# A Framework for Recommending Resource Allocation Based on Process Mining

Michael Arias(✉), Eric Rojas, Jorge Munoz-Gama, and Marcos Sepúlveda

Computer Science Department, School of Engineering,
Pontificia Universidad Católica de Chile, Santiago, Chile
{m.arias,eric.rojas}@uc.cl, {jmun,marcos}@ing.puc.cl

**Abstract.** Dynamically allocating the most appropriate resource to execute the different activities of a business process is an important challenge in business process management. An ineffective allocation may lead to an inadequate resources usage, higher costs, or a poor process performance. Different approaches have been used to solve this challenge: data mining techniques, probabilistic allocation, or even manual allocation. However, there is a need for methods that support resource allocation based on multi-factor criteria. We propose a framework for recommending resource allocation based on Process Mining that does the recommendation at sub-process level, instead of activity-level. We introduce a resource process cube that provides a flexible, extensible and fine-grained mechanism to abstract historical information about past process executions. Then, several metrics are computed considering different criteria to obtain a final recommendation ranking based on the BPA algorithm. The approach is applied to a help desk scenario to demonstrate its usefulness.

**Keywords:** Resource allocation · Process mining · Business processes · Recommendation systems · Organizational perspective · Time perspective

## 1 Introduction

Dynamic resource allocation is an important and challenging issue within business process management [8,20]. It can contribute significantly to the quality and efficiency of business processes, improve productivity, balance resource usage, and reduce execution costs. This article describes a framework that supports resource allocation based on multi-factor criteria, considering both resources capabilities, past performance, and resources workload.

An initial strategy is to assign to a given activity a resource whose profile is closest to the profile required by the activity. However, this strategy does not consider the current workload of the resource or how successful the resource has been performing similar tasks in the past. To fill this gap, it is possible to take advantage of historical information stored by today's information systems about business processes execution, knowing who executed what activity, when,

M. Reichert and H.A. Reijers (Eds.): BPM Workshops 2015, LNBIP 256, pp. 458–470, 2016.
DOI: 10.1007/978-3-319-42887-1_37

and how long it took. Moreover, recently it has been proposed to use historical information stored in event logs to improve resource allocation using process mining techniques [20].

Different mechanisms have been proposed to allocate resources to activities [6,8–12,16,18]. In [18], several workflow resource patters are identified. For example, three allocation types defined are: capability-based allocation, history-based allocation and role-based allocation. Capability-based allocation provides a mechanism for allocating a resource to an activity through matching specific requirements for an activity with the capabilities of the potential range of resources that are available to undertake it. History-based allocation involves the use of information on the previous execution history of resources when determining which of them to allocate to a given activity. Role-based allocation assigns a resource to an activity based on their position within the organization and their relationship with other resources. In this article we consider the first two and assume role-based allocation can be used a priori to filter the potential resources. In capability-based allocation, usually a profile is defined for specifying resources capabilities and activities requirements (cf. Table 1). Organizational models [13,16], and resource meta-models [6,10,18], have also been used to represent resources capabilities. Among the resource allocation algorithms, we can highlight: data mining techniques and machine learning algorithms to derive allocation rules based on log events [9,11,12,16], and dynamic context-based resource allocation based on Markov decision process [8] or resource allocation based on hidden Markov models [10]. A more recent approach [6] allows specifying preferences for different resources using expressions based on a Resource Assignment Language (RAL), and generating a resources ranking considering a meta-model.

**Table 1.** Comparison with the related work.

| | [18] | [16] | [12] | [8] | [10] | [9] | [11] | [6] | Proposed |
|---|---|---|---|---|---|---|---|---|---|
| Activity profile | | ✓ | | | | | | | ✓ |
| Resource profile | ✓ | ✓ | | | ✓ | | | ✓ | ✓ |
| Performance & quality | | | | | | | | | ✓ |
| Resource meta-model | ✓ | | | | ✓ | | | ✓ | ✓ |
| History | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| Process mining tool | | ✓ | | | ✓ | | | | ✓ |
| Allocation at sub-process level | | | | | | | | | ✓ |

In this article, we propose a framework for recommending resource allocation based on process mining. We introduce a resource process cube that provides a flexible, extensible and fine-grained mechanism to abstract historical information about past process executions, extracted from process event logs. One difference between our approach and the approaches proposed in the literature is that we consider sub-processes as the target allocation unit; however, it can also be used to allocate resources at the activity level or at the process level, as a whole (see Sect. 2). Also, several metrics are computed over the cube, considering different criteria: fitting between resources capabilities and the expertise required to perform an activity, and past performance (frequency, duration, quality and cost). These metrics are combined to obtain a final recommendation ranking based on the Best Position Algorithm (BPA). The request to recommend the allocation of a resource is described as follows:

**Definition 1 (Recommended Resource Allocation Request).** *A* recommended resource allocation request *function is a function* $req(c, i, w) = rank$*, that given a process characterization* $c$*, a resource allocation information (historical and contextual) of the process execution* $i$*, and the weights describing the importance of each criterion* $w$*, returns a ranking of the most suitable resources to be assigned.*

The remainder of the article explains the different elements of this request, and it is structured as follows: in Sect. 2 the characterization of a resource allocation request is presented. Section 3 proposes the use of historical and contextual information to measure six different process criteria for an accurate resource allocation request. Section 4 presents the weighting of the different criteria and the recommendation algorithm. The implementation of the approach and its experimental evaluation is discussed in Sect. 5. Finally, the paper is concluded and future work is discussed in Sect. 6.

## 2 Resource Allocation Characterization

The first necessary step for a proper resource allocation is to characterize the request, i.e., what part of the process is the resource request for, and how similar is this request to others in the past. Most of the approaches in the literature limit that characterization to a simple *activity* level [9,10,12,16], i.e., a resource is always assigned to a single activity of the process, and only historical information of the execution of that activity is considered for future allocations.

In this article we propose a more flexible resource allocation characterization, where the allocations are not done at an activity-level or a process-level, but at a *sub-process-level*, i.e., the overall process is decomposed into sub-processes, and a resource is allocated for the execution of each sub-process. The decomposition of the process may be done manually using the own semantics of the process. For instance, let us consider a help-desk process (`HelpDesk`), for a company that provides support for both printers and servers. The process is decomposed by the two levels of customer interaction: first-contact level 1 and expert level 2. Each one of the two levels may correspond with a different sub-process. The decomposition may also involve an automatic process decomposition [1], using some of the decomposition approaches proposed in the literature, like Passages [3] or Single-Entry Single-Exit (SESE) [14]. Notice that, by definition, an activity or the overall process are also sub-processes. Therefore, the flexible resource allocation characterization proposed allows also the classical allocations at the activity or process levels. The usage of context dependent activity ordering is seen as a current challenge within intention-centric business process domain [5]. Considering the execution context can be useful for selecting the appropriate sub-process or select the required tasks for each process instance, allowing the optimization of resource allocation. Additionally, the sub-process characterization is combined with a typology characterization, i.e., the historical information is classified and used depending on the typology of the request. For instance, different typologies of processes may distinguish between normal/VIP clients, English/Spanish/German languages, or Internet/call-center interactions. In the `HelpDesk` example we consider two types of requests: printer-related and server-related problems. Note that, increasing the number of typologies may narrow the focus, but it may cause also a scarcity problem, i.e., not having enough information of each typology for a proper recommendation.

**Definition 2 (Characterization of a Resource Allocation Request).** *A resource allocation request characterization* $c = (f_1, \dots, f_n)$ *is a multi-factor representation of the request properties. The two-factor characterization proposed is a tuple* $c = (SP, T)$*, where* $SP$ *defines the sub-process where the resource is being requested, and* $T$ *is the typology of the process execution that request the resource.*

In `HelpDesk`, $c_1 = (level1, printer)$ and $c_2 = (level2, server)$ represent two different request characterizations for the same help-desk process.

## 3 Resource Allocation Criteria

The simplest resource allocations rely on pure random assignments between resources and requests. As it is shown in Table 1, more advanced systems base their decisions on specific criteria, e.g., the resource that is estimated to spend less time, or the one with more experience performing a task. In this article we propose a six-dimension recommended allocation that uses both historical and contextual information. The proposed dimensions are:

– Frequency Dimension: measures the rate of occurrence that a resource has completed the requested characterization.
– Performance Dimension: measures the execution time that a resource has achieved performing the requested characterization.
– Quality Dimension: measures the customer evaluation of the execution of the requested characterization performed by a resource.
– Cost Dimension: measures the execution cost of the requested characterization performed by a resource.
– Expertise Dimension: measures the ability level at which a resource is able to execute a characterization.
– Workload Dimension: measures the actual idle level of a resource considering the characterizations executed at the time.

Notice that the flexible nature of the proposed framework allows the inclusion of new dimensions, and the extension with other metrics proposed in the literature. In the remainder of this section we formalize the historical and contextual information used on the resource allocation request, in terms of resource process cubes and expertise matrices, respectively (Sect. 3.1), and we propose metrics to assess each one of the dimensions (Sect. 3.2).

## 3.1 Resource Process Cube and Expertise Matrices

We define the *resource process cube* $\mathbb{Q}$ as the semantic abstracting all the historical execution information of the process to be analyzed. The resource process cube is inspired by the process cubes presented in [2], and its definition is closer to the well-known OLAP cubes [7], providing slice and dice operations for the analysis of each specific characterization and resource.

**Definition 3 (Resource Process Cube).** *Let $r$, $c$, and $d$, be a resource, resource allocation request characterization, and dimension, respectively. A resource process cube $\mathbb{Q}[r][c][d]$ abstracts all the historical information about the resource $r$ and the characterization $c$ necessary to analyze the dimension $d$. Similarly, $\mathbb{Q}[\ ][c][d]$ abstracts the historical information about all resources for the execution of the characterization $c$, and $\mathbb{Q}[r][\ ][d]$ abstracts the information for all the characterizations performed by $r$.*

For example, in `HelpDesk`, given a characterization $c_1 = (level1, printer)$ and a resource $r_1 = mike$, $\mathbb{Q}[r_1][c_1][p]$ provides all the historical information related about the performance, such as, what is the maximal and minimal time *mike* needed to perform $c_1$ (denoted as $\mathbb{Q}[r_1][c_1][p].max$ and $\mathbb{Q}[r_1][c_1][p].min$, respectively), or the average time required by *mike* to perform $c_1$ (denoted as $\mathbb{Q}[r_1][c_1][p].avg$). Similarly, $\mathbb{Q}[\ ][c_1][p].max$ represents the maximal time required considering all the resources.

Note that, the resource process cube is a high-level semantic abstraction of the historical information, rather than an implementational definition. Therefore, the cube can be implemented using any database (relational or non-relational) or OLAP technology, and including, for example, pre-calculated values, or shared values among cells.

Besides historical information, the expertise dimension requires contextual information, i.e., it compares the current level of expertise of each resource with the desired level of expertise for the characterization to be performed. In [15] the authors propose a Human Resource Meta-Model (HRMM) where the expertise of the resources is classified by competencies, skills, and knowledge. Based on that model, we represent the expertise of a resource $r$ as an array of naturals $\mathbb{E}_r[1:n]$, where each position represents a specific competence, skill or knowledge, and the value of $\mathbb{E}_r[i]$ range from $\perp_i$ (usually 0 indicating the lack of competence/skill/knowledge $i$) to $\top_i$ (complete expertise on the competence/skill/knowledge $i$). The set of arrays for all the resources is known as the expertise resource matrix. Similarly, we represent the desired level of expertise required for performing a characterization $c$ as the array $\mathbb{E}_c[1:n]$. For instance, given the characterization $c_1 = (level1, printer)$ and a resource $r_1 = mike$, $\mathbb{E}_{c_1} = [2,2]$ denotes a mid-high required level (assuming $\top = 3$ and $\bot = 0$ for both positions) on printer hardware (position 1) and printer software (position 2), while *mike* has a low or non existent knowledge on printers denoted as $\mathbb{E}_{r_1} = [0,1]$.

## 3.2 Resource Allocation Metrics

In this subsection we present metrics for each one of the six before mentioned dimensions. All the metrics proposed are normalized between 0 and 1, and they satisfy the set of properties proposed in [17]: *validity* (i.e., metric and property must be sufficiently correlated), *stability* (i.e., stable against manipulations of minor significance), *analyzability* (i.e., measured values should be distributed between 0 and 1 with 1 being the best and 0 being the worst), and *reproducibility* (i.e., the measure should be independent of subjective influence).

In the remainder of the section we consider a resource process cube $\mathbb{Q}$ representing the historical information of the process, and expertise matrices $\mathbb{E}_r$ and $\mathbb{E}_c$ representing the expertise information.

**Frequency Dimension:** Let $\mathbb{Q}[r][c][f].total$ be the number of times a resource $r$ has performed the characterization $c$. Let $\mathbb{Q}[\,][c][f].total$ be the number of cases of characterization $c$. We define the metric as (1):

$$Frequency\_Metric(r,c) = \frac{logarithm(\mathbb{Q}[r][c][f].total) + 1}{logarithm(\mathbb{Q}[\,][c][f].total) + 1} \tag{1}$$

We use a logarithmic scale since we are mainly interested in measuring different magnitude orders between potential resources.

**Performance Dimension:** Let $\mathbb{Q}[r][c][p].avg$ be an operation that returns the average duration, considering only cases in which the resource $r$ has taken part in executing the characterization $c$. Let $\mathbb{Q}$[ ][c][p].min and $\mathbb{Q}$[ ][c][p].max be the minimum and maximum duration for executing the characterization $c$. We define the metric as (2):

$$Performance\_Metric(r,c) = \frac{\mathbb{Q}[\,][c][p].max - \mathbb{Q}[r][c][p].avg}{\mathbb{Q}[\,][c][p].max - \mathbb{Q}[\,][c][p].min} \tag{2}$$

**Quality Dimension:** Let $\mathbb{Q}$[r][c][q].avg be an operation that returns the average quality, considering only cases in which the resource $r$ has taken part in executing the characterization $c$. Let $\mathbb{Q}$[ ][c][q].min and $\mathbb{Q}$[ ][c][q].max be the minimum and maximum quality evaluation for the executed characterization $c$. We define the metric as (3):

$$Quality\_Metric(r,c) = \frac{\mathbb{Q}[r][c][q].avg - \mathbb{Q}[\,][c][q].min}{\mathbb{Q}[\,][c][q].max - \mathbb{Q}[\,][c][q].min} \tag{3}$$

**Cost Dimension:** Let $\mathbb{Q}$[r][c][co].avg be an operation that returns the average cost, considering only cases in which the resource $r$ has taken part in executing the characterization $c$. Let $\mathbb{Q}$[ ][c][co].min and $\mathbb{Q}$[ ][c][co].max be the minimum and maximum cost for the executed characterization $c$. We define the metric as (4):

$$Cost\_Metric(r,c) = \frac{\mathbb{Q}[\ ][c][co].max - \mathbb{Q}[r][c][co].avg}{\mathbb{Q}[\ ][c][co].max - \mathbb{Q}[\ ][c][co].min} \tag{4}$$

**Expertise Dimension:** To determine if a resource $r$ is qualified to execute a characterization $c$, we present two metrics that uses the expertise matrices explained in Sect. 3.1. To evaluate this dimension, we compare the value of each level of expertise $\mathbb{E}_r$ with the corresponding value in $\mathbb{E}_c$, in order to measure the under-qualification or the over-qualification level of a resource. To define the under-qualification metric, we first calculate an under-qualification degree comparing each value as follows in (5):

$$under(i) = \begin{cases} \frac{\mathbb{E}_c[i]-\mathbb{E}_r[i]}{\mathbb{E}_c[i]-\perp_i} & \text{if } \mathbb{E}_c[i] \geq \mathbb{E}_r[i] \\ 0 & \text{otherwise} \end{cases} \tag{5}$$

Then the metric to measure the under-qualification is defined as (6):

$$UnderQualification\_Metric = 1 - \frac{1}{n}\sqrt{\sum_{i=1}^{n}\big(under(i)\big)^2} \tag{6}$$

Symmetrically, to determine the over-qualification metric, we define (7):

$$over(i) = \begin{cases} \frac{\mathbb{E}_r[i]-\mathbb{E}_c[i]}{\top_i-\mathbb{E}_c[i]} & \text{if } \mathbb{E}_r[i] \geq \mathbb{E}_c[i] \\ 0 & \text{otherwise} \end{cases} \tag{7}$$

The metric to measure the over-qualification of a resource is then defined as (8):

$$OverQualification\_Metric = 1 - \frac{1}{n}\sqrt{\sum_{i=1}^{n}\big(over(i)\big)^2} \tag{8}$$

In both qualification metrics, $n$ represents the number of expertise elements in the matrix. We use the Euclidean distance because all expertise features are equally relevant, are defined in the same scale, and to favor smaller differences in all features at the same time. Notice that if the expertise of a resource $r$ perfectly match with the expertise required for a characterization $c$, the value for both metrics will be 1.

**Workload Dimension:** Let $\mathbb{Q}$[r][ ][w].total be a function that returns the number of cases in which a resource $r$ is working at the moment when a new resource allocation request is required. Let $\mathbb{Q}$[r][ ][w].top and $\mathbb{Q}$[r][ ][w].bottom be the maximum and minimum number of cases that a resource can attend simultaneously. We define the metric as (9):

$$Workload\_Metric(r, c) = \frac{\mathbb{Q}[r][\ ][w].top - \mathbb{Q}[r][\ ][w].total}{\mathbb{Q}[r][\ ][w].top - \mathbb{Q}[r][\ ][w].bottom} \tag{9}$$

## 4 Recommended Resource Allocation

We face the challenge of allocating appropriate resources to execute characterizations dynamically. We propose a recommendation system to create a final resource ranking, considering the six dimensions presented in Sect. 3. The recommendation system is inspired on the portfolio-based algorithm selection [19]. To accomplish this goal, we consider the top-k queries, a technique that allows to obtain the k most relevant items in a dataset. According with [4], to give an answer to top-k queries we use $m$ lists of $n$ data items, so that each data item has a local score in each list, and the lists are ordered accordingly to the local score of its data items. With those lists, the BPA algorithm [4] can be used to get the top-k results.

In order to obtain the most appropriate resource, we need to generate an ordered list of resources according to their metric scores in every dimension. Before applying the algorithm, we need to combine all ordered metric score lists considering the weights specified for each dimension, e.g., we could give more importance to the cost and frequency dimensions, rather than quality or expertise. If the user does not want to incorporate weights for the recommendation, each $m$ list is not modified; otherwise, each local score is multiplied by the respective weight, generating updated $m$ lists.

Giving the $m$ lists, the final ranking can be calculated by applying the BPA algorithm, which is used to find the k-data items that have the highest overall score. BPA calculates the overall score for each data item, registering the best seen positions, and maintain in a set $Y$ the k-data items with the highest overall score. The algorithm allows an iterative approach to access and evaluate the resources based on their local score and the position in each list. If at same point the set $Y$ contains k-data items whose overall scores are higher than or equal to a generated *threshold*, then there is no need to continue scanning the rest of the lists. The output of the algorithm is an ordered list, where the final score for each resource is stored. The first value represents the resource with the highest overall score and therefore the best recommendation. For details on the algorithm we refer the reader to [4].

## 5 Implementation and Experimental Evaluation

A real-life help desk process (`HelpDesk`) was selected to evaluate our approach. We focused on two typologies: printers and servers. The `HelpDesk` process includes two attention levels and their corresponding activities (sub-process 1 and sub-process 2). For the executed experiments, event logs with different amount of cases were simulated. The attributes for each case include Case ID, Subprocess group, Process Typology, Resource, Cost, Customer Satisfaction (Quality), Creation date, Closing date and Priority. Three experiments were performed:

– **Experiment 1:** Calculate the top 3-queries processing over the sorted lists, considering each single metric by itself.
– **Experiment 2:** Reproduce an scenario for 3 types of companies: a large size called General Consulting, a small size called Service Guide, and a mid size company named DeskCo. Specific weight values were defined for each scenario.
– **Experiment 3:** A similar scenario as the described in experiment 2, but both the event log size and the amount of resources were increased. Due to the variety in the resource quantity, we calculated the top-2 queries with 28 resources, preserve the top-3 with 20, and use top-5 queries with 70 resources.

**Discussion:** Table 2 specifies the parameters used in the different experiments and their results. For experiments 1 and 2 (with 20 resources), an event log was simulated, which includes 3 resources whose frequency of participation in

**Table 2.** Resource recommendations for the 3 experiments

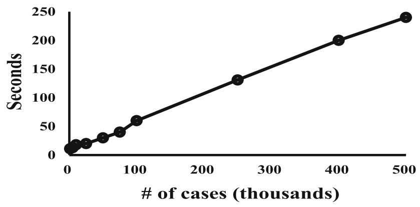
| Exp. | Weights (%) | # Cases | #R SP1 | #R SP2 | Ranking | Time (sec) |
|---|---|---|---|---|---|---|
| 1.1 | F:100 - others:0 | 1200 | 20 | 20 | R06: 0.601 - R04: 0.593 - R05: 0.554 | 0.954 |
| 1.2 | P:100 - others:0 | 1200 | 20 | 20 | R03: 0.851 - R02: 0.833 - R01: 0.832 | 0.954 |
| 1.3 | Q:100 - others:0 | 1200 | 20 | 20 | R09: 0.913 - R07: 0.864 - R08: 0.808 | 0.954 |
| 1.4 | C:100 - others:0 | 1200 | 20 | 20 | R18: 0.962 - R20: 0.962 - R19: 0.959 | 0.954 |
| 1.5 | U:100 - O:100 - others:0 | 1200 | 20 | 20 | R12: 1.000 - R13: 1.000 - R14: 1.000 | 0.954 |
| 2.1 | F:010 - P:050 - Q:010 C:100 - U:015 - O:000 | 1200 | 20 | 20 | R20: 0.647 - R03: 0.635 - R18: 0.632 | 11.122 |
| 2.2 | F:025 - P:015 - Q:100 C:030 - U:075 - O:065 | 1200 | 20 | 20 | R19: 0.802 - R14: 0.758 - R13: 0.754 | 11.565 |
| 2.3 | F:050 - P:050 - Q:050 C:050 - U:050 - O:050 | 1200 | 20 | 20 | R19: 0.725 - R03: 0.712 - R02: 0.675 | 10.897 |
| 3.1.1 | F:010 - P:050 - Q:010 C:100 - U:015 - O:000 | 1200 | 14 | 14 | R01: 0.795 - R02: 0.788 - R14: 0.784 | 10.942 |
| 3.1.2 | F:010 - P:050 - Q:010 C:100 - U:015 - O:000 | 10000 | 14 | 14 | R13: 0.769 - R02: 0.567 - R14: 0.758 | 17.160 |
| 3.1.3 | F:010 - P:050 - Q:010 C:100 - U:015 - O:000 | 100000 | 14 | 14 | R13: 0.767 - R14: 0.765 - R02: 0.764 | 59.063 |
| 3.2.1 | F:010 - P:050 - Q:010 C:100 - U:015 - O:000 | 1200 | 20 | 20 | R19: 0.649 - R20: 0.647 - R03: 0.635 | 11.122 |
| 3.2.2 | F:010 - P:050 - Q:010 C:100 - U:015 - O:000 | 10000 | 20 | 20 | R01: 0.586 - R03: 0.582 - R02: 0.573 | 17.642 |
| 3.2.3 | F:010 - P:050 - Q:010 C:100 - U:015 - O:000 | 100000 | 20 | 20 | R01: 0.834 - R20: 0.784 - R18: 0.783 | 58.913 |
| 3.3.1 | F:010 - P:050 - Q:010 C:100 - U:015 - O:000 | 1200 | 35 | 35 | R03: 0.626 - R05: 0.618 - R04: 0.572 | 11.014 |
| 3.3.2 | F:010 - P:050 - Q:010 C:100 - U:015 - O:000 | 10000 | 35 | 35 | R04: 0.608 - R05: 0.603 - R01: 0.599 | 17.739 |
| 3.3.3 | F:010 - P:050 - Q:010 C:100 - U:015 - O:000 | 100000 | 35 | 35 | R04: 0.593 - R02: 0.580 - R11: 0.428 | 58.637 |

F= Frequency, P= Performance, Q= Quality, C= Cost, U= Underqualified, O= Overqualified, R= Resource and others= Other dimensions the case resolution in `HelpDesk` is higher compared to the other resources of the same level. Equally, it was simulated the existence of 3 resources that have better resolution time resolving cases, 3 resources that perform better in quality, 3 resources that present the lowest costs, and 3 resources that fit the expertise level required.

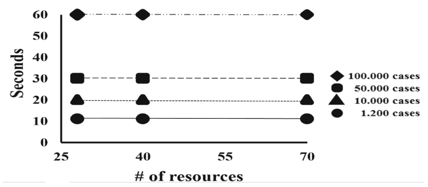
In **experiment 1**, it is possible to observe that the best specified resources for each dimension are the expected, existing a clear correlation between the proposed metric for each dimension.

In **experiment 2**, for each company different weights are specified, according to the priorities for each one (e.g., General Consulting (exp. 2.1) has an interest for cheaper and faster solutions; Service guide (exp. 2.2) gives more importance to quality services and DeskCo (exp. 2.3) prefers giving a medium value to all dimensions). Considering the criteria established in the Resource Allocation Request function, complex and high calculation results are obtained faster and simpler. If top-3 queries are applied, the results for each company establish a ranking with the recommended resources, different from experiment 1. The recommended resources are different for each company, proving that our approach produces resource recommendations based on the given requests. For example, for the General Consulting Company, R20, R03 and R18 are recommended under the criterion of low cost but balanced with the resolution mean time. For DeskCo, R19, R03 and R02, are the suitable ones to accomplish the activities based on the criteria established in the request.

In **experiment 3** (exp. 3.1.1 to exp. 3.3.3) a similar scenario to the experiment 2 was executed, but with changes on the amount of cases and resources per attention level. Logs with 1.200, 10.000, 100.000 and 500.000 cases were considered; and the amount of resources are 28, 40 and 70. Figure 1a displays the behavior of the processing time according to the amount of cases. As it can be seen, when more historical data is used to make the recommendation, a linear relation appears between the time and the log size. If the log size is larger, the information to be processed by the cube is larger and higher is the time to generate the ranking, but this dependency is linear. This proves that it is possible to process large amounts of information through the technique and get quick response times, and if its required, it is possible to include additional dimensions to get a better recommendation. Figure 1b displays that the result of using the BPA algorithm is not dependent of the number of resources used at the allocation request moment. It was proven, in an experimental way, it is possible to obtain the recommended resources ranking without having to visit all positions in the ordered list, thanks to the threshold management and the early stop condition of the BPA algorithm. This confirms that the results of the BPA algorithm are constant regarding the amount of resources given to resolve the top-k queries problem, which could be very useful for companies with high quantity of resources.



(a) Number of cases



(b) Number of resources

**Fig. 1.** Performance analysis

## 6 Conclusions and Future Work

We proposed a flexible framework for dynamically allocating the most appropriate resources to execute a sub-process. Our contributions are fourfold. First, while other approaches focus only on a single process perspective, the proposed framework considers the organizational, time and case perspectives. We define specific dimensions to assess different resource features: frequency, performance, quality, cost, expertise, and workload. Secondly, unlike others approaches in the literature that consider resource allocation only at an activity level, the proposed framework considers it at a generic sub-process level (an activity can be seen as a specific-case). Third, the resource allocation request, together with a precise characterization of both resources and activities, provide a fine-grained degree of customization. Finally, the conceptual framework is designed to be generic and extensible, being able to adapt to any company-specific scenario.

Our work has been implemented and tested in a `HelpDesk` scenario, and the experimental results show that given a specific characterization it is possible to obtain a final ranking of recommended resources based on multi-factor criteria. We tested the BPA algorithm with different event log sizes and resource amounts. We observed a linear relation between the algorithm performance and the log size. Moreover, the BPA algorithm confirms its efficiency to compute top-k results independent of the amount of resources.

As future work, we plan to extend the comparison with existing works in order to generate a comprehensive theoretical analysis and enhance the experimental evaluation. We aim to evaluate the effectiveness and the efficiency of alternative approaches and compare them with our framework in a partial or complete way. We plan to use artificial scenarios and case studies with real data to validate our recommendation technique. This could be useful to compare the results obtained by other approaches and real resource allocations, with the ones proposed by our framework. Incorporate new dimensions to the resource process cube for improving the analysis may also be considered. This work attempts to encourage organizations to use real performance time, quality and cost data, to generate better resource allocations.

**Acknowledgments.** This work is partially supported by Comisión Nacional de Investigación Científica – CONICYT – Ministry of Education, Chile, Ph.D. Student Fellowships, and by University of Costa Rica Professor Fellowships.

## References

1. van der Aalst, W.M.P.: Decomposing petri nets for process mining: a generic approach. Distrib. Parallel Databases **31**(4), 471–507 (2013)
2. van der Aalst, W.M.P.: Process cubes: slicing, dicing, rolling up and drilling down event data for process mining. In: Song, M., Wynn, M.T., Liu, J. (eds.) AP-BPM 2013. LNBIP, vol. 159, pp. 1–22. Springer, Heidelberg (2013)
3. van der Aalst, W.M.P., Verbeek, H.M.W.: Process discovery and conformance checking using passages. Fundam. Inform. **131**(1), 103–138 (2014)
4. Akbarinia, R., Pacitti, E., Valduriez, P.: Best position algorithms for efficient top-k query processing. Inf. Syst. **36**(6), 973–989 (2011)
5. van Beest, N., Russell, N., ter Hofstede, A.H.M., Lazovik, A.: Achieving intention-centric BPM through automated planning. In: 7th IEEE International Conference on Service-Oriented Computing and Applications (SOCA 2014), Matsue, Japan, November 17–19, 2014, pp. 191–198 (2014)
6. Cabanillas, C., García, J.M., Resinas, M., Ruiz, D., Mendling, J., Ruiz-Cortés, A.: Priority-based human resource allocation in business processes. In: Basu, S., Pautasso, C., Zhang, L., Fu, X. (eds.) ICSOC 2013. LNCS, vol. 8274, pp. 374–388. Springer, Heidelberg (2013)
7. Chaudhuri, S., Dayal, U.: An overview of data warehousing and OLAP technology. ACM Sigmod Rec. **26**(1), 65–74 (1997)
8. Huang, Z., van der Aalst, W.M.P., Lu, X., Duan, H.: Reinforcement learning based resource allocation in business process management. Data Knowl. Eng. **70**(1), 127–145 (2011)
9. Huang, Z., Lu, X., Duan, H.: Mining association rules to support resource allocation in business process management. Expert Syst. Appl. **38**(8), 9483–9490 (2011)
10. Koschmider, A., Yingbo, L., Schuster, T.: Role assignment in business process models. In: Daniel, F., Barkaoui, K., Dustdar, S. (eds.) Business Process Management Workshops. Lecture Notes in Business Information Processing, vol. 99, pp. 37–49. Springer, Heidelberg (2012)
11. Liu, T., Cheng, Y., Ni, Z.: Mining event logs to support workflow resource allocation. Knowl. Based Syst. **35**, 320–331 (2012)
12. Liu, Y., Wang, J., Yang, Y., Sun, J.: A semi-automatic approach for workflow staff assignment. Comput. Ind. **59**(5), 463–476 (2008)
13. Ly, L.T., Rinderle, S., Dadam, P., Reichert, M.: Mining staff assignment rules from event-based data. In: Bussler, C.J., Haller, A. (eds.) BPM 2005. LNCS, vol. 3812, pp. 177–190. Springer, Heidelberg (2006)
14. Munoz-Gama, J., Carmona, J., van der Aalst, W.M.P.: Single-entry single-exit decomposed conformance checking. Inf. Syst. **46**, 102–122 (2014)
15. Oberweis, A., Schuster, T.: A meta-model based approach to the description of resources and skills. In: AMCIS, p. 383 (2010)
16. Rinderle-Ma, S., van der Aalst, W.M.P.: Life-cycle support for staff assignment rules in process-aware information systems (2007)
17. Rozinat, A., van der Aalst, W.M.P.: Conformance testing: measuring the alignment between event logs and process models. BETA Research School for Operations Management and Logistics (2005)

18. Russell, N., van der Aalst, W.M.P., ter Hofstede, A.H.M., Edmond, D.: Workflow resource patterns: identification, representation and tool support. In: Pastor, Ó., Falcão e Cunha, J. (eds.) CAiSE 2005. LNCS, vol. 3520, pp. 216–232. Springer, Heidelberg (2005)
19. Xu, L., Hutter, F., Hoos, H.H., Leyton-Brown, K.: Satzilla: portfolio-based algorithm selection for SAT. CoRR abs/1111.2249 (2011)
20. Zhao, W., Zhao, X.: Process mining from the organizational perspective. In: Wen, Z., Li, T. (eds.) Foundations of Intelligent Systems. Advances in Intelligent Systems and Computing, vol. 277, pp. 701–708. Springer, Heidelberg (2014)