PAMUKKALE ÜNİVERSİTESİ
Mühendislik Bilimleri Dergisi

Araştırma makalesi | Research paper

PAMUKKALE UNIVERSITY
Journal of Engineering Sciences

Cilt | Vol: XX • Sayı | No: X • ss | pp: XXXX-XXXX • Yıl | Year: 20XX

Dynamic task assignment in IT departments using a rolling-horizon mixed-integer programming model

Canan Pehlivan

ARTICLE INFO

Dates:
Received: 30.04.2026
Accepted: 01.09.2026

Doi:
10.65206/pajes.1940880

Corresponding author:
Canan Pehlivan
(canan.pehlivan@yeditepe.edu.tr)

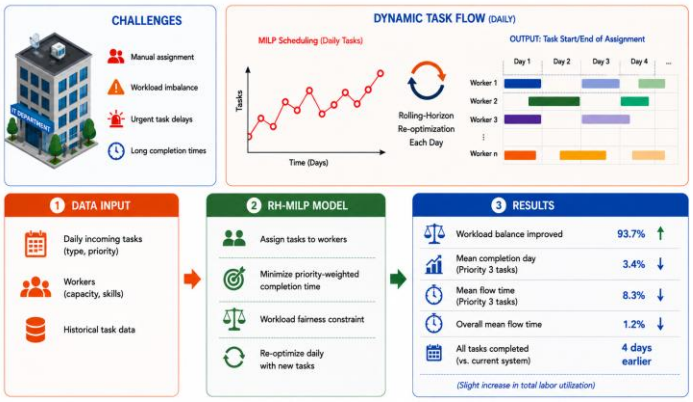
Author addresses:
Industrial Engineering Department, Faculty of Engineering and Natural Sciences, Yeditepe University, İstanbul, 34755, Türkiye
(canan.pehlivan@yeditepe.edu.tr)

ABSTRACT

Context—Information Technologies (IT) departments usually handle large volumes of support and development requests with varying urgency, complexity, and resource requirements every day. Efficient assignment of these requests to employees is essential for maintaining service quality, minimizing completion times, and ensuring balanced workforce utilization. However, in many organizations, including major telecommunications companies in Türkiye, task assignments are still performed manually and without a structured decision-support system. This often leads to workload imbalances, insufficient prioritization of urgent tasks, and long task completion times.

Objective—The aim of this study is to develop an optimization-based decision-support framework to efficiently assign the daily tasks to employees in an IT department.

Method—To address this problem, a Mixed Integer Linear Programming (MILP) model is developed to assign tasks considering the task type, priority level, and individual capacities of the workers. We propose a rolling-horizon framework where the MILP model re-optimizes newly arriving tasks each day while keeping previously implemented assignments.



In this way, the system can adapt to dynamic task flows. The proposed approach is implemented in Python using the Gurobi optimizer, tested with real operational data from a major telecommunications company in Türkiye.

Results—The computational results show that our approach improves both task completion performance and fairness compared with the current system. The workload balance improves substantially, with a 93.7% reduction in the standard deviation of cumulative workload, while the mean completion day and mean flow time of urgent (Priority 3) tasks decrease by 3.4% and 8.3%, respectively. These improvements come with a 6.7% increase in mean cumulative workload, indicating a trade-off among task completion performance, workload fairness, and total labor requirements. Despite higher task-day requirements, the optimized system reduces the overall mean flow time by 1.3%, and all tasks are completed four days earlier compared to the actual system. A sensitivity analysis is conducted on the workload balance parameter Δ (maximum allowable workload difference between workers). Results point out a clear trade-off between fairness and responsiveness: smaller Δ values provide better workload distribution but generally require higher task processing requirements (mean cumulative workload), whereas higher Δ values improve completion performance, especially for urgent tasks, at the expense of increased workload inequality.

Conclusion—Our findings demonstrate that the proposed rolling-horizon MILP framework provides an effective, practical, and adaptable decision-support tool for IT task assignment operations. The results show that optimization-based decision-support tools can improve workload balance and task completion performance while requiring slightly higher labor requirements. The results also demonstrate a trade-off between fairness and task completion speed, allowing decision makers to select the desired workload balance according to operational priorities. Future studies may incorporate task arrival forecasting, stochastic processing times, skill-based worker constraints, and real-time disruption scenarios to test the robustness and generalizability of the framework.

Key Words—Information technologies, Rolling-horizon MILP, Task-worker assignment, Workload balance.

I. INTRODUCTION

Today, information technology (IT) departments in modern organizations face increasing flow of requests from different business units. These tasks vary in terms of priority, complexity, and timing. However, even major telecommunications companies in Türkiye still manage these assignments without a systematic approach. In the company we studied, the daily task assignments are done manually using a simple rule-based heuristic: tasks are sorted by priority and release date, the tasks are typically assigned to the fastest available worker, considering current worker workload to some extent. Although practical and easy to apply, such manual procedures often lead to suboptimal operations, late completion times, underutilization of workers and uneven workload distribution among workers.

To address these issues, we develop a mathematical optimization-based framework for dynamic task assignments. Our approach integrates a Mixed Integer Linear Programming (MILP) model with a rolling-horizon (RH) planning framework to balance workload distribution while prioritizing urgent tasks. The rolling-horizon framework allows the model to re-optimize task assignments each day by incorporating newly arriving tasks while preserving previously implemented assignments.

The proposed model is implemented using a real dataset obtained from the IT operations of a major telecommunications company in Türkiye. The outputs of the model include the assignment of each task to a specific worker, the corresponding expected start and completion times, and the status of ongoing tasks. By considering task priorities, durations, and workload balance jointly, the proposed system improves operational efficiency, enabling more tasks to be completed within the same time frame while reducing workload variability and inequity among workers.

A. Background and literature review

The literature is organized into three streams. The first concerns workforce and task assignment models, with particular attention to studies that incorporate workload balance. The second addresses dynamic workforce and task assignment, where new jobs arrive over time and assignment decisions are updated as new information becomes available. The third concerns rolling-horizon optimization. We review these three streams and position our study accordingly. Table 1 compares the relevant studies in terms of key problem characteristics and solution approaches and positions our study within the existing literature.

Workforce and task assignment problems allocate tasks to workers while considering factors such as task priorities, requirements, processing times, worker capacity, and skills. However, when tasks are repeatedly assigned to the fastest or most experienced workers, serious workload differences occur which may lead to employee dissatisfaction, burnout risk, and may reduce long-term stability. For this reason, workload equity has become an increasingly important issue in workforce scheduling and task assignment literature [1], [2]. Eiselt and Marianov (2008) consider employee-task assignment while jointly accounting for task suitability, assignment cost, and inequity in employee workloads [3]. Liang et al. (2010) study project assignments to engineers with an objective to balance workload which is measured by the difference between the maximum and minimum total workload across engineers [4]. Lapègue et al. (2013) consider personnel task scheduling with an explicitly defined workload equity criterion [5]. Smet et al. (2014) address the assignment of scheduled tasks to multi-skilled employees and develop a hybrid heuristic for minimizing the required shifts [6]. Düzgit et al. (2019) develop a mixed-integer programming model for service-job assignment and routing [7]. Wang et al. (2023) integrate staff rostering and task assignment while incorporating fairness, and other real-world requirements [8]. Jin et al. (2025) study technician and task scheduling in a cloud-computing company while jointly considering processing efficiency, response delay and workload balance [9].

Workload equity has been modeled using different measures. While Liang et al. (2010) use the difference between maximum and minimum worker workloads [4], Karsu and Azizoğlu (2019) minimize the sum of squared agent workloads [10]. More generally, Karsu and Morton (2015) review approaches for incorporating equity into operational research models and discuss the trade-off that may arise between equity and efficiency. Workload-dispersion measures have also been used in related scheduling settings [11]. Cossari et al. (2012) study workload balancing on identical parallel machines using the normalized sum of squared workload deviations [12], while Stolletz and Brunner (2012) incorporate workload-related fairness considerations into physician scheduling [13]. More recently, Huang et al. (2026) investigate fairness-aware task assignment in airport operations by penalizing deviations between actual and expected worker workloads using a quadratic imbalance function [14]. These studies show that workload equity can be represented through different measures and incorporated into models in different ways.

The second stream of research concerns dynamic workforce and task assignment, where new tasks arrive over time and decisions must be updated repeatedly. Chen et al. (2017) study multi-period technician scheduling with stochastic customer arrivals and worker-dependent service times [15]. Chen et al. (2018) later propose an approximate dynamic programming approach for the same setting [16]. Annear et al. (2023) analyze the dynamic assignment of a multi-skilled workforce in job shops under

Table 1. Comparison of the proposed study with the related studies in literature.

Study	Application	Worker-dependent duration	Dynamic arrivals	Task Priority	Simultaneous Task Handling	Workload fairness	Solution approach
[4]	Project–engineer assignment	–	–	–	✓	✓	Two-stage heuristic
[5]	Personnel task scheduling	–	–	–	–	✓	Constraint prog.
[8]	Integrated staff rostering/task assignment	–	–	–	–	✓	MIP + rolling-horizon heuristic
[9]	Technician-task scheduling in cloud services	✓	–	–	✓	✓	Hierarchical optimization / GRASP
[10]	Task–agent assignment	✓	–	–	✓	✓	MILP + branch-and-bound
[14]	Airport staff task assignment	–	–	–	–	✓	Multi-objective model + approximation
[16]	Technician–task scheduling	✓	✓	–	–	–	Approximate dynamic prog.
[17]	Multi-skilled workforce/job-shop assignment	✓	✓	–	–	–	Approximate dynamic prog.
[18]	Online warehouse task assignment	✓	✓	–	–	✓	Data-driven optimization / MIP
This study	**IT task-worker assignment**	**✓**	**✓**	**✓**	**✓**	**✓**	**Rolling-horizon MILP**

demand uncertainty and variability in resource availability [17]. More recently, Wang et al. (2026) consider online warehouse task assignment with dynamically varying worker capabilities and develop a data-driven optimization approach for updating task assignments as operational conditions change [18]. These studies demonstrate the importance of adaptive assignment policies when future demand or worker performance is not fully known at the time of decision making.

The third-stream concerns rolling-horizon (RH) optimization, which is suitable for these dynamic environments. RH repeatedly solves smaller optimization models over a finite planning window and re-optimizes as new information becomes available. Stolletz and Zamorano (2014) develop a rolling planning horizon heuristic for scheduling multi-skilled airport check-in agents, where shorter horizon subproblems are solved iteratively [19]. Tarhan et al. (2023) apply a rolling-horizon personnel routing and scheduling framework in disaster-response operations [20]. Wang et al. (2023) develop a rolling-horizon heuristic for integrated staff rostering and task assignment [8].

Overall, previous studies address workload equity, heterogeneous worker-task assignment, dynamic demand, and rolling-horizon decision making in a variety of workforce settings. However, relatively limited work jointly considers dynamic daily task arrivals, worker-dependent task durations, simultaneous task handling, task priorities, continuity of ongoing assignments, and cumulative workload equity within a rolling-horizon framework for internal IT operations. Our study addresses this combination using real operational data from a telecommunications company. In contrast to approaches that incorporate workload equity through an additional objective term or dispersion measure, workload balance is imposed through an explicit bound on the difference between the maximum and minimum cumulative worker workloads. This allows the trade-off between workload equity and task-completion performance to be controlled via a single parameter.

II. METHOD

We propose a MILP approach within a rolling-horizon framework to solve a multi-day task assignment problem in a dynamic IT operations environment. The proposed approach combines mathematical modeling and iterative planning to generate daily task assignments while considering task priorities, limited capacity, and workload balance.

At each decision epoch, tasks are assigned to workers over a planning window of n days starting from the current day k. At the beginning of each day, new tasks arrive and are included in the decision process. Future task arrivals are unknown; thus, the MILP model repeatedly optimizes the schedules as new information becomes available.

Each task j is characterized by a fixed priority, task type, and a worker-dependent processing duration d_{ij} estimated from historical data. The processing duration represents the number of working days for which task j is active when assigned to worker i. Since a worker may handle multiple tasks simultaneously, daily worker capacity is defined as the maximum number of tasks that can be simultaneously active for a worker on a given day. Workload is measured in task-days, where one task processed by one worker on one day contributes one task-day to that worker's cumulative workload. For example, if a worker handles four active tasks on a given day, four task-days are added to the worker's cumulative workload.

Following the company's priority classification and preferences, Priority 1, Priority 2, and Priority 3 tasks are assigned weights of 1, 2, and 3, respectively. Thus, Priority 3 represents the highest-priority tasks and receives the largest weight in the objective function. All workers are assumed to be able to perform all tasks, but with different processing durations. Fairness among workers is maintained by Δ, which limits differences in cumulative task-day workload among workers. High-priority tasks are treated as non-preemptive, whereas medium- and low-priority tasks may be interrupted and resumed later.

The proposed methodology uses a MILP model within a rolling-horizon framework. At each decision epoch, the MILP model is solved to determine the task-worker assignments over the current planning window. Assignments that have already been implemented are fixed, while tasks that have not yet started remain flexible for rescheduling.

A. The mixed integer linear programming (MILP) model

This section presents the MILP formulation used to assign and schedule tasks over the planning window, considering the definitions listed in Table 2. The model aims to minimize the priority-weighted task completion times while ensuring a fair distribution of workload across workers and respecting capacity limits.

The objective function minimizes the total priority-weighted completion time of waiting tasks. This formulation promotes the early completion of urgent tasks over less urgent ones. The complete MILP formulation is presented in (1)-(12).

$$Minimize \qquad Z = \sum_{j \in J_k^{waiting}} p_j \, C_j.$$

subject to

$$\sum_{i \in I} y_{ij} = 1, \ \ \forall j \in J_k^{waiting} \tag{1}$$

$$\sum_{t \in T_k : t \le t_{\max} - d_{ij} + 1} s_{ijt} = y_{ij}, \ \ \forall i \in I, \ j \in J_k^{waiting} \tag{2}$$

$$\sum_{\tau = t}^{t + d_{ij} - 1} x_{ij\tau} \ge d_{ij} s_{ijt}, \ \ \forall i \in I, \ j \in J_k^{high} \tag{3.1}$$

$$\sum_{t \in T_k} x_{ijt} = d_{ij} y_{ij}, \ \ \forall i \in I, \ j \in J_k^{high} \tag{3.2}$$

$$\sum_{t \in T_k} x_{ijt} = d_{ij} y_{ij} - P_{ij}^k, \ \ \forall i \in I, \ j \in J_k^{flex} \tag{4.1}$$

$$x_{ijt} \le \sum_{\tau \in T_k : \tau \le t} s_{ij\tau}, \ \ \forall i \in I, \ j \in J_k^{flex}, \ t \in T_k \tag{4.2}$$

$$s_{ijt} = 0, \ \ \forall i \in I, \ j \in J_k^{waiting}, \ t > t_{\max} - d_{ij} + 1 \tag{5}$$

$$C_j \ge t \sum_{i \in I} x_{ijt}, \ \ \forall j \in J_k^{waiting}, \ t \in T_k \tag{6}$$

$$y_{ij} = \bar{y}_{ij}, \ \ \forall i \in I, \ j \in J_k^{ongoing}, t \in T_k \tag{7.1}$$

$$s_{ijt} = \bar{s}_{ijt}, \ \ \forall i \in I, \ j \in J_k^{ongoing}, t \in T_k \tag{7.2}$$

$$x_{ijt} = \bar{x}_{ijt}, \ \ \forall i \in I, \ j \in J_k^{ongoing}, t \in T_k \tag{7.3}$$

$$\sum_{j \in J_k^{all}} \bar{x}_{ijt} \le cap \, z_{it}, \ \ \forall i \in I, \ t \in T_k \tag{8}$$

$$O_i^{k,10} + \sum_{t \in T_k^{10}} (1 - z_{it}) \ge 2, \ \ \forall i \in I \tag{9.1}$$

$$O_i^{k,20} + \sum_{t \in T_k^{20}} (1 - z_{it}) \ge 4, \ \ \forall i \in I \tag{9.2}$$

$$O_i^{k} + \sum_{t \in T_k^{mon}} (1 - z_{it}) \le 8, \ \ \forall i \in I \tag{9.3}$$

$$w_i^{tot} = H_i + \sum_{j \in J_k^{all}} \sum_{t \in T_k} x_{ijt}, \ \ \forall i \in I \tag{10.1}$$

$$w_i^{tot} \le w_{\max}, \ \ \forall i \in I \tag{10.2}$$

$$w_i^{tot} \ge w_{\min}, \ \ \forall i \in I \tag{10.3}$$

$$w_{\max} - w_{\min} \le \Delta \tag{10.4}$$

$$x_{ijt}, z_{it}, y_{ij}, s_{ij\tau} \in \{0,1\}, \ \ \forall i, j, t \tag{11}$$

$$w_i^{tot}, w_{\max}, w_{\min}, C_j \ge 0, \ \ \forall i, j \tag{12}$$

Table 2. Variables and corresponding definitions for index, set, model, and decision parameters.

Parameters	Variables	Definitions
Index	n	Planning window length (days)
	k	Current day
	$t_{\min}$	First day of the planning window, $t_{\min} = k$
	$t_{\max}$	Last day of the planning window, $t_{\max} = k + n - 1$
Set	I	Set of workers
	T_k	Set of planning days in the planning window, $T_k = \{t \in Z_+ \mid t_{\min} \le t \le t_{\max}\}$
	J_k^{new}	Tasks arriving at the beginning of day k
	J_k^{done}	Tasks completed by the end of day $k-1$
	J_k^{open}	Tasks that have arrived by day k and are not completed, $J_k^{open} = (\cup_{\tau \le k} J_\tau^{new}) \setminus J_k^{done}$
	$J_k^{ongoing}$	Ongoing tasks at day k, $J_k^{ongoing} \subseteq J_k^{open}$
	$J_k^{waiting}$	Waiting tasks to be assigned, $J_k^{waiting} = J_k^{open} \setminus J_k^{ongoing}$
	J_k^{high}	High-priority tasks in the waiting set, $J_k^{high} = \{j \in J_k^{waiting} \mid p_j = 3\}$
	J_k^{flex}	Interruptible tasks in the waiting set, $J_k^{flex} = \{j \in J_k^{waiting} \mid p_j < 3\}$
	J_k^{all}	All open tasks at day k, $J_k^{all} = J_k^{open} (= J_k^{ongoing} \cup J_k^{waiting})$
	T_k^{mon}	subset of days in T_k that belong to the current calendar month
	T_k^{τ}	subset of current-month days up to the day-τ checkpoint.
Model	p_j	Priority weight of task j: $p_1 = 1$, $p_2 = 2$, $p_3 = 3$
	d_{ij}	Worker-dependent processing duration of task j when assigned to worker i (measured in working days)
	cap	Maximum number of tasks that can be simultaneously active for a worker on a day
	Δ	Maximum allowed cumulative workload imbalance
	H_i	Cumulative number of already processed task-days of worker i before day k
	O_i^k	Number of off-days already taken by worker i in the current month before day k
	$O_i^{k,\tau}$	Number of off-days already taken by worker i before day k and before the day-τ checkpoint of the current month.
	$\bar{y}_{ij}, \bar{s}_{ijt}, \bar{x}_{ijt}$	Fixed decisions carried from previous RH iterations
	P_{ij}^k	Number of days of interruptible task j already completed before day k
Decision	z_{it}	1 if worker i is available on day t
	y_{ij}	1 if task j is assigned to worker i
	s_{ijt}	1 if task j starts on day t by worker i
	x_{ijt}	1 if worker i processes task j on day t
	C_j	Completion day of task j
	w_i^{tot}	Cumulative workload of worker i, measured in task-days
	$w_{\min}, w_{\max}$	Minimum and maximum workloads

Constraint (1) ensures that each waiting task is assigned to exactly one worker. Constraint (2) ensures that if a task is assigned, it must have exactly one feasible start day. High-priority tasks should be treated as non-preemptive; thus, constraints (3.1) and (3.2) enforce non-preemption for high-priority tasks; once processing starts, it must continue without interruption. In contrast, medium and low-priority tasks may be flexibly rescheduled within the planning window after they start, thus Constraints (4.1) and (4.2) guarantee that the remaining processing duration of the interruptible jobs must be completed. Constraint (5) ensures that a task can start only if it can be completed within the current planning window, thus it prevents infeasible late starts. Tasks that cannot be assigned within the current planning window remain in the waiting set and are reconsidered at the next decision epoch as the planning window moves forward. Constraint (6) defines the completion time of each task as the last day of processing. In constraints (7.1-7.3), assignments and schedules of ongoing tasks are fixed based on the decision taken in the previous rolling-horizon iterations. Tasks already in progress preserve their previously determined assignments. In this way, the model maintains operational stability from one day to the next. Constraint (8) restricts the maximum number of tasks that can be simultaneously active for each worker on a day. This constraint reflects the company's practice to prevent overload. Constraints (9) regulate the distribution of off days throughout the month. According to company policy, each worker should take eight off-days in a 30-day month, with at least two taken before the 10th day of the month and at least four taken before the 20th day. Constraints (10.1-10.4) define the cumulative worker workload and enforce fairness bounds. In constraint (10.1), the cumulative workload of each worker (w_i^{tot}) is measured in task-days and is calculated as the sum of the days for which tasks are actively processed by that worker within the current planning window, combined with the historical task-day workload accumulated before the current day (H_i). Including H_i is important because fairness should be assessed cumulatively over time rather than only within a single horizon. Otherwise, a worker heavily loaded in previous periods could still receive many new tasks simply because the current horizon appears balanced. In constraints (10.2-10.4), the difference between the minimum and maximum cumulative workload across all workers is restricted by the parameter Δ. This means that no worker's cumulative workload can exceed another worker's by more than the allowed imbalance threshold. The value of Δ is important in determining the trade-off between fairness and operational responsiveness. Thus, we analyze its impact on the key performance metrics in the computational section. Finally, constraints (11) and (12) specify the domains of the decision variables.

B. The rolling horizon approach

The problem is addressed using a rolling-horizon (RH) approach in which the optimization window moves forward one day at each iteration. At the beginning of each day k, tasks are categorized into completed tasks (J_k^{done}), ongoing tasks ($J_k^{ongoing}$), open tasks (J_k^{open}), and waiting tasks ($J_k^{waiting}$).

Completed tasks are removed from the task list. Open tasks include all tasks that have arrived and are not yet completed. Open tasks may include tasks that have already started before day k. For ongoing high-priority tasks, worker assignment and schedule decisions are kept fixed to preserve non-preemption. For ongoing medium and low-priority tasks, worker assignment is kept fixed, while the remaining processing periods can be rescheduled within the planning window. The remaining open tasks that have not yet started create the waiting set and can be rescheduled.

The pseudocode of the RH procedure is presented in Table 3. At the start of each day, the sets and the planning window are updated. Completed tasks are removed from the ongoing set and added to the completed set. Newly arriving tasks are added to the open set, and the waiting set is updated accordingly. After the

Table 3. Rolling Horizon (RH) procedure.

Initialization	Set $k = 1$ and choose the planning-window length n (e.g., $n = 30$days). Initialize the sets $J_1^{done} = \emptyset$, $J_1^{open} = \emptyset$, and $J_1^{ongoing} = \emptyset$. Let $T_k = \{t \in \mathbb{Z}_+ \mid k \le t \le k + n - 1\}$.
Step 1	**At the beginning of day k:** **Update sets** Get new arrivals J_k^{new}. Update completed tasks J_k^{done}. Update open tasks: $J_k^{open} = \left(\bigcup_{\tau \le k} J_\tau^{new}\right) \setminus J_k^{done}$. Update ongoing tasks: $J_k^{ongoing} = \left\{j \in J_k^{open} : \sum_{i \in I} \sum_{t<k} s_{ijt} = 1\right\}$. Update waiting tasks: $J_k^{waiting} = J_k^{open} \setminus J_k^{ongoing}$. Set $J_k^{all} = J_k^{open}$.
Step 2	$y_{ij} = \bar{y}_{ij},\ x_{ijt} = \bar{x}_{ijt},\ s_{ijt} = \bar{s}_{ijt},\ \ \forall i \in I,\ t \in T_k$. Here, $(\bar{y}, \bar{x}, \bar{s})$ denote the previously committed decisions. Decisions for $j \in J_k^{waiting}$, including J_k^{new}, remain free.
Step 3	**Implement first-day decisions.** Commit only the decisions for day $t = k$; update the system state, including which jobs start on day k, progress on ongoing jobs, and any disruptions, if applicable.
Step 4	**Advance time.** Set $k \leftarrow k + 1$ and repeat until the end of the 60-day evaluation period.

update, the MILP is solved over the current planning window considering both ongoing and waiting tasks. Only the decisions corresponding to the current day are implemented. The planning window is then advanced by one day, and the process repeats. At each decision epoch k, the optimization model considers a planning window $T_k = \{k, \dots, k + n - 1\}$, where n is the planning window length and is set to 30 days in the baseline analysis. Throughout this paper, the term "planning window" refers to this n-day optimization window, "evaluation period" refers to the 60-day sequence of task arrivals and rolling-horizon decision epochs.

III. RESULTS and DISCUSSION

This section presents the dataset and current company assignment procedure, followed by computational results and sensitivity analyses.

A. Data description

This project is implemented using real operational data obtained from the IT department of a major telecommunications company in Türkiye. The dataset contains 118 support and development tasks arriving over 60 working days, with 1 to 5 new tasks entering the system each day.

Each task is characterized by its arrival day, task type, and priority level, where priority 1 denotes low urgency, priority 2 medium urgency, and priority 3 high urgency. Tasks are classified into four types reflecting the operational structure of a large telecommunications IT department: incident resolution, routine service requests, application maintenance, and project-based activities. Fig. 1a illustrates the daily task arrivals by priority level, showing both the pattern of task arrivals and the frequency of high-priority tasks. Fig. 1b summarizes the tasks by their type and priority level. The relationship between task type and priority level is examined using Pearson's chi-square test. The test indicated no statistically significant association between task type and priority level, $\chi^2(6) = 10.44$, $p = 0.107$, and $V_{Cramér} = 0.21$, suggesting that no particular task type systematically has a higher or lower priority level.

There are 10 workers. Every worker can perform all task types, but workers' task processing times may be different based on individual experience. Task durations are worker and task-type dependent and are estimated from the company's historical data, presented in Fig. 2. To prevent excessive workload, a limit is defined on the number of simultaneously active tasks for each worker based on historical capacity observations. High-priority tasks are assumed to be non-preemptive, meaning that once they start, they must be processed by the same worker on consecutive days until completion, whereas medium and low-priority tasks may be interrupted and resumed later.

B. Actual system: Company assignment procedure

In the company, daily task assignments are currently carried out manually according to a rule-based dispatching procedure. At the beginning of each day, newly arrived and previously waiting tasks are ordered by priority level, from highest to lowest, and then by release date, from earliest to latest. Each task is assigned to a worker with available capacity, giving preference to the worker with the shortest processing time for the corresponding task type. If there is a tie, the task is assigned to the worker with the fewest active tasks. If no worker has available capacity, the task remains in the waiting set and is reconsidered on the following day. Workers follow a fixed weekly off-day pattern, with two off-days per week. Each worker has a limited daily handling capacity and can process at most six active tasks simultaneously per day, which is decided by the company. Assigned tasks remain with the same worker until completion. On

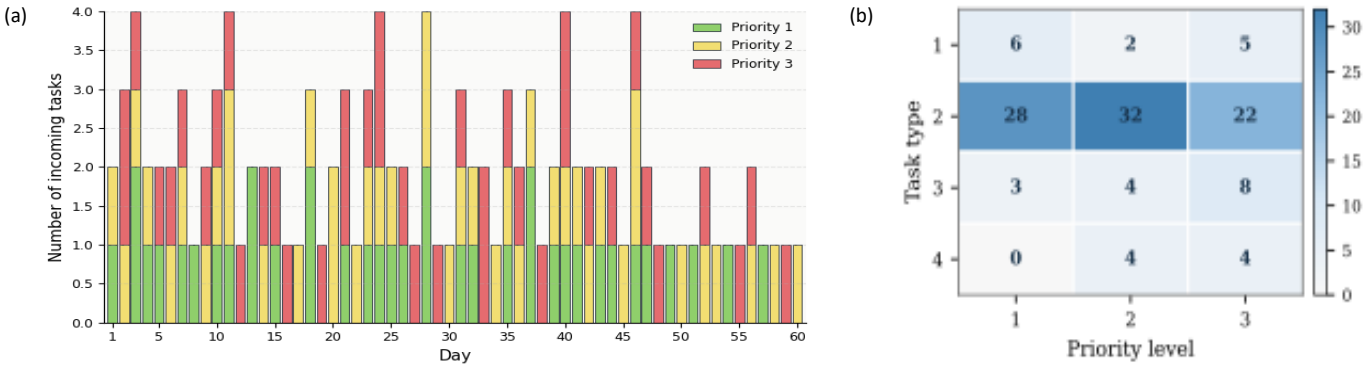


Fig. 1. (a) Daily incoming tasks by priority level, (b) Number of tasks by type and priority level.

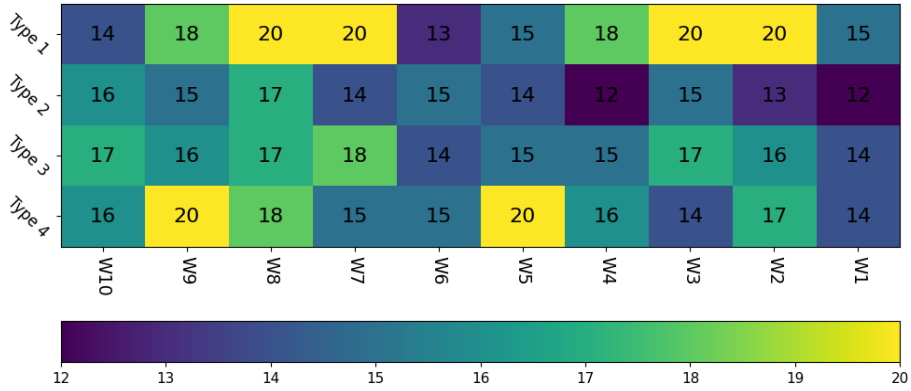


Fig. 2. Heatmap of Worker vs. Average task duration (number of days) of task types.

each working day, active tasks are processed according to priority and release date; unfinished tasks may be interrupted and continued subsequent working days.

This procedure is repeated daily as new tasks arrive. Although task arrivals occur over the 60-day evaluation period, processing continues beyond day 60 until all released tasks are completed. For a consistent comparison, we simulate this rule-based company procedure using the dataset described in Section III.A and compare its performance with our proposed approach.

C. Optimization model implementation and results

The proposed rolling-horizon MILP framework is coded in Python, and the optimization problems are solved by the Gurobi Optimizer. The baseline planning window is set to 30 days, as it covers the maximum task duration in the dataset. The effect of alternative lengths is examined in Section III.D. The computational experiment covers a 60-day evaluation period during which 118 tasks arrive, and the RH procedure is executed at 60 daily decision epochs. All 60 daily subproblems were solved to optimality. In the implementation, the baseline workload-balance parameter is set to $\Delta = 20$. Smaller values were also tested, but they led to infeasible instances. The daily worker capacity is set to $cap = 6$ based on company practice. Likewise, the off-day policy is determined according to company information: each worker may take at most eight off-days in a 30-day month, with at least two off-days before the 10th day and at least four off-days required before the 20th day.

At the start of day 1, the system is assumed to be empty, and all workers are fully available for assignment. At each daily decision epoch, the sets of completed, ongoing, and waiting tasks are updated. Assignment decisions for ongoing high-priority tasks are kept fixed, whereas started medium- and low-priority tasks may be resumed later. Newly arrived and waiting tasks are reoptimized within the 30-day planning window. The RH procedure is executed for 60 daily decision epochs. At the final decision epoch ($k = 60$), the MILP schedules the remaining open tasks within the 30-day planning window. Consequently, although task arrivals end on day 60, the resulting schedule extends beyond the evaluation period until all tasks are completed. Fig. 3 presents the final schedule for all 118 tasks,

Fig. 3. Task-worker assignments obtained by RH-MILP procedure.

with the last task completed on day 74. The Gantt chart shows that the model keeps high-priority jobs in compact blocks without any interruption, while medium and low-priority jobs provide scheduling flexibility.

The performance of the proposed model is evaluated with respect to the simulated historical company task assignment procedure described in Section III.B. We consider task completion performance and workload balance as the main performance criteria. Task-completion performance is evaluated using three metrics: mean completion day, mean flow time, and last completion day. Workload balance is evaluated using the standard deviation of cumulative workloads across workers. Additionally, we evaluate mean cumulative workload, calculated as the average cumulative workload across workers, measured in task-days. Table 4 compares the task completion performance in the actual and optimized systems using mean completion day, mean flow time, and last completion day. Overall, the optimization model particularly improves the performance of higher-priority tasks. The mean completion day decreases by 3.4% for Priority 3 tasks and by 1.8% for Priority 2 tasks, while increasing by 4.1% for Priority 1 tasks. Fig. 4 further illustrates this prioritization effect at the individual task level.

A similar pattern is observed for mean flow time, measured from task arrival to completion day. Mean flow time decreases by 8.28% for Priority 3 tasks and 4.46% for Priority 2 tasks, while increasing by 9.82% for Priority 1 tasks. Despite this trade-off, the overall mean flow time decreases from 18.08 to 17.85 days. Moreover, the last completion day of all tasks decreases from day 78 to day 74. Thus, the model improves the responsiveness of higher-priority tasks while introducing some delay for low-priority tasks, but still completes the overall task set earlier. Finally, the priority-weighted completion objective decreases from 10,644 in the actual system to 10,468 in the optimized schedule, corresponding to a 1.65% improvement.

The biggest advantage of using optimization over a rule-based policy is the fairer distribution of the workload among workers. The optimization model substantially reduces workload differences between workers. In Fig. 5, the bar plot in (a) demonstrates the cumulative workload of each worker under both the actual and the optimized schedule. In the actual system, the workload seems to be concentrated on a few workers (mainly Workers 1, 2,4) while others (Workers 8, 9, 10) are underutilized. This imbalance likely occurred because Workers 1, 2, and 4 have shorter processing times for the most frequent Type 2 tasks, which resulted in their higher utilization. This result is indeed consistent with the company's real operational conditions, where a serious workload imbalance existed among workers.

Fig. 5 further shows that in the actual system, the cumulative workload ranges from 17 to 288 task-days, while under the optimized schedule, this range narrows to 154-171 task-days, indicating a much more even distribution of workload across the workers. In addition, the box plots in Fig. 5(b) show that the standard deviation of worker cumulative workloads decreases substantially from 89.50 task-days in the actual system to 5.6 task-days in the optimized schedule. We can also observe that the workers who are underutilized in the actual system, especially Workers 8-10, are utilized better in the optimized system. The excessive workloads of Workers 1, 2, and 4 are also reduced. The improved workload balance comes with a slight increase in the total labor requirements. The average cumulative workload increases from 155.5 to 165.9 task-days, by approximately 6.7%. This occurs because the workload balance constraint prevents the model from repeatedly assigning tasks to the fastest workers, and lower priority tasks may be assigned to workers with longer processing duration. Despite this increase, Priority-3 mean flow time decreases by 8.3%, and the final task is completed four days earlier.

Fig. 6 presents the daily runtime profile of the RH-MILP procedure. The computational experiment consists of 60 daily decision epochs, each using a 30-day planning window. Thus, the procedure solves 60 overlapping 30-day subproblems. All daily subproblems were solved to optimality. The maximum daily runtime was approximately 141.67 seconds, and the total solver time was 1,107.90 seconds (18.47 minutes). These results indicate that the proposed approach is computationally suitable for daily operational use.

For comparison, all 118 tasks were also solved using a single static MILP while keeping their release-date constraints. The static model reached the best feasible priority-weighted completion value of 9,627 weighted task-days with 1% MIP gap, approximately 8% lower than the RH-MILP value of 10,468 weighted task-days. The static MILP also reduced the mean completion day from 44.45 to 40.86 and the mean flow time from 17.85 to 14.26 days. However, the RH-MILP required only 18.47 minutes in total, whereas the static MILP required approximately 1 hour and 22 minutes to reach a solution with a 1% MIP gap. Since the static model assumes knowledge of all task arrivals in advance, it is included only as a reference for comparison and is not directly implementable in the dynamic setting.

D. Sensitivity analysis

Overall, the results show that the proposed MILP-based rolling-horizon framework improves task completion performance for high-priority tasks while substantially improving workload balance. Under the baseline setting of $\Delta = 20$, the standard deviation of cumulative workload is considerably lower than in the actual system, while the mean completion day and mean flow time of Priority 3 tasks are also reduced.

In our setting, the key parameter controlling the workload balance is Δ, defined as the maximum allowable difference in cumulative workload between workers. Note that in our computational analysis we set $\Delta = 20$, the lowest feasible value, since smaller values resulted in infeasibility. In this subsection, we examine how the task completion performance changes as Δ increases, as we allow more flexibility in workload differences. To investigate this, we also test the values $\Delta = 40$ and $\Delta = 60$.

Table 4. Comparison of actual and optimized task completion performance by priority level.

Metric	Actual	Optimized	Improvement (%)
Mean completion day (all tasks)	44.68	44.45	0.5
Mean completion day (priority 3)	44.15	42.66	3.4
Mean completion day (priority 2)	45.90	45.09	1.8
Mean completion day (priority 1)	43.84	45.62	-4.1
Mean flow time (all tasks)	18.08	17.85	1.27
Mean flow time (priority 3)	17.95	16.46	8.28
Mean flow time (priority 2)	18.12	17.31	4.46
Mean flow time (priority 1)	18.16	19.94	-9.82
Last completion day (all tasks)	78	74	5.12
Last completion day (priority 3)	74	72	2.70
Last completion day (priority 2)	78	74	5.13
Last completion day (priority 1)	72	74	-2.78
Priority-weighted completion	10,644	10,468	1.65

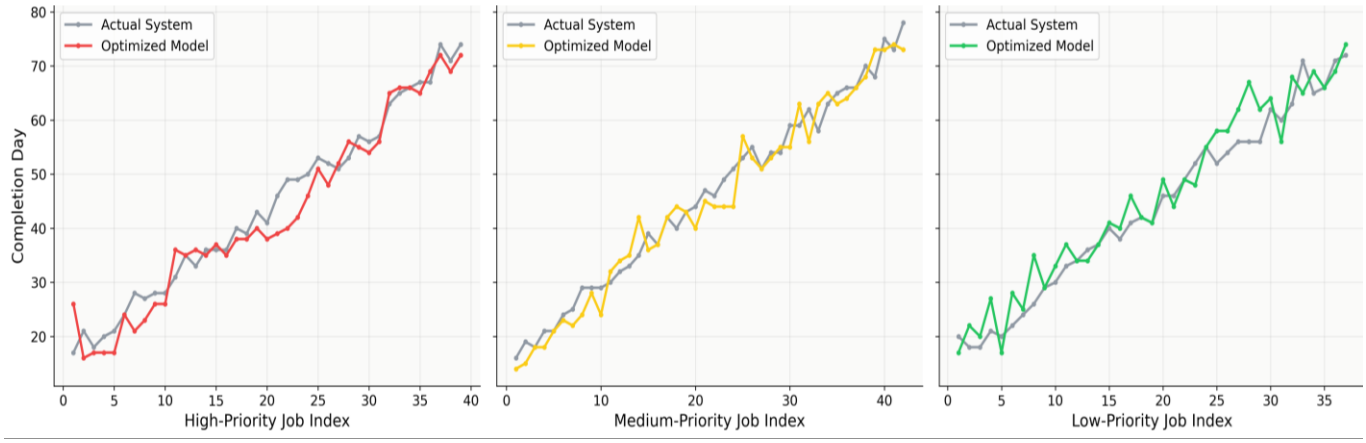


Fig. 4. Actual vs. optimized task completion days by priority level. Task indices are assigned based on the arrival time.

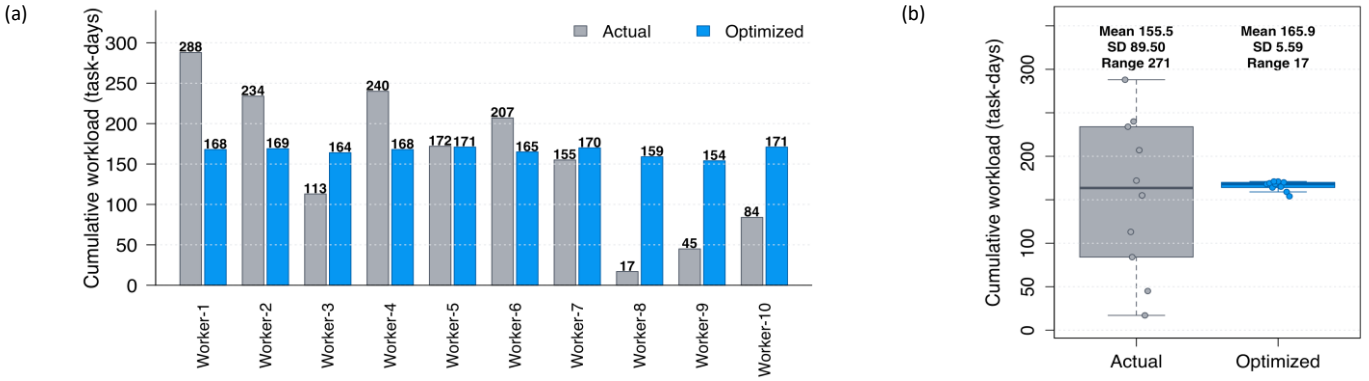


Fig. 5. Cumulative workload distribution in the actual vs. optimized system: (a) Cumulative task-days assigned to each worker; (b) Worker workload dispersion.

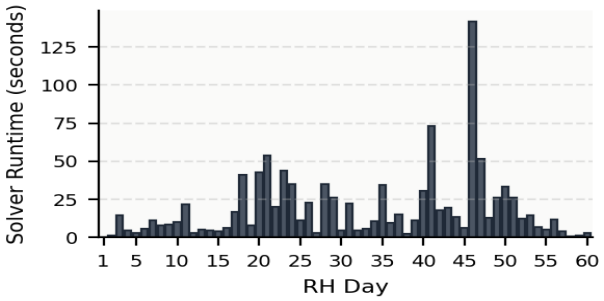


Fig. 6. Daily solver runtime across the rolling horizon.

Table 5 and Fig. 7 present the results of the sensitivity analysis. As Δ increases from 20 to 40 and 60, the standard deviation of cumulative worker workload increases from 5.59 to 14.68 and 18.65 task-days, respectively. Relaxing the workload-balance constraint leads to more workload inequality. Meanwhile, task completion performance generally improves. The overall mean flow time decreases from 17.85 days at $\Delta = 20$ to 17.41 and 16.37 days at $\Delta = 40$ and $\Delta = 60$, respectively. For Priority 3 tasks, mean flow time decreases from 16.46 to 16.36 and 15.74 days, while mean completion day decreases from 42.66 to 42.56 and 41.95 days. A similar pattern is observed for Priority 2 tasks. For Priority 1 tasks, delay was observed under $\Delta = 20$ and 40,

Table 5. Sensitivity analysis of workload balance parameter (Δ)

	Optimized System			
Key Performance Measures	**$\Delta = 20$**	**$\Delta = 40$**	**$\Delta = 60$**	**Actual System**
Mean completion day (all jobs)	44.45	44.01	42.97	44.68
Mean completion day (priority 3)	42.66	42.56	41.95	44.15
Mean completion day (priority 2)	45.09	44.62	43.40	45.90
Mean completion day (priority 1)	45.62	44.84	43.57	43.84
Mean flow time (all jobs)	17.85	17.41	16.37	18.08
Mean flow time (priority 3)	16.46	16.36	15.74	17.95
Mean flow time (priority 2)	17.31	16.83	15.62	18.12
Mean flow time (priority 1)	19.94	19.16	17.89	18.16
Last completion day (all jobs)	74	75	74	78
Last completion day (priority 3)	72	74	73	74
Last completion day (priority 2)	74	74	74	78
Last completion day (priority 1)	74	75	72	72
Cumulative Workload Mean	165.9	164.8	164	155.5
Cumulative Workload Std. Dev.	5.59	14.68	18.65	89.5

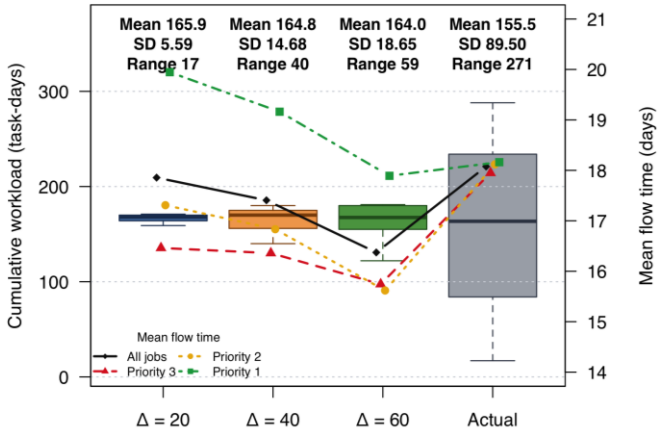


Fig. 7. Sensitivity of cumulative workload and mean flow time to the workload balance parameter (Δ).

whereas at $\Delta = 60$ both mean completion day and mean flow time are lower than in the actual system.

Overall, the sensitivity results demonstrate a clear trade-off between workload balance and task-completion performance. Smaller Δ values provide equity in workload distribution but somewhat slower completion times, while higher Δ values improve task completion performance at the expense of increased workload inequality. Therefore, selecting an appropriate Δ depends on managerial priorities: if workload equity is critical, a smaller Δ is preferable, whereas larger Δ provides better performance when faster task completion is more important.

Another sensitivity analysis was conducted for the planning-window lengths of $n = 20, 30$ and 40 days while keeping $\Delta = 20$ and all other parameters unchanged. The n=20 setting became infeasible at the fourth decision epoch. Increasing the window from 30 to 40 days did not improve overall solution quality: mean flow time increased slightly from 17.85 to 17.93 days, the priority-weighted completion value increased from 10,468 to 10,481, and the last completion day increased from 74 to 75. Moreover, total solver runtime increased from 1,107.90 to 2,843.16 seconds. Although Priority-3 mean flow time improved marginally from 16.46 to 16.41 days, this difference was operationally negligible. These findings support n=30 as the shortest feasible and more computationally efficient planning-window setting among the tested alternatives.

IV. CONCLUSION

This study proposes a dynamic task assignment framework for IT operations based on a rolling-horizon approach with a MILP model. The model considers task priorities, worker and task type dependent processing times, daily capacity limits, continuity requirements, and workload fairness. Using real operational data from a major telecommunications company in Türkiye, the proposed approach demonstrates substantial improvements over the company's existing manual assignment process within the studied 60-day evaluation period.

The computational results show a substantial improvement in workload balance, with the standard deviation of cumulative workloads reduced from 89.5 to 5.59 task-days, corresponding to a 93.7% reduction. The model also improves the completion performance of higher-priority tasks. The mean completion day and mean flow time of urgent (Priority 3) tasks decrease by 3.4% and 8.28%, respectively, while the overall mean flow time decreases by 1.27%. Moreover, the final task is completed four days earlier than in the actual system. These improvements come with a slight increase in mean cumulative workload, reflecting the additional labor requirement associated with a more balanced assignment of tasks. The results therefore demonstrate a trade-off among task completion performance, workload fairness, and total labor requirements.

The sensitivity analysis further illustrates this trade-off. Using a strict fairness setting produces a more balanced workload distribution, whereas relaxing the allowable workload difference improves task completion performance but increases workload inequality. For example, increasing the workload-balance parameter reduces overall mean flow time from 17.85 to 16.37 days, while the standard deviation of cumulative workload increases from 5.59 to 18.65 task-days. Thus, the framework allows managers to select the desired balance between fairness, completion performance, and labor requirements according to the operational priorities. Based on these results, a practical decision-support tool was developed and delivered to the company's IT department to support daily task assignment decisions.

Despite its effectiveness, several limitations exist. First, although the rolling-horizon framework updates assignments dynamically as new tasks arrive each day, the model does not anticipate future task arrivals. For example, it may be useful to keep some workers idle to accommodate high-priority tasks that are expected in the near future. This has not been considered in the current formulation. Integrating forecasting methods or stochastic arrival information could further enhance performance. Second, task processing times are defined by fixed average durations estimated from historical data. In practice, task durations may vary depending on task complexity, interruptions, or worker skills. Incorporating stochastic task durations would make the model more realistic. Third, we assume that all workers can perform all task types, which may not necessarily be true in other IT systems. This limitation can be easily handled by introducing skill-based constraints. Finally, real-time disruptions such as unexpected worker absences, interruptions after schedule release, or task cancellations during the day may affect the schedule performance. Indeed, the RH procedure can respond to such real-time disruptions, since the schedule can be re-optimized whenever new information is available. However, these disruption scenarios were not considered in the computational experiments of the current study. Examining the performance of the proposed framework under different disruption scenarios would be a valuable direction for future research.

The computational evaluation is based on a single 60-day dataset from one telecommunications company; therefore, the reported improvements should be interpreted as evidence of the effectiveness of the proposed approach in the studied operational setting rather than as generalizable results. Further validation using longer observation periods, multiple datasets, or operational scenarios would be necessary to assess the robustness and generalizability of the results.

Overall, this study contributes both methodologically and practically by demonstrating how optimization-based decision support tools can improve workload fairness, task completion performance, with only a slight increase in labor requirements in dynamic task assignment problems. The results show the strong potential of rolling-horizon MILP approaches for managing internal service operations, particularly in large-scale IT environments.

AUTHOR STATEMENT

Plagiarism Check—The article has been scanned with iThenticate and found to be compliant with the journal's plagiarism policy.
Conflict of Interest—There is no conflict of interest with any person/organization.
Ethics Committee Approval—Ethics committee approval is not required for this article.
Use of Artificial Intelligence Tools—In this study, ChatGPT-5 is used for grammar correction, minor language refinement, and to assist in preparing the graphical abstract. It was not used for content creation,

literature review, data analysis or drawing conclusions. All content reflects the original contribution of the author.

Funding—No institutional/financial support was received for this study.

Data availability—The data used in this study were provided by a company under confidentiality restrictions. Due to the proprietary and confidential nature of the data, neither the dataset nor the identity of the company can be publicly disclosed.

CRediT Author Contribution—Conceptualization, Methodology, Software, Validation, Formal Analysis, Investigation, Resources, Data Curation, Visualization, Writing – Original Draft, Writing – Review & Editing were all done by the sole author of the article.

Acknowledgment—The author would like to thank Gamze Binici and Uğur Yıldırım for their preliminary work on this topic as part of their undergraduate graduation project in the Department of Industrial Engineering at Yeditepe University, conducted under the author's supervision.

REFERENCES

[1] A. T. Ernst, H. Jiang, M. Krishnamoorthy, D. Sier, "Staff scheduling and rostering: A review of applications, methods and models", *European Journal of Operational Research*, 153(1), 3–27, 2004. https://doi.org/10.1016/S0377-2217(03)00095-X.

[2] P. De Bruecker, J. Van den Bergh, J. Beliën, E. Demeulemeester, "Workforce planning incorporating skills: State of the art", *European Journal of Operational Research*, 243(1), 1–16, 2015. https://doi.org/10.1016/j.ejor.2014.10.038.

[3] H. A. Eiselt, V. Marianov, "Employee positioning and workload allocation", Computers & Operations Research, 35(2), 513–524, 2008. https://doi.org/10.1016/j.cor.2006.03.014.

[4] Z. Liang, Y. Li, A. Lim, S. Guo, "Load balancing in project assignment", *Computers & Operations Research*, 37(12), 2248–2256, 2010. https://doi.org/10.1016/j.cor.2010.03.016.

[5] T. Lapègue, O. Bellenguez-Morineau, D. Prot, "A constraint-based approach for the shift design personnel task scheduling problem with equity", *Computers & Operations Research*, 40(10), 2450–2465, 2013. https://doi.org/10.1016/j.cor.2013.04.005.

[6] P. Smet, T. Wauters, M. Mihaylov, G. Vanden Berghe, "The shift minimisation personnel task scheduling problem: A new hybrid approach and computational insights", *Omega-International Journal of Management Science*, 46, 64–73, 2014. https://doi.org/10.1016/j.omega.2014.02.003.

[7] Z. Düzgit, A. Ö. Toy, S. Çoban, Z. Alibaşoğlu, Ö. T. Özkeskin, M. Karakaya, Y. Bayrak, "Design of job assignment and routing policies in service logistics", *Pamukkale Üniversitesi Mühendislik Bilimleri Dergisi*, 25(9), 1071–1079, 2019. https://doi.org/10.5505/pajes.2019.84658.

[8] W. Wang, K. Xie, S. Guo, W. Li, F. Xiao, Z. Liang, "A shift-based model to solve the integrated staff rostering and task assignment problem with real-world requirements", *European Journal of Operational Research*, 310(1), 360–378, 2023. https://doi.org/10.1016/j.ejor.2023.02.040.

[9] S. Jin, J. Tao, M. Lai, Q. Hu, "Scheduling multi-skill technicians and reassignable tasks in a cloud computing company", European Journal of Operational Research, 321(3), 717–733, 2025. https://doi.org/10.1016/j.ejor.2024.09.050.

[10] Ö. Karsu, M. Azizoğlu, "An exact algorithm for the minimum squared load assignment problem", *Computers & Operations Research*, 106, 76–90, 2019. https://doi.org/10.1016/j.cor.2019.02.011.

[11] Ö. Karsu, A. Morton, "Inequity averse optimization in operational research", *European Journal of Operational Research*, 245(2), 343–359, 2015. https://doi.org/10.1016/j.ejor.2015.02.035.

[12] A. Cossari, J. C. Ho, G. Paletta, A. J. Ruiz-Torres, "A new heuristic for workload balancing on identical parallel machines and a statistical perspective on the workload balancing criteria", *Computers & Operations Research*, 39(7), 1382–1393, 2012. https://doi.org/10.1016/j.cor.2011.08.007.

[13] R. Stolletz, J. O. Brunner, "Fair optimization of fortnightly physician schedules with flexible shifts", *European Journal of Operational Research*, 219(3), 622–629, 2012. https://doi.org/10.1016/j.ejor.2011.10.038.

[14] L. Huang, Y. Gao, F. Xiao, "Balancing workload fairness in task assignment: modeling via piecewise linear approximation", *Applied Sciences*, 16(4), 1747, 2026. https://doi.org/10.3390/app16041747.

[15] X. Chen, B. W. Thomas, M. Hewitt, "Multi-period technician scheduling with experience-based service times and stochastic customers", *Computers & Operations Research*, 82, 1–14, 2017. https://doi.org/10.1016/j.cor.2016.12.026.

[16] X. Chen, M. Hewitt, B. W. Thomas, "An approximate dynamic programming method for the multi-period technician scheduling problem with experience-based service times and stochastic customers", *International Journal of Production Economics*, 196, 122–134, 2018. https://doi.org/10.1016/j.ijpe.2017.10.028.

[17] L. M. Annear, R. Akhavan-Tabatabaei, V. Schmid, "Dynamic assignment of a multi-skilled workforce in job shops: an approximate dynamic programming approach", *European Journal of Operational Research*, 306(3), 1109–1125, 2023. https://doi.org/10.1016/j.ejor.2022.08.049.

[18] K. Wang, Y. Xiang, W. Chen, "Online task assignment in warehouse considering the pickers' dynamic capability: A data-driven optimization approach", *Transportation Research Part E: Logistics and Transportation Review*, 214, 105040, 2026. https://doi.org/10.1016/j.tre.2026.105040.

[19] R. Stolletz, E. Zamorano, "A rolling planning horizon heuristic for scheduling agents with different qualifications", *Transportation Research Part E: Logistics and Transportation Review*, 68, 39–52, 2014. https://doi.org/10.1016/j.tre.2014.05.002.

[20] I. Tarhan, K. G. Zografos, J. Sutanto, A. Kheiri, H. Suhartanto, "A multi-objective rolling horizon personnel routing and scheduling approach for natural disasters", *Transportation Research Part C: Emerging Technologies*, 149, 104029, 2023. https://doi.org/10.1016/j.trc.2023.104029.