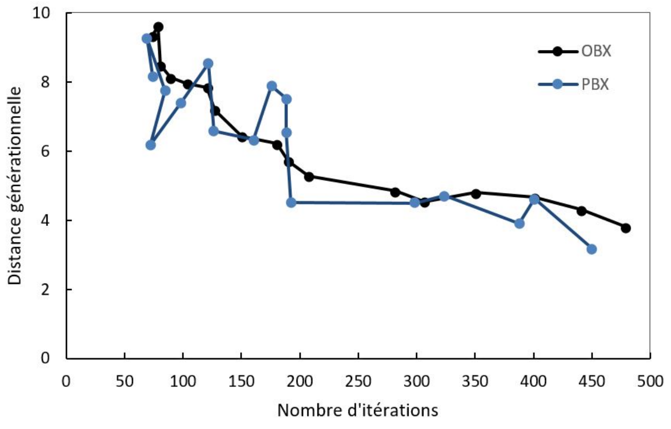

Fig. 4. Results of Crossover Operators

3) Results of Quotas Calculation Procedure: In this section, we present the experimental results of the MILP presented in Section V-A. Recall that, this mathematical model is used by the hybrid method to calculate an allocation (maximum quotas) of employees on projects. This allocation becomes the starting point for generating an approximate Pareto front using the NSGA-II method.

The MILP model was tested on the different sets of instances. Table VI shows the computational results obtained after a time limit of 5 minutes per instance. In this table, from left to right, the columns refer to the type of instance, the number of feasible instances, the number of solutions that are proven to be the optimal solutions, the average time required to prove optimality, and the average deviations from optimal for instances that are not optimally solved.

4) NSGA-II and Greedy Heuristic Comparison: The objective of the experiments presented in this section is to evaluate the improvement provided by the NSGA-II over the initial solutions obtained by the GH. To this end, we compare the results of both methods with those obtained by the MIGP. Note that, we use the best combination of NSGA-II parameters presented in Table V. The solutions returned by the GH are collected after a computation time (AG_{max}) limited to 5 minutes. We run both methods 10 times and the average values are calculated. For each run of the genetic algorithm, we used the same initial solutions generated by the GH, in order to ensure the consistency of the tests.

Table VII compares the experimental results of the NSGA-II and greedy heuristic with those obtained with the GPNE model. We used four different performance indicators: the average Pareto front size (PFS), the average hypervolume (HV), the average generational distance (GD), and the average computation time in seconds (in the CPU column).

5) Experimental Analysis: The comparison results in Fig. 4 show that the OBX operator has a better performance compared to the other PBX operator. In fact, the genetic algorithm has a better and faster convergence with the OBX genetic operator. Moreover, reassuringly it was found that computation time for both crossover operations is almost the same for each problem size.

The experiments in Table VI prove the performance of the MILP model in terms of feasibility and optimality. The model finds a feasible solution for each instance on a set. Moreover, all the solutions obtained for the instances on sets $T1, ..., T4$ are optimal solutions. The model finds only 2, 5, 8 non-optimal solutions (among 40) for the sets of instances $T5$, $T6$, $T7$, respectively. For $T8$ instances, a large number of instances (11) are not solved to the optimum. However, the deviation from the optimum is very acceptable (8).

Finally, the results presented in Table VII allow us to clearly conclude that the NSGA-II algorithm significantly improves the solutions obtained by the greedy algorithm. All the metrics used show that the quality of the solutions is always better for all types of instances. Thus, we can also see that the computation time is very acceptable for a heuristic.

VII. Conclusion

We studied herein a two-agent multi-skill resource-constrained scheduling problem with a global objective. Motivated by a real scheduling case, we considered that each agent manages one or more projects and wants to minimize the total weighted tardiness of its tasks. We consider a pool of employees in which each one can perform a set of skills with heterogeneous performance levels. We assumed that there are some constraints that can be violated when there is no feasible schedule for the problem. Thus, the global objective function seeks to minimize the constraint violations by reducing the undesirable deviations in the soft constraints from their respective goals. The overall objective is to find a schedule that minimizes at the same time both agents objective functions and the global objective function. We provided a mixed-integer goal programming (MIGP) formulation for the problem. Furthermore, we provided a hybrid algorithm combining an exact procedure, a greedy heuristic, and a genetic algorithm to find an approximate Pareto solution set. The performance of the heuristics is evaluated on a set of simulated instances. The results show that the NSGA-II algorithm is the best performing method.

Finally, in future research, we will also focus on other types of two-agent multi-skilled resources scheduling problems. Such as the constrained optimization problem, where the goal is to minimize the global function, subject to the constraints that the objective values of the other do not exceed a given threshold.

References

[1] A. Agnetis, P. B. Mirchandani, D. Pacciarelli, and A. Pacifici, "Scheduling problems with two competing agents," *Operations Research*, vol. 52, no. 2, pp. 229–242, 2004. [Online]. Available: https://doi.org/10.1287/opre.1030.0092

[2] K. Baker and J.-C. Smith, "A multiple-criterion model for machine scheduling," *Journal of Scheduling*, vol. 6, pp. 7–16, 2003. [Online]. Available: https://doi.org/10.1023/A:1022231419049

[3] P. Brucker, B. J. Bernd, and A. Krämer, "Complexity of scheduling problems with multi-purpose machines," *Annals of Operations Research*, vol. 70, no. 0, pp. 57–73, 1997.

[4] E. Néron and D. Baptista, "Lower bounds for the multi-skill project scheduling problem," in *the Eighth International Workshop on Project Management and Scheduling*, 2002, pp. 274–277.

[5] B. Afshar-Nadjafi, "Multi-skilling in scheduling problems: A review on models, methods and applications," *Computers and Industrial Engineering*, vol. 151, no. November 2020, p. 107004, 2021.

[6] S. Hartmann and D. Briskorn, "An updated survey of variants and extensions of the resource-constrained project scheduling problem," *European Journal of Operational Research*, 2021.

[7] M. C. Wu and S. H. Sun, "A project scheduling and staff assignment model considering learning effect," *The International Journal of Advanced Manufacturing Technology*, vol. 28, pp. 1190–1195, 2006.

[8] C. Heimerl and R. Kolisch, "Scheduling and staffing multiple projects with a multi-skilled workforce," *OR Spectrum*, vol. 32, no. 2, pp. 19–25, 2010.

[9] R. Kolisch and C. Heimerl, "An efficient metaheuristic for integrated scheduling and staffing it projects based on a generalized minimum cost flow network," *Naval Research Logistics (NRL)*, vol. 59, no. 2, pp. 111–127, 2012.

[10] T. Felberbauer, W. J. Gutjahr, and K. F. Doerner, "Stochastic project management: multiple projects with multi-skilled human resources," *Journal of Scheduling*, vol. 22, pp. 271–288, 2019.

[11] M. Walter and J. Zimmermann, "Minimizing average project team size given multi-skilled workers with heterogeneous skill levels," *Computers & Operations Research*, vol. 70, pp. 163–179, 2016.

[12] W. J. Gutjahr, S. Katzensteiner, P. Reiter, C. Stummer, and M. Denk, "Multi-objective decision analysis for competence-oriented project portfolio selection," *European Journal of Operational Research*, vol. 205, no. 3, pp. 670–679, 2010.

[13] R. Chen, C. Liang, D. Gu, and J. Leung, "A multi-objective model for multi-project scheduling and multi-skilled staff assignment for it product development considering competency evolution," *International Journal of Production Research*, vol. 55, no. 21, pp. 6207–6234, 2017.

[14] M. Hematian, M. Esfahani, I. Mahdavi, N. Mahdavi-Amiri, and J. Rezaeian, "A multiobjective integrated multiproject scheduling and multi-skilled workforce assignment model considering learning effect under uncertainty," *Computational Intelligence*, vol. 36, no. 1, pp. 276–296, 2020.

[15] M. Walter, *Multi-Project Management with a Multi-Skilled Workforce*. Springer Gabler, Wiesbaden, 2015.

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

[17] C. T. Ng, T. C. Cheng, and J. J. Yuan, "A note on the complexity of the problem of two-agent scheduling on a single machine," *Journal of Combinatorial Optimization*, vol. 12, no. 4, pp. 386–393, 2006.

[18] A. Agnetis, G. De Pascale, and D. Pacciarelli, "A lagrangian approach to single-machine scheduling problems with two competing agents," *Journal of Scheduling*, vol. 12, no. 4, pp. 401–415, 2009.

[19] J. Li, Y. Gajpal, and S. S. Appadoo, "Algorithms for a two-agent single machine scheduling problem to minimize weighted number of tardy jobs," *Journal of Information and Optimization Sciences*, vol. 42, no. 4, pp. 785–811, 2021.

[20] X. Zhang, "Two competitive agents to minimize the weighted total late work and the total completion time," *Applied Mathematics and Computation*, vol. 406, p. 126286, 2021.

[21] W. C. Lee, J. Y. Wang, and H. W. Su, "Algorithms for single-machine scheduling to minimize the total tardiness with learning effects and two competing agents," *Concurrent Engineering Research and Applications*, vol. 23, no. 1, pp. 13–26, 2015.

[22] T. C.E.Cheng, S.-R. Cheng, W.-H. Wu, P.-H. Hsu, and C.-C. Wu, "A two-agent single-machine scheduling problem with truncated sum-of-processing-times-based learning considerations," *Computers and Industrial Engineering*, vol. 60, no. 4, pp. 534–541, 2011. [Online]. Available: http://dx.doi.org/10.1016/j.cie.2010.12.008

[23] Y. Yin, S.-R. Cheng, T. Cheng, C.-C. Wu, and W.-H. Wu, "Two-agent single-machine scheduling with assignable due dates," *Applied Mathematics and Computation*, vol. 219, no. 4, pp. 1674–1685, 2012.

[24] D.-J. Wang, Y. Yin, J. Xu, W.-H. Wu, S.-R. Cheng, and C.-C. Wu, "Some due date determination scheduling problems with two agents on a single machine," *International Journal of Production Economics*, vol. 168, pp. 81–90, 2015.

[25] E. Gerstl and G. Mosheiov, "Scheduling problems with two competing agents to minimized weighted earliness–tardiness," *Computers & Operations Research*, vol. 40, no. 1, pp. 109–116, 2013. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0305054812001359

[26] S. S. Li, R. X. Chen, and Q. Feng, "Scheduling two job families on a single machine with two competitive agents," *Journal of Combinatorial Optimization*, vol. 32, no. 3, pp. 784–799, 2016.

[27] S. N. Sahu, Y. Gajpal, and S. Debbarma, "Two-agent-based single-machine scheduling with switchover time to minimize total weighted completion time and makespan objectives," *Annals of Operations Research*, vol. 269, no. 1-2, pp. 623–640, 2018.

[28] B.-C. Choi, M.-J. Park, and J. Du, "Scheduling two projects with controllable processing times in a single-machine environment," *Journal of Scheduling*, vol. 23, no. 1, pp. 619–628, 2020.

[29] H. Balasubramanian, J. Fowler, A. Keha, and M. Pfund, "Scheduling interfering job sets on parallel machines," *European Journal of Operational Research*, vol. 199, no. 1, pp. 55–67, 2009.

[30] K. Zhao and X. Lu, "Approximation schemes for two-agent scheduling on parallel machines," *Theoretical Computer Science*, vol. 468, pp. 114–121, 2013.

[31] W.-C. Lee, J.-Y. Wang, and M.-C. Lin, "A branch-and-bound algorithm for minimizing the total weighted completion time on parallel identical machines with two competing agents," *Knowledge-Based Systems*, vol. 105, pp. 68–82, 2016.

[32] K. Zhao and X. Lu, "Two approximation algorithms for two-agent scheduling on parallel machines to minimize makespan," *Journal of Combinatorial Optimization*, vol. 31, no. 1, pp. 260–278, 2016. [Online]. Available: http://dx.doi.org/10.1007/s10878-014-9744-y

[33] J. Y.-T. Leung, M. Pinedo, and G. Wan, "Competitive two-agent scheduling and its applications," *Operations Research*, vol. 58, no. 2, pp. 458–469, 2010.

[34] G. Wan, S. R. Vakati, J. Y.-T. Leung, and M. Pinedo, "Scheduling two agents with controllable processing times," *European Journal of Operational Research*, vol. 205, no. 3, pp. 528–539, 2010.

[35] D. Li and X. Lu, "Two-agent parallel-machine scheduling with rejection," *Theoretical Computer Science*, vol. 703, pp. 66–75, 2017. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0304397517306527

[36] J. Pei, J. Wei, B. Liao, X. Liu, and P. M. Pardalos, "Two-agent scheduling on bounded parallel-batching machines with an aging effect of job-position-dependent," *Annals of Operations Research*, vol. 294, no. 1-2, pp. 191–223, 2020.

[37] F. Sadi and A. Soukhal, "Complexity analyses for multi-agent scheduling problems with a global agent and equal length jobs," *Discrete Optimization*, vol. 23, pp. 93–104, 2017.

[38] D. Elvikis, H. W. Hamacher, and V. T'kindt, "Scheduling two agents on uniform parallel machines with makespan and cost functions," *Journal of Scheduling*, vol. 14, no. 5, pp. 471–481, 2011.

[39] D. Elvikis and V. T'kindt, "Two-agent scheduling on uniform parallel machines with min-max criteria," *Annals of Operations Research*, vol. 213, no. 1, pp. 79–94, 2014.

[40] Y. Yin, S.-R. Cheng, T. Cheng, D.-J. Wang, and C.-C. Wu, "Just-in-time scheduling with two competing agents on unrelated parallel machines," *Omega*, vol. 63, pp. 41–47, 2016. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0305048315002042

[41] Y. Yin, Y. Chen, K. Qin, and D. Wang, "Two-agent scheduling on unrelated parallel machines with total completion time and weighted number of tardy jobs criteria," *Journal of Scheduling*, vol. 22, no. 3, pp. 315–333, 2019.

[42] R. Graham, E. Lawler, J. Lenstra, and A. Kan, "Optimization and approximation in deterministic sequencing and scheduling: a survey," in *Discrete Optimization II*, ser. Annals of Discrete Mathematics, P. Hammer, E. Johnson, and B. Korte, Eds. Elsevier, 1979, vol. 5, pp. 287–326.

[43] A. Agnetis, J.-C. Billaut, S. Gawiejnowicz, D. Pacciarelli, and A. Soukhal, *Multiagent Scheduling: Models and Algorithms*. Springer, Berlin, Heidelberg, 2014.

[44] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 2, pp. 182–197, 2002.

[45] N. Srinivas and K. Deb, "Muiltiobjective Optimization Using Nondominated Sorting in Genetic Algorithms," *Evolutionary Computation*, vol. 2, no. 3, pp. 221–248, 1994.

[46] G. Syswerda, *Scheduling optimization using genetic algorithms*, New York, NY, 1991.

[47] T. Kellegöz, B. Toklu, and J. Wilson, "Comparing efficiencies of genetic crossover operators for one machine total weighted tardiness problem," *Applied Mathematics and Computation*, vol. 199, no. 2, pp. 590–598, 2008.

[48] E. Zitzler and L. Thiele, "Multiobjective optimization using evolutionary algorithms — a comparative case study," in *Parallel Problem Solving from Nature — PPSN V*, A. E. Eiben, T. Bäck, M. Schoenauer, and H.-P. Schwefel, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 1998, pp. 292–301.

[49] D. A. V. Veldhuizen and G. B. Lamont, "Multiobjective evolutionary algorithms: Analyzing the state-of-the-art," *Evolutionary Computation*, vol. 8, no. 2, pp. 125–147, 2000.