Improving Operational Performance in Service Delivery Organizations by Using a Metaheuristic Task Allocation Algorithm

Rahul Paul[1]*, R. P. Jagadeesh Chandra Bose[2], Stephan K. Chalup[1], and Gurulingesh Raravi[2]

[1] The University of Newcastle, Callaghan, Australia
[2] Conduent Labs, India
Rahul.Paul@uon.edu.au, Jagadeesh.Prabhakara@conduent.com, Stephan.Chalup@newcastle.edu.au, gurulingesh@gmail.com

Abstract. Efficient allocation of tasks to employees is crucial in service delivery organizations. It facilitates meeting the service level agreements (SLAs), utilizes the employees well and improves the operational performance. Task allocation is a challenging problem that addresses the inter-dynamics of tasks and employees, considering factors such as diversity, utilization, skills and fairness. The combination of task deadlines and associated SLA requirements adds another dimension to the complexity of the problem. In this paper, we propose a tabu search algorithm for efficient task allocation. The algorithm is employee utilization, productivity and fairness aware. We evaluate the proposed algorithm using real world transaction processing data from a large service delivery organization. The results show that the proposed approach can reduce deadline misses substantially in comparison to the organization's current approach.

Keywords: Operational Excellence, Transaction Processing, Task Allocation, Deadline Violation, Tabu-search, Meta heuristic

1 Introduction

Service delivery organizations (SDOs) provide operations support, sales services and information technology services like transaction processing. It is not uncommon for organizations to handle large volumes of transactions (of the order of a few hundred thousands) every day. In addition to the volume, organizations often face the issue of inter-arrival rate of the task and different types of transactions that they handle. Furthermore, organizations have to meet SLAs agreed upon with clients and are always required to perform better in terms of optimal cost and their operational efficiency. An example SLA could be that 98% of transactions should be completed within one hour (i.e., the *turnaround time* of transactions should be less than one hour for at least 98% of transactions).

* M. Brambilla, T. Hildebrandt (Eds.): BPMN 2017 Industrial Track Proceedings, CEUR-WS.org, 2017. Copyright 2017 for the individual papers by the papers' authors. Copying permitted for private and academic purposes. This volume is published and copyrighted by its editors.

Apart from SLAs, organizations typically define operational *key performance indicators* (KPIs) to monitor an organization's performance and its way of working. Processing time of transactions (i.e., the actual amount of time spent in processing a transaction), resource utilization and productivity are a few examples of KPIs [1]. Client-level SLAs are often translated into some operational KPIs to keep a close track on an organization's performance. For example, turnaround time of each transaction is broken down into expected processing times (referred to as *baselines*) for individual activities/tasks involved in the execution of the transaction. Once baselines are defined for each task, the organization can keep track of the *number of baseline violations* and the *magnitude by which these baselines are violated*[3] and take corrective actions (if need be) before it percolates to a SLA violation (in terms of turnaround time).

Task allocation is one of the key planning exercises that plays a significant role in an organization's quest to meet SLAs and to achieve operational excellence. Employees within an organization assume roles based on their skills, proficiency, and experience. Since the execution of tasks requires specific skills, tasks need to be assigned to employees (referred to as *resources*) with appropriate skills/proficiency. Two resources possessing same skills but having different proficiencies in those skills may take different times to process a task. In spite of all these intricacies, most SDOs still resort to manual allocation of tasks.

Such a manual allocation in SDOs is generally done by a team lead. A team lead handles the incoming volume of transactions and depending on the type of transaction and the tasks that need to be executed to process it, manually allocates the tasks to resources that she manages. While doing so, the team lead needs to consider a multitude of factors such as task complexity, skill requirements, baseline processing times and resource skills/proficiency, workload, utilization, fairness, etc. All of these factors need to be mentally processed making the problem of task allocation extremely challenging for the team leads and making it almost impossible to keep in check the above mentioned factors and thereby has a higher risk of impacting the KPIs and SLAs. Manual task allocation further imposes a danger of inherent biases that team leads may adopt. Such biases will impact resources, whose incentive payouts are dependent on the tasks that they process and their productivity. These factors warrant for the design of efficient and automated task allocation schemes for SDOs to achieve operational excellence via fair task allocation, by better utilizing the resources, improving their productivity, reducing the costs and improving the operational KPIs thereby meeting the SLAs.

Problem Description. In this paper, we address the problem of efficiently allocating tasks to resources in a service delivery organisation with the objective of allocating tasks in a real time environment. We try to get final allocations within certain optimality gap. We also consider several other constraints like the number of baseline violations is within a specified limit (KPI/SLA-aware), assigned workload is fair among the resources, utilization of each resource should be within the specified lower and upper bounds to ensure that all the resources are well utilized (utilization-aware) and, productivity of each resource is within a specified upper limit as described in [1].

[3] Typically, SLAs are defined such that the penalty for violations vary based on the magnitude of violation.

Contributions of this work. The main contribution of this paper is: Although [1] has provided an ILP based solution for the problem, ILP based solution takes much more time as the number of resources and number of tasks increases. For example, the ILP based approach doesn't provide any feasible solution (within 2 hours) even for a scenario where we have 22 resources and 2000 tasks. As service delivery organisations need the allocation matrix for the resources in (near-)real time, ILP based solution cannot be deployed in practice. Therefore, we propose a meta-heuristic based approach in this paper. It involves two steps:

1. We convert the problem to meta-heuristic based problem in which tabu search algorithms have been proven to be effective and fast.
2. We propose a tabu-search (TS) based algorithm.

In our evaluations, the proposed approach is able to reduce the number of violations by 20% and the magnitude of violations by up to 75%, furthermore it increases the productivity of resources by 10% and average utilization and workload has been reduced by 35% compared to the currently practiced manual task allocation in the organization.

Organization of the paper. This paper is organized as follows. Section 3 presents the notations and the system model. Our task allocation approach based on Tabu search is presented in Section 4. Section 6 presents the experimental setup and discusses the efficacy of the proposed solution when applied to real-world data from a large SDO. Finally, Section 7 concludes the paper.

2 Related work

Task allocation problem is well studied in the past [2–9] with a focus on assigning tasks to machines (*non-human* resources). These works can not be applied to the problem under consideration since they cannot handle some of the human related aspects such as productivity and utilization of employees.

Some of the other works [10–14] have specifically looked into the problem of allocation of tasks to employees. [11] studied the problem of assigning employees to shifts based on the estimate of the skills required for the jobs in those shifts. [10] discuss a class of personnel task scheduling problems (that arise in rostering applications), mostly related to scheduling of employees in different shifts and assigning different tasks to a set of shifts. [12] propose a hybrid-heuristic approach in which the objective is to assign the tasks such that minimum number of resources are used, a problem clearly different from ours. The approach considers skills of employees and the demand while recommending the reallocation. The allocation is done considering the availability of employees, skills of employees and skills required to perform a task and start and end date for the task. Some of the other works [13, 14] study the problem with fairness objective while ensuring the regulatory requirements. These works do not consider the effect of allocation on operational KPIs and also the key human factors such as employees productivity and utilization. Thus, none of these works can be directly applied to the problem under consideration in this work.

Several mechanisms have been proposed in [15–20] which perform allocation of task to employees that are skill- and fairness-aware. Many of these techniques [15–18] are not utilization, productivity- and SLA-aware (e.g., minimizing the number and magnitude of baseline time violations). In [19], an engine configured to determine an assignment of a resource to a task considering the task and the resource profile is proposed. It takes into account the resource utilization. However, it is not productivity-aware and since there is no deadline for tasks, its objective is different than ours. In [20], a constraint programming based allocation technique is proposed. It does skill based allocation and also considers fairness and tries to minimize the cost. However, the allocation strategy is not productivity-aware, utilization-aware and does not focus on minimizing the deadline violations.

3 System model

We consider the problem of allocating a set $\Pi = \{\pi_1, \pi_2, \ldots, \pi_n\}$ of n tasks to a set $R = \{r_1, r_2, \ldots, r_m\}$ of m employees (referred to as *resources*) in a services delivery setting. Each task π_j is characterized by a processing time and a *baseline time* b_j (also referred to as *deadline*). The processing time of a task depends on the resource that processes it and the processing time of π_j when processed by r_i is denoted by t_{ij}, where $j \in \{1, 2, \ldots, n\}$ and $i \in \{1, 2, \ldots, m\}$. A task π_j that cannot be processed by a resource r_i (e.g., if the resource does not have the skills to process it) is modeled by setting t_{ij} to ∞. Ideally, processing of each task π_j has to be completed within b_j time units.

For each resource π_i, we define the following terms. The workload w_i of π_i is defined as:

$$w_i \overset{\text{def}}{=} \sum_{\pi_j \in \Pi(i)} t_{ij} \tag{1}$$

where $\Pi(i)$ is the set of tasks assigned to r_i. Informally, it is the sum of the processing times that the resource takes for each task (t_{ij}) that is assigned to it. The utilization u_i of π_i is defined as:

$$u_i \overset{\text{def}}{=} \frac{w_i}{s} \tag{2}$$

where s is the duration of the shift in which π_i works. Informally, utilization of a resource indicates the fraction of time the resource will be occupied with the assigned tasks in a shift. The productivity p_i of π_i is defined as:

$$p_i \overset{\text{def}}{=} \frac{\sum_{\pi_j \in \Pi(i)} b_j}{\sum_{\pi_j \in \Pi(i)} t_{ij}} \tag{3}$$

Informally, it is the ratio of the amount of expected time in which π_i needs to complete all the tasks assigned to her to the amount of actual time in which π_i will complete those tasks.

We also use a few other below mentioned notations in the paper. Upper limit on the productivity of any resource is denoted by $\bar{p}$ and upper limit on the number of baseline violations is denoted by $\bar{v}$. The lower and upper bounds on the utilization of any resource is denoted by $\underline{u}$ and $\bar{u}$, respectively. (*Bar* is used over a letter to denote the upper bound and *underbar* is used to denote the lower bound.) The average workload

$\tilde{w}$ is defined as: $\tilde{w} = \frac{1}{m}\sum_{i=1}^{m} w_i$. The lower and upper bounds on the workload of a resource are $(1 - \bar{w}) \times \tilde{w}$ and $(1 + \bar{w}) \times \tilde{w}$, respectively.

4 Tabu Search

Tabu search is generally implemented as a single search trajectory direct search method. The concept was originally proposed by Glover [21] and since than this research technique has been used in many applications. Tabu search has been applied to discrete combinatorial optimization problems such as graph coloring and Travelling Salesman. Tabu search is initiated at a feasible starting point within a solution. After that, it identifies sequences of moves and whilst that process is executed, a candidate list is generated. Evaluation process can determine whether the member belongs to the list or not. Tabu search has three main advantages [21] : (1) the use of flexible attribute based design to permit better evaluation criteria and historical search information to exploit the problem more thoroughly; (2) an associated mechanism of control based on the interplay between conditions that constrain and free the search process; (3) Intensification strategies reinforce move combinations and solution features historically found good.

We use the tabu search approach to refine the solution obtained by using the given heuristic algorithm. Although a tabu search based algorithm is proposed in [22] for the Orienteering problem, our algorithm is different from it. Our algorithm works for the resource allocation matrix with the score of each resource, which is different from the graph discussed in [22]. In [23] TS-based algorithm is used for wireless relay network where they simplified the problem as orienteering problem. Then they apply TS algorithm to get best path from the current with a penalty funtion of budget and cost of path. Furthermore, our algorithm only cares about the last assignment for each resource in the allocation matrix. Thus, the proposed algorithm is more suitable for our problem.

5 Tabu search based task allocation

According to tabu search we need a first solution to start with. In our method, we have started with the allocation given from the savings of the resources. Saving of a resource can be calculated by :

$$savings_i = b_j - t_{ij} \tag{4}$$

where b_j is the baseline of the task and t_{ij} is the timeline of task j assigned to resource i. This t_{ij} has been taken from random distribution of same task perform by various resources. We calculate the savings of resources and which ever resource has the largest saving and the resource which can able to handle same type of work, we allocate the task to that resource in the first step. But, if the savings of all the resources are negative i.e. $t_{ij} \geq b_j$, we assign the task to the resource which has less negative savings. Then we check for the performance for the initial allocation. We calculate the utilization, productivity, workload and number of violation from the allocation matrix $\mathbb{X}$. The calculation of these vectors has been discussed in the equations (1), (2), and (3). After that we make a list of all violated tasks to perform our TS-based algorithm.

Algorithm 1 Tabu

```
1: function TABU($R, \mathit{\Pi}, t, b, s$)    ▷ Where R - Resource array, $\mathit{\Pi}$ - Set of Task, t - Timeline matrix, b - Baseline matrix, s - Shift time ▷ i - Resource index, j - Task index, $\mathbb{X}$ - Allocation matrix
2:     Initialize $\mathbb{X}$
3:     Call PERFORMANCEEVAL($\mathbb{X}, t, b$)
4:     do
5:         if $\mathbb{X}_{ij} == 1$ and $t_{ij} > b_j$ then
6:             $V\_task.append(j)$
7:         end if
8:         for each $j$ in $V\_task$ do
9:             for each $i$ in $R$ do
10:                if $t_{ij} \neq -1$ then
11:                    if $U_i \leq \bar{u}\, and\, P_i \leq \bar{p}\, and\, W_i \leq \bar{w}$ then
12:                        $Tabu\_list_j.append(i)$
13:                    end if
14:                end if
15:            end for
16:        end for
17:        $\mathbb{X}, allocation\_update$ from TABUSEARCH($Tabu\_list, \mathbb{X}, t, b$)
18:    while $allocation\_update \neq 0$
19:    return $\mathbb{X}$
20: end function
```

To validate or make the allocation close to the optimal solution we work with only violated tasks in our TS-based algorithm. Take each violated task from the first allocation $\mathbb{X}$ and create a candidate list of resources which can do that task. Before adding the resource to the candidate list we need to check the current utilization, productivity and workload of the resource. The current values of these metrics should be less than $\bar{u}$, $\bar{p}$, $\bar{w}$ which are user driven input to the system. Then only the resource will be added to the current set of resources to perform the violated task. Then we call the tabu search algorithm with the tabu list and current allocation $\mathbb{X}$. Before going into the Tabusearch algorithm we need to evaluate a score to choose the best candidate from the $Tabu\_list$. A score can be evaluated by the summation of workload and productivity with the penalty of the solution, i.e.,

$$score_i = (norm(p_i) + norm(w_i)) + pen_i \tag{5}$$

where pen_i is the penalty associated with the solution, and it is given by

$$pen_i = \begin{cases} 0, & \text{if } b_j \leq r_i \\ \phi * (r_i - b_j), & \text{if } b_j > r_i \end{cases} \tag{6}$$

Where r_i is the remaining time of the resource from its shift. And ϕ is a penalty parameter dynamically updated during the search. The parameter ϕ is initialized at a value of 1. But the algorithm is robust with respect to this parameter.

Now, we call the TS algorithm with the Tabu list. First we generate the score for each resources which are present in the tabu list for particular task. Then we select the

Algorithm 2 Tabu Search

```
1: function TabuSearch($Tabu\_list, \mathbb{X}, t, b$)    ▷ Where $Tabu\_list$ - Set of candidate list, $\mathbb{X}$ - Allocation matrix ▷ i - Resource index, j - Task index, $score_i$ - score of resource from Eq. 5
2:     $allocation\_update = 0$
3:     for $iteration = 1$ to $max_iterations$ do
4:         $tabu\_update \leftarrow 0$
5:         $\phi \leftarrow 1$
6:         for each $j$ in $Tabu\_list$ do
7:             $\mathbb{X}' \leftarrow \mathbb{X}$
8:             select $i^{new} \in Tabu\_list[j]$ with the best $score_{i^{new}} - score_{i^{cur}}$
9:             $\mathbb{X}'_{i^{cur}j} \leftarrow 0$
10:            $\mathbb{X}'_{i^{new}j} \leftarrow 1$
11:            Call PerformanceEval($\mathbb{X}', t, b$)
12:            Call PerformanceEval($\mathbb{X}, t, b$)
13:            if $magV' \leq magV$ then
14:                $\mathbb{X} \leftarrow \mathbb{X}'$
15:                $\phi \leftarrow \phi/2$
16:                $allocation\_update \leftarrow 1$
17:                $tabu\_update \leftarrow 1$
18:            end if
19:        end for
20:        if $tabu\_update == 0$ then
21:            $\phi \leftarrow \phi * 2$
22:        end if
23:    end for
24:    return $\mathbb{X}, allocation\_update$
25: end function
```

best resource for the task and assign the task to that perticular resource. We fix the tabu list elements, which are obtained from the previous function and will be updated in the score function if assignment is being done. We update the $\mathbb{X}$ with new selected assignment. Then again we call the PerformanceEval function to check the obtain solution (Magnitude of violation) is better or not. If yes, we keep the updated assignment and update the W, P, U. Other wise we go to the next violated task. After we loop through all the task which are there in the violated task list, if we get any updated element in $\mathbb{X}$, we repeat the whole experiment and try to improve our result iteratively. Thus, we approach for the optimal solution.

6 Experimental Results and Discussion

The proposed task allocation mechanism has been implemented, and in this section, we discuss the results of applying our approach on real-world data from a transaction processing business unit within a large SDO. The resources in the organization are skilled on processes. Different resources are skilled to execute different processes and the proficiency levels of resources for the skills can vary. Each team lead is responsible for handling transactions (of various types) pertaining to certain processes. The team lead monitors the volume of transactions arriving every day (that he/she is expected to handle) and allocates them to resources in his/her team based on their skills/proficiency and the complexity of the transactions. The team lead manages all these mentally day-in and day-out. Each transaction type comes with a *baseline* (unit of time) within which transaction instances of that type is expected to be completed. Any transaction extending beyond the baseline is considered to be violating the baseline KPI. We keep track of this KPI via the *number of violations* metric, which measure the number of transactions that violated the baseline. We also keep track of the *magnitude of violation*, the magnitude of time by which the baseline is violated. For example, if the baseline of a transaction type is 100 time units and if an instance of that transaction type took 125 time units, then the magnitude of violation is 25 time units. It is important to keep track of this because the penalty that the organization needs to pay also depends on the magnitude of violation.

We have applied the proposed approach on the transactions handled by several team leads to assess the efficacy of our approach. We present the results on the transactions handled by four team leads, TL_1, TL_2, TL_3, and TL_4 for a period of six months, January to June 2016. In order to study the efficacy of our approach, we need to estimate the likely number of violations and the magnitude of violations if the transactions are assigned to resources. To enable this, we extracted the *historical processing time distributions* for each of the employees for the various transaction types. For any transaction assigned to a resource, we randomly sample[4] the time from that resources' processing time distribution for the transaction type (pertaining to that transaction). Before taking randomly sampled processing time from the distribution, we also discard the outliers values. That means we discard the processing times which are much less than that of the baseline (e.g., 25% of baseline). We are doing this because these outlier values can affect the productivity and utilization from the standard values. Assuming that the resource would take the sampled time had he/she been assigned that transaction, we check if that time exceeds the baseline. If so, we record that as a violation and capture the magnitude by which it exceeds the baseline. For each transaction, we do this assignment *five* times and take the average metrics along with their confidence intervals.

The initial assignment for tabu search is done using a simple heuristic. For each transaction from Π, check and assign the transaction to that resource who has the largest saving for that transaction where saving is defined by :

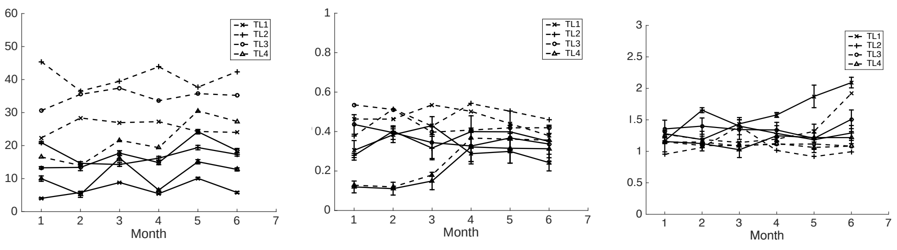
$$savings_{ij} = b_j - t_{ij} \tag{7}$$

Here b_j signifies the baseline time for the transaction and t_{ij} denotes the sampled time for that transaction for resource i. If all the resources have negative savings for a particular task we assign the task to the least negative savings. The initial allocation matrix X, obtained thus, is then subjected for tabu search as explained in Section 4.

Table 1 presents the characteristics of the transactions handled by different team leads and compares the current manual approach w.r.t. our proposed automated task allocation. We can see that the current approach leads to significant number of violations

[4] Other sampling mechanisms such as weighted sampling, where more weights to sampling processing times from the recent past is applied. A detailed analysis and discussion of different sampling mechanisms is beyond the scope of this paper.

for the team leads, the percentage of violations being in the range of 14% to 46%. Using our proposed tabu allocation, the deadline violations are in the range between 2% and 24%. On average, the number of violations is improved by 60% across all the team leads. Fig. 1 depicts the comparison of the proposed approach w.r.t. the current approach. Fig. 1(a) depicts the percentage of violations for the current approach (dotted lines) for the four team leads and the average percentage of violations along with the 95% confidence intervals (over five runs) for the proposed tabu-based allocation (solid lines). Similarly Fig. 1(b) and Fig. 1(c) depict the average utilization and productivity for the current approach (dotted lines) and the average utilization and productivity along with the 95% confidence intervals (over five runs) for the tabu-based allocation (solid lines) respectively. As we can see our approach outperforms the current approach. The percentage of violations is consistently much smaller than the current approach. Also, the utilization of resources is smaller than that of the current approach. In other words, due to the new allocation, the current resources are able to finish the tasks much earlier. These resources can be better utilized by assigning them other tasks or reallocating them to other groups. Furthermore, the productivity of resources is higher than that of the current approach.



(a) percentage of violation (b) utilization (c) productivity

Fig. 1. Comparison of percentage of violations, average utilization and productivity metrics for the four team leads for both the current approach and the proposed tabu-based approach. Solid line in the figures correspond to the tabu-based approach while the dotted line represents the current method.

In our evaluations, the proposed approach is able to reduce the number of violations by 20% and the magnitude of violations by up to 75%, furthermore it increases the productivity of resources by 10% and the average workload and utilization has been reduced by 35% compared to the currently practiced manual task allocation in the organization.

Our proposed approach is computationally tractable with running times of less than 3 minutes. Fig 2 shows the comparison of time taken in ILP and our approach for one team lead TL1 with one month data. Each day team lead encounters different number of transaction. For example, from graph (Fig. 2) shows that on 22^{nd} day we have 24 resources with 200 transaction, our tabu-search approach takes only 6.5 seconds with 98% confidence interval in percentage in violation. Another example, for a team lead which has 1006 tasks in his pool and 20 resources. According to Fig. 2, ILP takes more

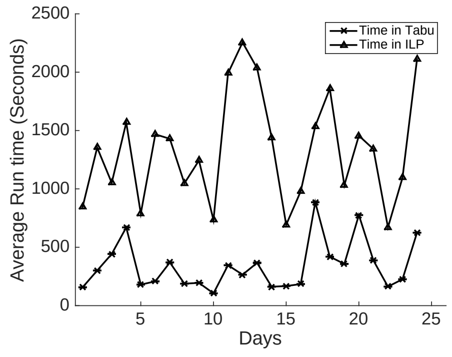
Table 1. Comparison of the number of violations and their magnitudes for the transactions handled by four team leads of a large service organization for the duration of six months in the year 2016, where tabu based allocation gives us less violation than the current aloocation.

TL$_1$	**# Res.**	**# Trans.**	**Current allocation**				**Tabu-based allocation**			
			% Viol.	**Cum. Mag. Viol. (in secs)**	**Avg. Util.**	**Avg. Prod.**	**Avg. % Viol.**	**Cum. Mag. Viol. (in secs)**	**Avg. Util.**	**Avg. Prod.**
Jan	22	308	16.55	23154	12.79	114.72	10.00	11132	11.93	115.59
Feb		100	14.00	11253	11.94	114.73	5.00	3077	11.05	112.30
Mar		111	21.62	62229	18.00	108.34	16.21	11064	14.94	102.50
Apr		1620	19.38	140358	36.69	112.34	6.54	46075	32.19	124.70
May		2700	30.52	324849	36.17	106.14	15.11	134256	31.53	121.83
Jun		2346	27.32	234297	35.67	108.31	12.79	109459	31.30	121.80
TL$_2$										
Jan	17	5742	22.34	563531	46.35	127.21	3.03	79614	30.48	193.98
Feb		4782	28.37	572090	46.30	118.75	2.80	83481	28.36	186.89
Mar		6072	26.96	634317	53.54	114.03	2.81	120633	30.10	233.67
Apr		6647	27.26	732574	50.25	119.36	2.45	92278	28.78	207.77
May		6493	24.29	457616	44.01	131.65	10.08	121145	29.94	187.24
Jun		5164	23.99	424336	37.88	192.02	5.80	53523	24.27	209.18
TL$_3$										
Jan	32	3737	30.61	985369	53.54	115.17	13.28	429000	43.59	135.58
Feb		2888	35.53	854770	51.17	109.16	13.40	241739	39.45	140.03
Mar		2116	37.43	532662	39.69	109.27	17.72	245589	34.40	134.52
Apr		1825	33.60	607748	40.77	111.12	14.85	151599	32.68	133.90
May		1933	35.75	674532	35.75	111.61	24.21	355669	36.76	120.57
Jun		2198	35.21	935117	41.93	108.25	18.47	292266	33.69	150.83
TL$_4$										
Jan	22	1731	45.40	671828	37.68	95.29	20.92	226034	27.90	116.73
Feb		1824	36.51	548773	51.46	105.99	14.65	194856	40.20	165.33
Mar		1317	39.48	490642	42.75	140.01	14.27	146027	31.63	140.01
Apr		1488	43.88	470765	54.15	101.34	16.20	144945	39.97	128.46
May		2979	37.70	939586	50.31	91.58	19.40	342126	39.98	118.40
Jun		4560	42.40	793910	46.14	99.30	17.34	297356	34.98	129.12

than 1 hour to assign tasks to the resources but with our approach the assignment takes less than 10 minutes. This proves our second claim of this paper.

7 Conclusions

In this paper, we studied the problem of automated allocation of tasks to human resources in a service delivery organizational setting with a larger goal of improving the operational key performance indicators. For this problem, we proposed a metaheuristic based task allocation scheme which aims to minimize the number of tasks missing the deadlines and to minimize the magnitude by which they are missed taking into consideration the resource skills, utilization, productivity, fairness and KPIs while allocating the tasks. In the experimental evaluations on transaction data of a large services organization, the proposed approach reduced the number of deadline misses and the magnitude of violations by 75%. As future work, we intend to design an algorithm that dynamically reallocates tasks periodically by closely monitoring the realtime performance of



(a) running time comparison

Fig. 2. Comparison of time taken for ILP and the proposed tabu-search approach.

resources. Also, we can use various sampling strategies selecting the processing times for a particular task instead of a random distribution.

8 Acknowledgement

The first author was supported by a UNRSC50:50 PhD scholarship at the University of Newcastle, Australia.

References

1. A. Mulla, G. Raravi, T. Rajasubramaniam, R. P. J. C. Bose, and K. Dasgupta. Efficient task allocation in services delivery organizations. In *2016 IEEE International Conference on Services Computing (SCC)*, pages 555–562, June 2016.
2. Camille C Price. Task allocation in distributed systems: A survey of practical strategies. In *Proceedings of the ACM'82 conference*, pages 176–181. ACM, 1982.
3. Ronald L Graham, Eugene L Lawler, Jan Karel Lenstra, and AHG Rinnooy Kan. Optimization and approximation in deterministic sequencing and scheduling: a survey. *Annals of discrete mathematics*, 5:287–326, 1979.
4. Ethel Mokotoff. Parallel machine scheduling problems: a survey. *Asia-Pacific Journal of Operational Research*, 18(2):193, 2001.
5. Eugene L Lawler, Jan Karel Lenstra, and AHG Rinnooy Kan. Recent developments in deterministic sequencing and scheduling: a survey. In A. H. G. Rinnooy Kan M. A. H. Dempster, J. K. Lenstra, editor, *Deterministic and stochastic scheduling*, pages 35–73. Springer, 1982.
6. Edward G Coffman Jr, Michael R Garey, and David S Johnson. Approximation algorithms for bin-packingan updated survey. In *Algorithm design for computer system design*, pages 49–106. Springer, 1984.
7. TCE Cheng and CCS Sin. A state-of-the-art review of parallel-machine scheduling research. *European Journal of Operational Research*, 47(3):271–292, 1990.
8. Kathryn A Dowsland and William B Dowsland. Packing problems. *European Journal of Operational Research*, 56(1):2–14, 1992.

9. Andreas Wiese, Vincenzo Bonifaci, and Sanjoy Baruah. Partitioned edf scheduling on a few types of unrelated multiprocessors. *Real-Time Systems*, 49(2):219–238, 2013.
10. Mohan Krishnamoorthy and Andreas T Ernst. The personnel task scheduling problem. In *Optimization methods and applications*, pages 343–368. Springer, 2001.
11. Andreas T Ernst, Houyuan Jiang, Mohan Krishnamoorthy, and David Sier. Staff scheduling and rostering: A review of applications, methods and models. *European journal of operational research*, 153(1):3–27, 2004.
12. Pieter Smet, Tony Wauters, and Greet Vanden Berghe. Hardness analysis and a new approach for the shift minimisation personnel task scheduling problem. In *Evolutionary Computation, 2002. CEC'02. Proceedings of the 2002 Congress on*, volume 1. the 27th Annual Conference of the Belgian Operations Research Society (ORBEL), 2013.
13. Tanguy Lapègue, Odile Bellenguez-Morineau, and Damien Prot. Methods for the shift design and personnel task scheduling problem. In *MOSIM 2014, 10ème Conférence Francophone de Modélisation, Optimisation et Simulation*, 2014.
14. Damien Prot, Tanguy Lapegue, and Odile Bellenguez-Morineau. A two-phase method for the shift design and personnel task scheduling problem with equity objective. *International Journal of Production Research*, 0:1–13, 2015.
15. Timothy Su. Four-dimensional resource allocation system, October 7 2002. US Patent App. 10/065,341.
16. G Edward Powell, Mark T Lane, and Runar Indseth. Method and system for allocating personnel and resources to efficiently complete diverse work assignments, April 19 1999. US Patent App. 09/294,251.
17. Debasis Mitra, John A Morrison, and Kajamalai Gopalaswamy Ramakrishnan. Method for resource allocation and routing in multi-service virtual private networks, December 18 2001. US Patent 6,331,986.
18. Randall K Fields, Paul R Quinn, and Todd Blackley. System and method for making staff schedules as a function of available resources as well as employee skill level, availability and priority, May 5 1992. US Patent 5,111,391.
19. Brian Fletcher, Robert Noel William Laithwaite, Evgeny Selensky, and Paul Sexton. Optimizing resource assignment, October 30 2012. US Patent App. 13/664,338.
20. Youssef Hamadi and Claude-Guy Quimper. Allocating resources to tasks in workflows, January 30 2007. US Patent App. 11/669,098.
21. Fred Glover. Tabu search: A tutorial. *Interfaces*, 20(4):74–94, 1990.
22. Yun-Chia Liang, Sadan Kulturel-Konak, and Alice E Smith. Meta heuristics for the orienteering problem. In *Evolutionary Computation, 2002. CEC'02. Proceedings of the 2002 Congress on*, volume 1, pages 384–389. IEEE, 2002.
23. Hao Zhou, Yusheng Ji, and Baohua Zhao. Tabu-search-based metaheuristic resource-allocation algorithm for svc multicast over wireless relay networks. *IEEE Transactions on Vehicular Technology*, 64(1):236–247, 2015.