Human Resource Allocation or Recommendation Based on Multi-factor Criteria in On-demand and Batch Scenarios

Michael Arias*

Computer Science Department, School of Engineering,
Pontificia Universidad Católica de Chile,
Vicuña Mackenna 4860, 7820436 Macul, Chile
E-mail: m.arias@uc.cl
*Corresponding author

Jorge Munoz-Gama

Computer Science Department, School of Engineering,
Pontificia Universidad Católica de Chile,
Vicuña Mackenna 4860, 7820436 Macul, Chile
E-mail: jmun@ing.puc.cl

Marcos Sepúlveda

Computer Science Department, School of Engineering,
Pontificia Universidad Católica de Chile,
Vicuña Mackenna 4860, 7820436 Macul, Chile
E-mail: marcos@ing.puc.cl

Juan Carlos Miranda

Division Manager,
Software & Consulting Group,
36th Street, San José, Costa Rica
E-mail: jmiranda@scgint.com

Abstract: Dynamic resource allocation is considered a major challenge in the context of business process management. At the operational level, flexible methods that support resource allocation and which consider different criteria at run-time are required. It is also important that these methods are able to support multiple allocations in a simultaneous manner. In this paper, we present a framework based on multi-factor criteria that proposes a recommender system which is capable of recommending the most suitable resources for executing a range of different activities, while also considering individual requests or requests made in blocks. To evaluate the proposed framework, a number of experiments were conducted using different test scenarios. These scenarios provide evidence that our approach based on multi-factor criteria successfully allocates the most suitable resources for executing a process in real and flexible environments. In order to demonstrate this assertion, we use a help-desk process as a real case study.

Keywords: Human Resource Allocation; Human Resource Recommendation; Multi-factor Criteria; On-demand; Batch; Dynamic Resource Allocation; Recommender System, Business Process Management

Biographical notes: Michael Arias is Associate professor at the Universidad de Costa Rica and researcher at the Universidad de Costa Rica and Pontificia Universidad Católica de Chile (PUC). At present, professor Arias is currently enrolled in the Process Mining UC research group at PUC. Research fields of interest include process mining, business process management, human resource management, process oriented data science and project management.

Jorge Munoz-Gama is Assistant professor and co-director of the Process Mining UC research group at Pontificia Universidad Católica de Chile. He is member of the IEEE Task Force on Process Mining, and his research interests include process mining, process oriented data science, and their application for healthcare, education and business processes.

Marcos Sepúlveda is Associate professor, co-director of the Process Mining UC research group at Pontificia Universidad Católica de Chile, and director of the Centro de Estudios de Tecnologías de Información de la Pontificia Universidad Católica de Chile (CETIUC). He is member of the IEEE Task Force on Process Mining, and his research interests include business process management and modelling, process mining and business intelligence.

Juan Carlos Miranda Master's Degree on Computer Science, specialisation on Project Management. More than 10 years' experience developing software, consulting and project management. Certified consultant and developer for SAP® Business One.

1 Introduction

It is important that resource management undergoes further maturity within the discipline of business process management (BPM), despite ongoing developments in the field (Cabanillas et al. 2015). The organizational perspective (van der Aalst 2016), also known as the resource perspective (Dumas et al. 2013), provides an in-depth focus on the behaviour of resources and how they interact in the execution of a process so that improvements can be proposed to optimize their use within the process lifecycle, and also support changing challenges in today's organisations (Braganza et al. 2017). One of the main challenges from the organizational perspective is dynamic human resource allocation to activities during the execution of a business process. This is significant because an inappropriate allocation may impact on process performance (Liu et al. 2014, Zhao & Zhao 2014), affect the efficient use of the resources (Kumar et al. 2002), or lead to an increase in execution costs (Huang, van der Aalst, Lu & Duan 2011). It should be noted that human and non-human resources can be involved in the execution of processes (Russell et al. 2005). Nevertheless, in this paper we focus on human resources due to their importance in the execution and management of business processes. (Havur et al. 2015, Senderovich et al. 2014). The literature (see Section 2) outlines distinct techniques and methods that integrate different types of information regarding resource allocation. For example, certain methods include information about the profile required for the activity to be undertaken, in relation to resource capabilities (Russell et al. 2005, Rinderle-Ma & van der Aalst 2007, Koschmider et al. 2011, Cabanillas et al. 2013). However, we identify five limitations among the approaches currently available, as follows: 1) resource allocation is only executed at the activity level, even though undertaking the allocation in other allocation levels (e.g., the process or sub-process level) would also be desirable; 2) typically, requests are undertaken for the allocation of a single resource to a single activity, i.e., it is not possible to allocate resources to multiple activities simultaneously; 3) methods capable of allocating resources dynamically are required, including the consideration of different criteria to evaluate how good a resource might be in order to perform an activity; 4) there is a lack of mechanisms that complement the allocation of individual resources with other, more flexible, mechanisms, e.g., ones that provide a ranking of possible resources for allocation; 5) greater focus is required on the use of historical information stored in event logs for evaluating the suitability of a resource to perform a given activity based on his/her performance in the past.

Dynamic resource allocation based on process mining seeks to improve allocation efficiency and may help to reduce the cost and execution time of business processes (Zhao & Zhao 2014). By means of process mining (van der Aalst 2016), it is possible to extract knowledge from the historical information stored in event logs. This knowledge can subsequently be used to understand how the resources are involved with the process and how they are interrelated to one another. In turn, this enables new methods to be proposed to help undertake more appropriate allocations. To address this challenge, we propose a dynamic and flexible framework that can be utilised to return the most suitable resource for allocation, or to return a ranking of the most suitable resources recommended for allocation. Specifically, this approach uses a number of different criteria, including: historical information relating to process execution, including frequency, performance, quality, and cost; the fit between the competencies of the resources and the competencies required to undertake an activity, sub-process or process; and considering the workload of the resources when carrying out the allocation or recommendation. The framework considers four use cases: *On-demand Resource Allocation*, *Batch Resource Allocation*, *On-demand Resource Recommendation*, and *Batch Resource Recommendation*. It should be noted that, in fact, the On-demand cases are particular cases of the Batch use cases. However, their separation is worthwhile because the solution methods for the On-demand cases are simpler. These different use cases are handled using a recommender system based on two methods: 1) resource allocation based on Integer Linear Programming (ILP); and 2) resource recommendation based on the Best Position Algorithm (BPA). Previously, Arias et al. (2015) presented a framework to recommend a prioritised list of the most suitable resources to be allocated to a single request (only the *On-demand Resource Recommendation* use case). In this paper, we extend and enhance the aforementioned framework in several aspects (see details in Section 2). We implement the framework extension by means of two plug-ins developed using the tool ProM (Verbeek et al. 2010), which is widely used throughout the process mining community. Furthermore, we define an extension to the Java OpenXES library to standardise and represent the historical and contextual information used to generate the required knowledge. To assess the proposed framework, we undertake an empirical analysis using a set of experiments to verify the performance of the proposed solution. Furthermore, we conduct a case study by applying the framework to a real-life help-desk process that responds to incidents arising in a consultancy firm specialising in business software solutions.

The remainder of the paper is structured as follows. Section 2 describes related work. Section

3 gives a general overview of the framework. Section 4 presents concepts and definitions used as part of the solution arising from our approach. In Section 5, we present distinct concepts, including: the characterization of a resource allocation/recommendation request; how to abstract the use of historical and contextual information to accurately evaluate each request; and the different metrics used. In Section 6, we propose the four use cases for resource allocation. The implementation and empirical analyses are included in Section 7. Section 8 outlines the results generated from analysing our approach by means of the aforementioned case study. The role of the defined weights is presented in Section 9. Finally, Section 10 highlights the main conclusions.

2 Literature review

Human resource allocation performs an important role in the context of BPM. Different approaches covered in the literature describe mechanisms that seek to boost the efficiency of resource allocation (Huang, van der Aalst, Lu & Duan 2011, Russell et al. 2005, Rinderle-Ma & van der Aalst 2007, Koschmider et al. 2011, Cabanillas et al. 2013, Liu et al. 2008, Huang, Lu & Duan 2011, Liu et al. 2012). For example, Russell et al. (2005) present a collection of resource patterns in the context of process-aware information systems (van der Aalst 2009), that have been evaluated via different commercial workflow systems. Three different types of resource patterns defined include: history-based allocation; capability-based allocation; and role-based allocation. History-based allocation undertakes the allocation of work based on the execution history of each resource. Capability-based allocation provides a mechanism for resource allocation by comparing the requirements for executing the activity, in conjunction with the capability profile of the potential resources that may execute that activity. In the literature revised, organizational models (Rinderle-Ma & van der Aalst 2007, Ly et al. 2005) and resource meta-models (Koschmider et al. 2011, Cabanillas et al. 2013, Liu et al. 2008) are used to represent resource capabilities. Role-based allocation undertakes resource allocation by considering the role the resources perform and the position they hold within the organization. We use the first two patterns in our framework, whereas we assume role-based allocation a priori as an alternative for filtering potential resources. Data mining techniques and machine learning algorithms have been proposed for resource allocation, based on the use of historical process information, in order to support the decision-making process (Rinderle-Ma & van der Aalst 2007, Liu et al. 2008, Huang, Lu & Duan 2011, Liu et al. 2012, Ly et al. 2005, Kim et al. 2013, Obregon et al. 2013, Liu et al. 2007). Furthermore, Markov models have been proposed to improve resource allocation. For example, in (Huang, van der Aalst, Lu & Duan 2011) a mechanism is proposed that is modeled as a Markov Decision Process, which applies reinforcement learning to analyse feedback in real time and undertake a dynamic resource allocation with the aim of minimising the overall cost of a given case and ensuring the best process execution performance. The hidden Markov model has been useful for recommending allocation based on roles for process activities (Koschmider et al. 2011), as well as in terms of forming part of a method of discovery of proposed processes to enhance resource allocation by means of a probabilistic approach (Carrera & Jung 2014). Additional efforts propose alternatives that explore the structural characteristics of business processes and the resources available for undertaking a resource allocation (Xu et al. 2008). Alternatively, they rely on Ant Colony Optimization-type algorithms to identify the optimum solution for allocating resources (Huang et al. 2012).

Cabanillas et al. (2013) present an approach that helps to specify preferences for different resources using expressions that are based on a Resource Assignment Language (RAL), which, in turn, generates a ranking of resources using a meta-model. Furthermore, in (Cabanillas et al. 2015) a series of design-time analysis operations is proposed, which help to identify how resources are involved with process activities and, using the RAL language, can determine the conditions for selecting resource candidates to execute activities. The desire to improve the efficiency of resource allocation has led to the incorporation of characteristics associated with resource behaviour. In our approach, we use expertise and availability as two dimensions within the framework. Nevertheless, other studies suggest measuring the compatibility between resources (Kumar et al. 2013); considering factors such as preferences and cooperation (Nakatumba & van der Aalst 2009); optimizing allocation by addressing the cognitive aspects of resources, including motivation, satisfaction and training (Vanderfeesten & Grefen 2015); or even implementing a Q-learning algorithm as part of a method to measure social relationships between two resources (Liu et al. 2014). Recently, Wibisono et al. (2015) introduced a dynamic allocation approach based on the Naïve Bayes model, in an attempt to improve the performance prediction of resources in terms of completion time. However, this approach does not incorporate other factors that help to optimize the proposed prediction (e.g., incorporating workflow resource patterns (Russell et al. 2005)), or which consider the transfer of resources between activities. Kumar and Wang (Kumar & Wang 2015) propose a resource-driven workflow framework that can be used to schedule the resources that are involved in a process, thereby avoiding the occurrence of allocation conflicts. This approach proposes the execution of a data dependency analysis in order to validate whether some hard or soft constraints between two tasks are satisfied. However, the proposed framework only assesses the resources in terms of the role and resource availability; it neither considers additional factors during the resource evaluation, nor the use of historical information regarding previous process executions. Moreover, this resource-driven approach does not support the scheduling of resources with relation to a Batch of requests that require solving in a simultaneously manner.

Resource allocation is an increasingly important topic in the context of BPM. Nevertheless, the level of complexity of these methods remains high in terms of interpretation and understanding by process owners. Having conducted a literature review, certain limitations were identified: methods should allow for the simultaneous allocation of resources; methods should be more user-oriented; methods should be flexible and extensible; methods should enable the combination of different criteria in order to evaluate the resources to be allocated; and methods should be able to undertake resource allocation at different levels of abstraction, even helping to adjust the identification of the most suitable allocation unit, whether this relates to activities, sub-processes or the entire process. Previously, Arias et al. (2015) presented a Framework for Recommending Resource Allocation based on Process Mining. This prior framework helped to define a Resource Allocation Request at execution time, and used a recommender system based on BPA (Akbarinia et al. 2011), to recommend a prioritised list of the most suitable resources to be allocated to requests made on an individual basis. A characterization based on multi-factor criteria was proposed and the resource recommendation could be performed at the sub-process level, rather than merely at the activity level. They introduced a resource process cube $\mathbb{Q}$ as a flexible and extensible mechanism for abstracting historical information relating to the process execution stored in event logs. Based on this information, distinct metrics were calculated for the different allocation criteria. Finally, by means of BPA, a ranking was compiled with the recommended resources to be allocated.

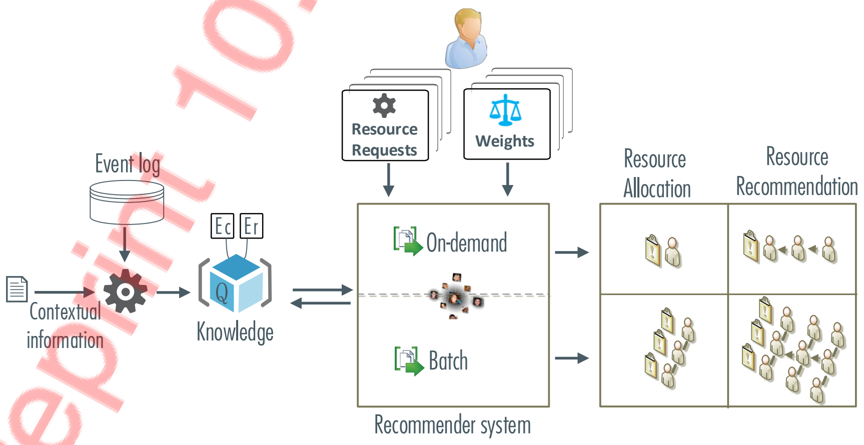
In the current paper, we extend and enhance the aforementioned framework in the following ways:

- We propose undertaking the resource allocation or recommendation by considering individual (On-demand) or block-based (Batch) requests.
- The previous paper proposed a single use case to provide a ranking of resources for each individual request (*On-demand Resource Recommendation*). In this paper, we add three more use cases (explained in Section 6): *On-demand Resource Allocation*, *Batch Resource Allocation*, and *Batch Resource Recommendation*.
- We propose the use of BPA2 as an alternative to BPA, which was utilised previously by the recommender system. The second version of BPA is more efficient in terms of execution time.
- In addition to the recommendation method based on BPA2, we add an allocation method based on ILP.

3 Framework overview

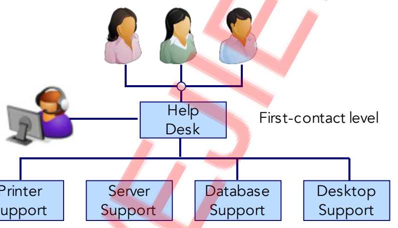
Figure 1 shows the proposed framework for allocating/recommending the most suitable resources for executing activities in a business process. This framework helps to specify one or various resource requests at run-time. The allocations consider contextual and historical information of the process execution, as well as weights that describe the level of importance of each individual criterion, in line with the priorities established by the respective user responsible for undertaking the allocation. Different metrics enable the resources to be evaluated in accordance with the specified criteria. By means of the resource process cube $\mathbb{Q}$ and the expertise matrices $\mathbb{E}_c$ and $\mathbb{E}_r$ (see Section 5), information can be abstracted and the required knowledge generated in order to help determine the suitability of the resources. The recommender system is thereafter able to generate the allocation or recommendation, according to the aforementioned settings.

Figure 1: General framework to allocate/recommend resources



The improvements implemented in our framework enable resource allocation or recommendation to be undertaken more efficiently and with greater flexibility. The results obtained demonstrate the usability of the framework, in which the generation of allocations/recommendations is enabled using the two proposed methods, while simultaneously considering different criteria. These methods can be applied to a wide variety of scenarios, for example, to resolve requests in a typical help-desk process (see Figure 2), in which the task of allocating resources is a common requirement. Frequently, a company with a help-desk service provides support in a range of distinct areas, including those relating to printers, servers, databases and desktops, among others. This support process may include different levels of customer interaction, for instance, first-contact level 1 and expert level 2. We use the help-desk scenario as a running example throughout the entire paper, including the case study (see Section 8). Furthermore, evaluation by means of a case study helps to provide evidence of the applicability of the framework in a real-life business environment.

Figure 2: Overview of the help-desk process example



4 Preliminaries

We address the problem of resource allocation in our framework by analysing two decision scenarios. The first scenario occurs when the framework must allocate a single resource to each request. The second scenario occurs when the person responsible wishes to generate a priority ranking of alternative resources that can be allocated to each request, so that he/she can ultimately decide whom to allocate. In this section, we present the ILP technique and the BPA, which we use to address the two aforementioned scenarios.

4.1 Integer Linear Programming problem definition

Regarding resource allocation, we use an ILP (Schrijver 1998) approach to solve the problem of identifying the optimal resource designated for allocation from a set of resources that are evaluated according to different metrics. The underlying ILP problem can be posed as follows (de Leoni & van der Aalst 2013):

Let $x_1, \ldots, x_n$ be a set of variables, each of which can be Boolean, continuous or integer. ILP problems can be formulated as follows:

A linear function $f(.)$ to be maximized/minimized.
max/min $f(x_1, \ldots, x_n)$

And a set of m linear constraints, with m finite.

$$a_{11}x_1 + \ldots, a_{1n}x_n \leqslant b_1$$
$$\vdots$$
$$a_{m1}x_1 + \ldots, a_{mn}x_n \leqslant b_m$$

The linear function, also called the *objective function*, is the equation that requires optimisation, considering a set of constraints and decision variables that need to be minimised or maximised using linear programming techniques. Besides the objective function, the linear constraint consisted of either an equality or an inequality associated with certain linear combinations of the decision variables (Vanderbei 2001). Other forms, such as problems with constraints of different forms, can always be rewritten into an equivalent problem in this form. For example, an equality constraint $a_{11}x_1$ +... + $a_{1n}x_n$ = b_1 can be replaced with two inequality constraints: $a_{11}x_1$ +... + $a_{1n}x_n \leqslant b_1$ and $-\ a_{11}x_1\ -$ ... $-\ a_{1n}x_n \leqslant -\ b_1$. A constraint $a_{11}x_1$ + ... + $a_{1n}x_n$ <b_1 can be replaced with $a_{11}x_1$ + ... + $a_{1n}x_n \leqslant b_1 - \epsilon$, where ϵ is a sufficiently small number. ILP problems assume variables to be integers or Booleans, with Boolean variables represented as integers that are assignable with a value of either 0 or 1. The complexity of solving an ILP problem is NP-hard (McDonald 2007).

4.2 Best Position Algorithm problem definition

The challenge of recommending a set of resources (*On-demand* or *Batch Resource Recommendation*) to each request can be posed as a problem of finding the best k resources in a set of ordered lists corresponding to the metrics that are being considered. In order to provide a recommendation ranking, we use the portfolio-based algorithm selection (Xu et al. 2011) as a strategy for obtaining the k most relevant items in a group of data (top-k technique), while considering multiple criteria. Akbarinia et al. (2011) have produced the BPA for processing the top-k queries from ordered data lists. This algorithm proposes the use of a mechanism for obtaining the results of the top-k in a more efficient manner than the Threshold Algorithm (TA) (Fagin et al. 2003). Thus, an improved and more efficient version of the algorithm, based on BPA, is presented, called BPA2 (Akbarinia et al. 2011). Accordingly, The authors of this paper present the general problem of answering top-k queries as follows: let D be a set of n resources, and L_1, L_2,..., L_m be m lists, such that each list L_i contains n pairs of the form (d, s_i(d)) where d∈D and s_i(d) is a non-negative real number that denotes the local score of d in L_i. Any data item d∈D appears only once in each list. Each list L_i is arranged in descending order of its local scores, hence it being called a "sorted list". The *Overall Score* of each data item d is computed as f(s_1(d), s_2(d), ..., s_m(d)) where f is a given scoring function. Thus, the *Overall Score* is the output of f where the input relates to the local scores of d in all lists. Akbarinia et al. (2011) assume that the scoring function is monotonic. The k data items, for which the *Overall Scores* are the highest among all data items, are called the top-k data items. Consequently, the problem addressed by the authors of this paper is the following: let L_1, L_2,..., L_m be m sorted lists, and D be the set of data items involved in the lists. Given a top-k query that involves a number k≤n and a monotonic scoring function f, our goal is to identify a set D'⊆D such as $|D'|$= k, and $\forall d_1 \in D'$ and $\forall d_2 \in (D \setminus D')$ in which the *Overall Score* of d_1 is greater than or equal to the *Overall Score* of d_2, while simultaneously minimising execution time.

We decided to use the BPA2 to solve the ranking problem, since the number of items generated in the lists can be extremely large. In the *Batch Resource Recommendation* use case, the number of items depends on the Cartesian product of all possible combinations of resources that can be allocated to the different resource requests (see subsection 6). In addition, the flexibility of our approach allows new metrics to be considered in the evaluation of the resources, and, therefore, the number of lists may also increase.

5 Definition of resource allocation metrics

During the resource allocation procedure, the person responsible may confront two challenges. The first challenge is the quantity of requests to which resources must be simultaneously allocated; specifically, whether this pertains to one request or multiple requests. The second challenge relates to the aforementioned decision scenarios and whether or not the framework proposes which resource to allocate (top-1), or whether it proposes a ranking of resources (top-k), so that the person responsible decides which resource to allocate. To address these challenges, we propose four use cases to undertake the resource allocation (explained in further detail in Section 6): 1) *On-demand Resource Allocation*: for a single request, a resource to be allocated is obtained; 2) *On-demand Resource Recommendation*: for a single request, a ranking of resources that can be allocated is produced; 3) *Batch Resource Allocation*: for a set of N requests, a resource to be allocated to each particular request is obtained; and 4) *Batch Resource Recommendation*: for a set of N requests, a ranking pertaining to the set of resources to be allocated to the set of requests is produced. In this section, we define the elements that characterize the Resource Allocation or Recommendation Request and describe the allocation criteria utilised. Furthermore, we present the resource process cube and the expertise matrices to support the measurement of the distinct allocation criteria. These criteria are evaluated by means of different metrics to produce an accurate resource allocation. While the objective of this paper is to both allocate and recommend resources for one or more requests, hereinafter the term ‘allocate’ shall be used in a generic manner, except in cases in which an explicit distinction between ‘allocate’ and ‘recommend’ are deemed necessary.

5.1 Resource request characterization description

The act of allocating or recommending a resource involves responding to a previously determined allocation request. Approaches proposed in the literature show that the process activity usually corresponds to the allocation unit used to allocate resource (Rinderle-Ma & van der Aalst 2007, Koschmider et al. 2011, Cabanillas et al. 2013, Liu et al. 2008, Huang, Lu & Duan 2011). Due to the flexibility of our approach, allocation can be undertaken on different allocation units, i.e., it can be undertaken at the activity level, at the sub-process level, or by considering the entire process as one single unit. For instance, let’s consider the example of the help-desk process (`HelpDesk`) of a company that provides support to products such as printers or servers. The process can be broken down according to two levels of interaction: first-contact level 1 and expert level 2. Each of these levels can be associated with a sub-process. To determine the type of association, the breakdown can be conducted manually by incorporating the semantics of the process, or via an automatic process decomposition (van der Aalst 2013), using techniques such as Single-Entry Single-Exit (SESE) (Munoz-Gama et al. 2014), or Passages (van der Aalst & Verbeek 2014). These techniques help to break down a process into smaller parts that can subsequently undergo independent analysis.

To produce a suitable allocation, it is necessary to characterise each request, i.e., determine which part of the process relates to the request and what the specific characteristics of the requirement are, since this characterization may affect the decision regarding which resource to allocate for execution.

Resource Request Characterization: Let $i = 1, \ldots, n$ be the desired request properties, while a Resource Request Characterization $c = (f_1, \ldots, f_n)$ is a multi-factor representation of the desired request properties that a resource must fulfill in order to be allocated to a given request, and where f_i is an element of a finite set that represents the feasible values for each request property.

In the examples used in this paper, the two-factor characterization proposed is a factor-tuple $c = (U, T)$, where U defines the unit where the resource is being requested, and T is the typology of the process execution instance that requests the resource.

The first factor relates to the level at which the allocation is required, for example first-contact level 1 or expert level 2. The second factor is a typology characterization, which helps to classify and utilise historical information according to the typology of the process required. For example, there can be distinct typologies, such as language (English/Spanish), type of client (normal/premium), or geographic region (the Americas/EMEA/APJ domains). In our case, we consider one typology: hardware, since requests can be printer-related or server-related issues. As such, and using the example of the `HelpDesk`, $c_0 = (level1, printer)$ and $c_2 = (level2, server)$, are two different characterizations for the help-desk process.

5.2 *Resource allocation criteria*

Regardless of the use of applications for managing processes and the proposal of different mechanisms for allocating resources (see Section 2), allocation remains a manual task (Zhao & Zhao 2014). In order to mitigate the need to rely on a single criterion when undertaking allocation, we propose six dimensions that use contextual and historical execution information of the process for allocating resources. These dimensions are as follows:

- Frequency dimension: this measures the rate of occurrence in which a resource has completed a request characterization. This metric captures the experience of each resource when executing a given activity, in the assumption that such experience will help to generate an improved performance (de Leoni et al. 2012).

- Performance dimension: this measures the execution time achieved by a resource while performing a request characterization.

- Quality dimension: this measures the customer evaluation of the execution of a request characterization performed by a resource.

- Cost dimension: this measures the execution cost of a request characterization performed by a resource.

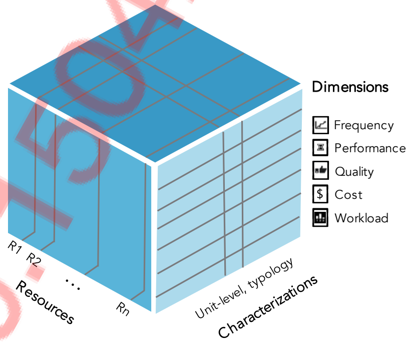
- Expertise dimension: this measures the ability level at which a resource is able to execute a request characterization.
- Workload dimension: this measures the actual idle level of a resource by considering the request characterizations executed at the time.

It can be observed that the flexible nature of the proposed framework allows for the inclusion of new dimensions, as well as its extension considering other dimensions proposed in the literature. We selected these dimensions in order to evaluate the resources, since information associated with these individuals is usually available in existing information systems. Moreover, this information can be interpreted and used quickly by those responsible for allocating resources.

5.3 Resource Process Cube

The term we use to describe the component that helps to manage the historical execution data of the process that requires analysis in order to calculate the metrics is 'knowledge base'. This knowledge base is abstracted in the resource process cube $\mathbb{Q}$, which is the semantic proposal that enables the resources to be evaluated. Thus, we are able to access subsets of information by means of basic OLAP operations, such as slice and dice (Agrawal et al. 1997).

Figure 3: Resource Process Cube



Resource Process Cube: Let r, c and d, be a resource, a Resource Request Characterization, and a dimension, respectively. A resource process cube $\mathbb{Q}[r][c][d]$ abstracts all historical information pertaining to resource r and the characterization c necessary to analyze the dimension d. Similarly, for a given dimension d, $\mathbb{Q}[\,][c][d]$ abstracts the historical information pertaining to all resources for the execution of the characterization c, and $\mathbb{Q}[r][\,][d]$ abstracts the information for all the characterizations performed by resource r.

Figure 3 shows the representation of the resource process cube $\mathbb{Q}$. For example, in the `HelpDesk` process, given a characterization $c_0 = (level1, printer)$ and resource $r_1 =$

$Mike$, $\mathbb{Q}[r_1][c_0][p]$ provides all the historical information pertaining to the performance of $Mike$ who is executing the characterization $(level1, printer)$, including the minimum and maximum time $Mike$ needed to perform c_0 (denoted as $\mathbb{Q}[r_1][c_0][p].min$ and $\mathbb{Q}[r_1][c_0][p].max$, respectively), or the average time required by $Mike$ to perform c_0 (denoted as $\mathbb{Q}[r_1][c_0][p].avg$). Similarly, $\mathbb{Q}[\,][c_0][p].max$ represents the maximum time required to perform c_0 considering all 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.

5.4 Expertise matrices

The expertise matrices help to represent the contextual information necessary to compare the level of expertise required to execute a request with the level of expertise of each resource. Using the competency model of the Human Resource Meta-Model (Oberweis & Schuster 2010) as a reference, expertise can be classified according to competencies, skills and knowledge. Based on this model, it is possible to define the expertise matrices as follows:

Resource Expertise Matrix: Let r and e be a resource and an expertise, respectively. A resource expertise matrix $\mathbb{E}_r[r][e]$ stores the expertise of a resource r on an expertise e (a specific competence, skill or knowledge). The value of $\mathbb{E}_r[\,][e]$ ranges from $\perp_e$ (usually, 0 indicates a lack of competence, skill or knowledge e) to $\top_e$ (complete expertise on the competence, skill or knowledge e).

Required Expertise Matrix: Let c and e be a Resource Request Characterization and an expertise, respectively. A required expertise matrix $\mathbb{E}_c[c][e]$ stores the desired level of expertise e required for performing a characterization c. The range of values of $\mathbb{E}_c[\,][e]$ is the same as $\mathbb{E}_r[\,][e]$; however $\mathbb{E}_c[c][e]$ is usually greater than $\perp_e$.

For example, given a characterization $c_2 = (level2, server)$, a resource $r_1 = Mike$ and an expertise $e_3 = server\ software\ knowledge$, $\mathbb{E}_c[c_2][e_3] = 4$ denotes a middle to high required level of expertise on e_3 (assuming $\perp e_3 = 1$ and $\top e_3 = 5$), while $Mike$ has low knowledge regarding server software, denoted as $\mathbb{E}_r[r_1][e_3] = 1$.

5.5 Resource allocation metrics

Regarding the dimensions previously described in subsection 5, metrics are proposed for evaluating the suitability of the resources for any given request. Table 1 shows these metrics and includes a description of each one. For the expertise dimension, it is necessary to determine how qualified a resource r is for executing a characterization c. Accordingly, we define two metrics that use the aforementioned expertise matrices. To measure the level of underqualification or overqualification of the resources, we compare the value of each level of expertise $\mathbb{E}_r[r][e]$ with the corresponding value in $\mathbb{E}_c[c][e]$. To determine the underqualification metric, it is first necessary to calculate the degree of underqualification (1), by comparing each value, as follows:

$$under(r,c,e) = \begin{cases} \frac{\mathbb{E}_c[c][e] - \mathbb{E}_r[r][e]}{\mathbb{E}_c[c][e] - \bot_e} & \text{if } \mathbb{E}_c[c][e] \geq \mathbb{E}_r[r][e] \;\; and \;\; \mathbb{E}_c[c][e] > \bot_e \\ 0 & \text{otherwise} \end{cases} \quad (1)$$

Similarly, to determine the overqualification metric, it is necessary to calculate the degree of overqualification (2):

$$over(r,c,e) = \begin{cases} \frac{\mathbb{E}_r[r][e] - \mathbb{E}_c[c][e]}{\top_e - \mathbb{E}_c[c][e]} & \text{if } \mathbb{E}_r[r][e] \geq \mathbb{E}_c[c][e] \;\; and \;\; \mathbb{E}_c[c][e] < \top_e \\ 0 & \text{otherwise} \end{cases} \quad (2)$$

These degrees of qualification are used to calculate the qualification metrics. In both qualification metrics, n_e represents the number of expertise elements in the matrix. Accordingly, we use the Euclidean distance, since all expertise features are equally relevant and defined according to the same scale, as well as favoring smaller differences in all features at the same time. It should be noted that if the expertise of a resource r perfectly matches with the expertise required for a characterization c, the value for both metrics shall be 1.

To calculate the other metrics for each resource r that executes a characterization c, we evaluate the resource process cube $\mathbb{Q}$ that represents the historical information of the process. All the metrics proposed are normalized between 0 and 1, and they satisfy the set of properties proposed in (Rozinat & van der Aalst 2005): *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). Henceforth, we shall refer to the metrics in a generic way as $metric_j(r,c)$. It should be noted that the use of the minimum, maximum and average in the definition of the metrics requires the presence of regular behaviour in terms of the resources during the execution of the process. For example, there cannot be a very large discrepancy in the performance of the resources while executing the different activities, and the event log cannot contain outliers. Therefore, prior to the application of the metrics, it is first necessary to verify that the event log contains reliable data. Otherwise, if the data in the event log is inaccurate (e.g., a resource omits to mark when an activity has been completed and hence the maximum duration of that resource when executing the respective activity is extensive), the use of the median in the metrics would be a more appropriate choice.
After having calculated each metric, we produce the *Overall Score* for every resource. Efforts were taken to maximise this *Overall Score* in order to determine the most suitable resources to be allocated.

Overall Score: Let c and r be a resource characterization request and a resource, respectively. Let $metric_j(r,c)$ be the metric *j* obtained by resource *r* when assigned to the characterization *c*, and w_j be the weighted value assigned to the metric *j*. The *Overall Score*, $score(r,c)$, is the weighted value of the different m metrics, as follows:

$$score(r,c) = \sum_{j=1}^{m} metric_j(r,c) * w_j \quad (3)$$

We use the weighted sum to define the *Overall Score*, considering that all metrics were normalised between 0 and 1. As mentioned above, we also assume the data in the event

Table 1: Proposed resource allocation metrics, based on (Arias et al. 2015)

Description	Metric
$\mathbb{Q}[r][c][f].total$: number of times a resource r has performed the characterization c. $\mathbb{Q}[\,][c][f].total$: number of cases of characterization c.	Frequency_Metric(r,c)= $\frac{logarithm(\mathbb{Q}[r][c][f].total)+1}{logarithm(\mathbb{Q}[\,][c][f].total)+1}$
$\mathbb{Q}[r][c][p].avg$: average duration, resource r executing characterization c. $\mathbb{Q}$[ ][c][p].min and $\mathbb{Q}$[ ][c][p].max, the minimum and maximum duration executing characterization c.	Performance_Metric(r,c) = $\frac{\mathbb{Q}[\,][c][p].max-\mathbb{Q}[r][c][p].avg}{\mathbb{Q}[\,][c][p].max-\mathbb{Q}[\,][c][p].min}$
$\mathbb{Q}[r][c][q].avg$: average quality, resource r executing characterization c. $\mathbb{Q}$[ ][c][q].min and $\mathbb{Q}$[ ][c][q].max, the minimum and maximum quality evaluation for the executed characterization c.	Quality_Metric(r,c) = $\frac{\mathbb{Q}[r][c][q].avg-\mathbb{Q}[\,][c][q].min}{\mathbb{Q}[\,][c][q].max-\mathbb{Q}[\,][c][q].min}$
$\mathbb{Q}[r][c][co].avg$: average cost, resource r executing characterization c. $\mathbb{Q}$[ ][c][co].min and $\mathbb{Q}$[ ][c][co].max the minimum and maximum cost for the executed characterization c.	Cost_Metric(r,c) = $\frac{\mathbb{Q}[\,][c][co].max-\mathbb{Q}[r][c][co].avg}{\mathbb{Q}[\,][c][co].max-\mathbb{Q}[\,][c][co].min}$
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 or the over qualification degree.	UnderQualification_Metric(r,c)=1 - $\frac{1}{n_e}\sqrt{\sum_{e=1}^{n_e}\left(under(r,c,e)\right)^2}$
	OverQualification_Metric(r,c) = 1 - $\frac{1}{n_e}\sqrt{\sum_{e=1}^{n_e}\left(over(r,c,e)\right)^2}$
$\mathbb{Q}$[r][ ][w].total: number of cases in which a resource r is working when a new resource allocation request is required. $\mathbb{Q}$[r][ ][w].top and $\mathbb{Q}$[r][ ][w].bottom be the maximum and minimum number of cases that a resource can attend simultaneously.	Workload_Metric(r,c) = $\frac{\mathbb{Q}[r][\,][w].top-\mathbb{Q}[r][\,][w].total}{\mathbb{Q}[r][\,][w].top-\mathbb{Q}[r][\,][w].bottom}$

log to be reliable and to reflect the regular behaviour of the resources when performing the process, thereby preventing the influence of outliers on the metrics.

One of the objectives of this paper is to ensure that multiple requests can be allocated simultaneously. Accordingly, these metrics can be generalised to measure the allocation of a tuple of resources to a tuple of Resource Request Characterizations:

Average Metric: Let **tc** and **tr** be an n-tuple of Resource Request Characterizations and an n-tuple of resources, respectively. For a given metric j, the average metric, $avg\_metric_j(\mathbf{tr}, \mathbf{tc})$ is defined in (4), as follows:

$$avg\_metric_j(\mathbf{tr}, \mathbf{tc}) = \frac{1}{n}\sum_{i=1}^{n} metric_j(tr_i, tc_i) \quad (4)$$

Standard Deviation Metric: Let **tc** and **tr** be an n-tuple of Resource Request Characterizations and an n-tuple of resources, respectively. For a given metric j, the standard deviation metric, $sd\_metric_j(\mathbf{tr}, \mathbf{tc})$ is defined in (5), as follows:

$$sd\_metric_j(\mathbf{tr}, \mathbf{tc}) = \sqrt{\frac{1}{n}\sum_{i=1}^{n} \left[metric_j(\mathbf{tr}_i, \mathbf{tc}_i) - avg\_metric_j(\mathbf{tr}, \mathbf{tc})\right]^2} \quad (5)$$

The standard deviation metric enables the measurement of the dispersion of a given metric among all resources that are being allocated, at the same time as measuring a set of Resource Request Characterizations. If two allocations have the same average metric, our preference is for the allocation to have a lower standard deviation metric.

Average Overall Score: Let **tc** and **tr** be an n-tuple of Resource Request Characterizations and an n-tuple of resources, respectively. The average Overall Score, $score(\mathbf{tr}, \mathbf{tc})$, is defined in (6), as follows:

$$score(\mathbf{tr}, \mathbf{tc}) = \frac{1}{n}\sum_{i=1}^{n} score(tr_i, tc_i) \quad (6)$$

In our approach, we consider that a resource can perform multiple activities at the same time, although we recognise that this workload is limited (e.g., a resource can handle up to four activities simultaneously). In a situation in which a resource can only execute one activity at a time, while the amount of resources is equal to the number of activities, the issue becomes a classic allocation problem, for which polynomial time algorithms exist. For example, it is possible to use the Kuhn-Munkres algorithm (Kuhn 1955, Munkres 1957), also known as the Hungarian algorithm, to solve this problem.

6 Resource allocation and recommendation use cases

As previously explained, it is possible to identify two challenges regarding resource allocation. The first relates to whether to undertake the allocation upon the arrival of each request, on an individual basis, or at scheduled times (e.g., every morning and every afternoon), whereby all accumulated requests must be resolved simultaneously. The second relates to the aforementioned decision scenarios in which it may be the case that only one resource needs to be allocated to each request, or rather, that a ranking of resources needs to be proposed for each request. On this basis, we define the following four use cases:

6.1 On-demand resource allocation

The *On-demand Resource Allocation* occurs when there is only one request and only one resource that requires allocation.

On-demand Resource Allocation: Let c be a Resource Request Characterization and R be a set of resources that can be allocated to c. An *On-demand Resource Allocation* allocates a resource r_{max} to c such that $score(r_{max}, c) = max_{r \in R}(score(r, c))$.

To resolve this problem, the *Overall Score*, $score(r, c)$, of each resource r is calculated and the greatest value is identified. For example, for the characterization $c_0 = (level1, printer)$, given a similar weight (50%) for all metrics, the framework allocates: *Top 1: (R19) score: 0.725.*

6.2 On-demand resource recommendation

The *On-demand Resource Recommendation* occurs when the recommendation of a ranking of resources is generated for a single request. By means of the method based on BPA2, a ranking is generated that shows the $Top - k$ resources recommended to resolve a given request.

On-demand Resource Recommendation: Let c be a Resource Request Characterization and R be a set of resources that can be allocated to c. An *On-demand Resource Recommendation* proposes a set of resources $R' \subseteq R$ that can be assigned to c, such that $|R'| = k$, and $\forall r_1 \in R'$ and $\forall r_2 \in (R \setminus R')$ $score(r_1, c) >= score(r_2, c)$.

BPA2 is used to solve this problem. The lists considered in BPA2 correspond to L_1, ..., L_m, being $j = 1, ..., m$ the different metrics considered. In general, if the j-th metric is $metric_j(r, c)$, L_j contains $|R|$ pairs of the form $(r, metric_j(r, c) * w_j)$ where $r \in R$ and w_j is the weighted value given to the metric j. L_j is sorted in descending order by the value of $metric_j(r, c) * w_j$.

For example, for the characterization $c_0 = (level1, printer)$, $top - k = 3$, and given the same weight (50%) for all metrics, the final recommendation suggests the ranking as follows: *Top 1: (R19) score: 0.725*; *Top 2: (R03) score: 0.712*; and *Top 3: (R02) score: 0.675*.

6.3 Batch Resource Allocation

The *Batch Resource Allocation* occurs when there is a set (Batch) of requests that needs to be resolved simultaneously and a single resource is allocated to each request. In this case, we use a method based on ILP, which helps to optimise the use of resources. This approach works by allocating the set of resources that provide the best average *Overall*

Score when allocated to the corresponding requests.

Batch Resource Allocation: Let C be a set of Resource Request Characterizations and R be a set of resources that can be allocated to each request $c \in C$. A *Batch Resource Allocation* allocates a tuple of $|C|$ resources to the set of requests C such that they are a solution to the following ILP problem:

$$Max \sum_{r \in R,\ c \in C} X_{rc} * score(r, c) \tag{7}$$

subject to:

$$\forall c \in C, \sum_{r \in R} X_{rc} = 1 \tag{8}$$

$$\forall r \in R, \sum_{c \in C} X_{rc} <= \mathbb{Q}[r][\,][w].top - \mathbb{Q}[r][\,][w].total \tag{9}$$

where:

- $X_{rc} \in [0, 1]$ is a binary variable that indicates the allocation of resource r to request c.
- $score(r, c)$ is the *Overall Score* obtained by resource r if allocated to request c.
- $\mathbb{Q}$[r][ ][w].top is the maximum number of requests that resource r can attend to simultaneously.
- $\mathbb{Q}$[r][ ][w].total is the number of cases on which resource r is working when a new Resource Allocation Request is required.

Equation (8) establishes that only one resource r can be allocated to each request c, whereas Equation (9) establishes that the quantity of requests allocated to a resource r cannot exceed its maximum workload, considering the requests that the resource is already handling.

For example, if there are two requests $c_0 = (level1, printer)$ and $c_2 = (level2, server)$, and given the same weight (50%) for all metrics, the following allocation is suggested: *Top 1: (R03, R19) score: 0.59*.

6.4 Batch resource recommendation

The *Batch Resource Allocation* occurs when there is a set (Batch) of requests that needs to be resolved simultaneously, and a recommendation of the ranking of resources is generated to provide alternatives for allocation to the proposed requests.

Batch resource recommendation: Let $\mathbf{tc} \in C^n$ be a tuple of n Resource Request Characterizations and the Cartesian power R^n be the set of all n-tuples of R resources that can be allocated to the requests in **tc**. A *Batch Resource Recommendation* proposes a set of tuples of resources $TR' \subseteq R^n$, such as $|TR'| = k$ and $\forall \mathbf{tr'} \in TR'$ and $\forall \mathbf{tr''} \in (R^n \setminus TR')$ $score(\mathbf{tr'}, \mathbf{tc}) >= score(\mathbf{tr''}, \mathbf{tc})$. Each tuple $\mathbf{tr} \in TR'$ represents the allocation of a resource $tr_i \in \mathbf{tr}$ to a request $tc_i \in \mathbf{tc}$, $\forall i = 1, ..., n$.

A method based on BPA2 algorithm is used to solve this problem. In this case, the elements of the lists were defined for each tuple $\mathbf{tr} \in R^n$. The lists considered were avg_L_1, ..., $avg\_L_m, dsp\_L_1, ..., dsp\_L_m$, being $j = 1, ..., m$ being the different metrics considered. The first m lists store the weighted average of each metric, while the last m lists store the weighted dispersion of each metric.

In general, if the j-th average metric is $avg\_metric_j(\mathbf{tr}, \mathbf{tc})$, avg_L_j contains $|R^n|$ pairs of the form $(\mathbf{tr}, avg\_metric_j(\mathbf{tr}, \mathbf{tc}) * w_j * avg\_w_j)$ where $\mathbf{tr} \in R^n$, and w_j is the weighted value assigned to the metric j. L_j is arranged in descending order by the value of $avg\_metric_j(\mathbf{tr}, \mathbf{tc}) * w_j * avg\_w_j$.

We shall use $1 - sd\_metric_j(\mathbf{tr}, \mathbf{tc})$ to measure the dispersion of a metric among all allocations. Therefore, dsp_L_j contains $|R^n|$ pairs of the form $(\mathbf{tr}, (1 - sd\_metric_j(\mathbf{tr}, \mathbf{tc})) * w_j * dsp\_w_j)$ where $\mathbf{tr} \in R^n$ and w_j is the weighted value given to the metric j. dsp_L_j is sorted in descending order by the value of $(1 - sd\_metric_j(\mathbf{tr}, \mathbf{tc})) * w_j * dsp\_w_j$.

The specified weight for each metric w_j is broken down so that a portion of this weight is designated to the corresponding avg_L_j, while the other portion is designated to the respective dsp_L_j, in such a way that $avg\_w_j + dsp\_w_j = 100\%$. In this paper, we use $avg\_w_j = 90\%$ and $dsp\_w_j = 10\%$ for all metrics.

In the `HelpDesk` example, regarding two requests with a characterization $c_0 = (level1, printer)$ and $c_2 = (level2, server)$, $top - k = 3$, and by applying the same weight (50%) for all metrics, the final ranking produced as a recommendation is as follows: *Top 1: (R03, R19) score: 0.823*, *Top 2: (R15, R04) score: 0.812*, and *Top 3: (R04, R15) score: 0.810*.

6.5 Relationship among the four use cases

In this section, we define the four use cases proposed in this paper. It should be noted that the *On-demand Resource Allocation* use case is a case particular to the *Batch Resource Allocation* use case, when the number of requests is 1. Similarly, the *On-demand Resource Recommendation* use case is a case particular to the *Batch Resource Recommendation* use case, when the number of requests is 1. However, each use case is particular due to their own unique complexities and, therefore, the method for solving each one is different. Subsequently, we explain the differences between the methods used for solving each use case.

For the *Batch Resource Allocation* use case, an optimisation problem is formulated and thereafter solved using ILP. Conversely, the method for solving the *On-demand Resource Allocation* use case is more straightforward, since it simply requires evaluating the available resources according to the metrics, calculating the *Overall Score* of each resource, and then selecting the resource with the highest *Overall Score*. It is a similar scenario in the remaining two cases. The method proposed for solving the *On-demand Resource Recommendation* use case is more straightforward. In this method, it is only necessary to evaluate the metrics for each resource applied to a single request and, based on this evaluation, to generate the top-k ranking using BPA2. Alternatively, for the *Batch Resource Recommendation* use case, it is first necessary to calculate the Cartesian product of all possible combinations of the available resources. Subsequently, it is necessary to evaluate the metrics for each resource, prior to evaluating the dispersion of each of the metrics. Only thereafter is it possible to apply BPA2.

7 Validation experiments

The framework presented in this paper was implemented in two plug-ins (Arias et al. 2016) within ProM (Verbeek et al. 2010), which is an open-source framework that allows for a standardised implementation of process mining techniques and tools.

To use the first plug-in, called *Generate Resource Knowledge*, we have first to import the information (contextual and historical) required for the necessary decision-making within the framework. For this purpose, we created a standardised format to store all considered information. We created an extension to the Java OpenXES library (https://svn.win.tue.nl/trac/prom/browser/Packages/ResourceRecommendation/Trunk/src/org/processmining/resourcerecommendation/utils/resrecxes) that helped to standardise and manipulate the information used to generate the required knowledge base. The historical information of the process abstracted in the resource process cube $\mathbb{Q}$ and the expertise matrices were utilized to calculate the metrics for each dimension, helping to generate the knowledge base as a result.

The second plug-in, called *Recommend Resources*, considered the knowledge base and the requests entered into the system. Subsequently, the plug-in generated the final allocation or recommendation, having assessed the specified weights for each metric. Both plug-ins are available in ProM Nightly-Builds (www.promtools.org/prom6/nightly), within the *ResourceRecommendation* package.

7.1 Empirical evaluation

To evaluate the efficiency of our approach, we conducted a series of controlled experiments. The design of the experiments undertaken is described below, while subsection 7.2 provides an analysis of the results obtained.

We used the `HelpDesk` process to evaluate our approach. The overall purpose was to assess the efficacy of the proposed solution with regard to the quality of the recommendations generated and the performance of the algorithms. We considered of the four use cases described in Section 6 to evaluate the behaviour of the framework, undertaking the resource allocation using the method based on linear programming, or by generating a ranking using BPA2. To characterize each resource request, we proposed two factors: 1) $UnitFactor$: level 1 or level 2, which are units that arrange the activities corresponding to the first or second level of `HelpDesk` attention; and 2) $TypologyFactor$, which has two typologies: printers and servers.
In the experiments, we considered different quantities of requests, resources and solicited recommendations. Four sets of experiments were conducted. In the first three sets, BPA2 method was used, while the allocation method based on ILP was used in the final set.

- **Experiment Set 1:** the purpose of this set of experiments was to provide evidence regarding the ability of the framework to correctly make recommendations, by considering each metric in an independent manner. In each test, the selected metric had a weight of 100%, compared to 0% for the others. Four allocation requests, 20 resources, and $top - k = 3$ were considered.

- **Experiment Set 2:** the purpose of this set of experiments was to provide evidence regarding the ability of the framework to correctly make recommendations when allocating different weights to each metric. Accordingly, three distinct scenarios were represented for three distinct types of company: large, medium and small. Different weights were established for each metric according to the priorities of each company. Four allocation requests, 20 resources, and $top - k = 3$ were considered.

- **Experiment Set 3:** the purpose of this set of experiments was to analyse the performance of the recommendation algorithm when modifying the three dimensions of the problem: 1) quantity of requests (1, 2, 3, 4); 2) quantity of resources (5, 10, 15, 20); and 3) $top - k$ requested recommendations (1, 3, 5, 7, 9).

- **Experiment Set 4:** the purpose of this set of experiments was to analyse the performance of the allocation algorithm when modifications were made to two dimensions of the problem: 1) quantity of requests (1, 2, 3, 4, 10, 20); and 2) quantity of resources (5, 10, 15, 20).

Regarding the workload, it was assumed that in all experiments, each resource had an upper limit of cases that could be handled at any one time, as well as a maximum quantity of cases that could be handled simultaneously at the moment the request was made. Consequently, each resource was only considered for requests for which he/she was available to allocate. The attributes for each case in the log included: Case ID, Unit Level, Process Typology, Resource, Creation Date, Closing Date, Cost, and Customer Satisfaction (Quality).

All experiments were conducted on a computer with an Intel Core i5 (1.80GHz) processor, 8 GB of RAM, and a 64-bit operating system. Prior to conducting the experiments, we evaluated the creation performance of the knowledge base used to allocate resources. For this evaluation, we assessed logs with 100, 1,000, 10,000, 50,000 and 100,000 cases, while the quantity of resources used for each one was 20. As expected, the measurements indicated that when a greater amount of information was available in the log, a longer period of time was required to generate the knowledge base. However, it is possible to state that the procedure for devising the resource process cube $\mathbb{Q}$ was executed quickly. For example, it took 0.4 seconds to process information from 50,000 cases, and 0.55 seconds to process 100,000 cases. 10,000 cases were considered as a baseline in conducting the aforementioned experiments.

7.2 *Obtained results and analysis*

Findings from the empirical evaluation results enabled us to identify certain contributions. Our framework is capable of generating the allocation or recommendation of resources for one or multiple requests, using the two methods presented. A number of criteria can be considered to evaluate the suitability of the resources, in addition to the effective availability of each resource. The method based on ILP allows for a simple and fast allocation of resources, which increases on a linear basis as the quantity of requests or resources rises. Alternatively, the recommendation method based on BPA2 allows for a ranking of resources to be generated in order to guide the decision-making process of the person responsible for undertaking the allocation.

Table 2 shows the parameters and results from Experiments 1 and 2; Table 3 shows the parameters and results from Experiment 3; and Table 4 shows the parameters and results those from Experiment 4. In Experiments 1 and 2, we simulated an event log that included three resources (R04, R05, R06), the participation frequency of whom, in terms of resolving cases in the `HelpDesk`, was greater compared to the other resources operating at the same level of attention. Similarly, we simulated the presence of three resources who took less time to resolve the cases (R01, R02, R03), as well as three resources that did so to a greater degree of quality (R07, R08, R09), three resources that entailed lower costs (R18, R19, R20), three resources that precisely met the level of expertise required (R12, R13, R14), and three resources with a greater availability to respond to cases (R12, R17, R19).

In **Experiment Set 1**, the resource recommendation for four requests was sought, all four with the characterization $c_0 = (level1, printer)$. It is possible to observe in Table 2 that the recommended resources for each sub-experiment were as expected, demonstrating the suitability of the proposed metric for each dimension and the efficacy of the recommendation algorithm.

In **Experiment Set 2**, different weights were specified in accordance with the preferences of each company; the large company, experiment (exp.) 2.1, was interested in fast and low-cost solutions; the small company, exp.2.2, preferred criteria relating to quality and the required level of expertise for executing the requests; and the medium-sized company, exp.2.3, chose an approach based on all dimensions consisting of the same weight. As Table 3 shows, the multi-criteria approach produced different resource recommendations based on the criteria established in each case. Given the presence of three distinct companies with their own particular preferences, and using the same scenario, different results were generated for each example.

In **Experiment Set 3**, we varied the quantity of requests (exp.3.1.1 to exp.3.1.4), the quantity of available resources (exp.3.2.1 to exp.3.2.4), and the $top-k$ (exp.3.3.1 to exp.3.3.5). We included characterizations of type $c_0 = (level1, printer)$ and type $c_1 = (level1, server)$. Our approach generated a ranking of feasible resource allocations for the tuple of requests, having assessed the resources according to the request characterization, the weights given, and the availability of each resource at the moment in which the recommendation was undertaken. For each case, a time performance analysis was undertaken. In Experiment 3.1 (Figure 4a), an exponential relationship between processing time and the quantity of requests was observed. As the quantity of requests rise, the processing time increased exponentially; a limiting factor when a large quantity of requests requires to be processed simultaneously. This occurred as a result of using the Cartesian product of resource combinations, prior to applying the allocation method based on BPA2. Alternatively, in Experiment 3.2 (Figure 4b), a linear relationship between processing time and the quantity of resources used to generate the recommendation was observed. Furthermore, in Experiment 3.3 (Figure 4c), the algorithm BPA2 was shown to enable the generation of rankings with different $top-k$ without impacting on the processing time, i.e., the latter remained constant across all cases. This was possible due to the threshold management and the early stop condition of BPA2 (Akbarinia et al. 2011).

In **Experiment Set 4**, we varied the quantity of requests (exp.4.1.1 to exp.4.1.6), and the quantity of resources (exp.4.2.1 to exp.4.2.4) to analyze the behavior of the ILP method. It was shown that modifying the quantity of requests or the quantity of resources can lead to a variation in resource allocation. A linear relationship was also observed between processing time and the quantity of requests (Figure 5a), as well as between processing

Table 2: Resource recommendations for experiments 1 and 2 with BPA2 method

Exp.	Weights (%)	nRes	nReq	Top-k	Recommendation					Overall	Time (sec)
					Requests#(c_0	c_0	c_0	c_0)			
1.1	F:100 - others:0				Top-1:	R04	R05	R06	R05	1.14	
					Top-2:	R04	R06	R05	R05		
					Top-3:	R05	R06	R06	R04		
1.2	P:100 - others:0				Top-1:	R01	R03	R02	R03	0.95	
					Top-2:	R03	R01	R02	R03		
					Top-3:	R01	R02	R03	R03		
1.3	Q:100 - others:0	20	4	3	Top-1:	R08	R09	R07	R09	0.90	8.8
					Top-2:	R09	R08	R07	R09		
					Top-3:	R07	R09	R08	R09		
1.4	C:100 - others:0				Top-1:	R18	R19	R19	R19	1.05	
					Top-2:	R19	R18	R19	R19		
					Top-3:	R18	R19	R19	R19		
1.5	U:100 - O:100 - others:0				Top-1:	R12	R13	R14	R12	2.24	
					Top-2:	R13	R12	R14	R12		
					Top-3:	R12	R14	R13	R12		
1.6	W:100 - others:0				Top-1:	R12	R19	R17	R17	1.19	
					Top-2:	R12	R17	R19	R17		
					Top-3:	R17	R12	R19	R17		
2.1	F:010 - P:050 - Q:010 C:100 - U:015 - O:000 W:010				Top-1:	R18	R02	R01	R03	1.24	
					Top-2:	R18	R01	R03	R02		
					Top-3:	R18	R01	R02	R03		
2.2	F:025 - P:015 - Q:100 C:030 - U:075 - O:065 W:010	20	4	3	Top-1:	R07	R08	R09	R07	2.2	8.8
					Top-2:	R07	R09	R08	R07		
					Top-3:	R08	R07	R09	R07		
2.3	F:050 - P:050 - Q:050 C:050 - U:050 - O:050 W:050				Top-1:	R12	R17	R19	R17	2.28	
					Top-2:	R12	R19	R17	R17		
					Top-3:	R17	R12	R19	R17		

F= Frequency, P= Performance, Q= Quality, C= Cost, U= Underqualified, O= Overqualified, W= Workload R= Resource and others= other dimensions

Table 3: Resource recommendations for experiment 3 with BPA2 method

Exp.	Weights (%)	nRes	nReq	Top-k	Recommendation	Overall	Time (sec)
					Requests# (c_0)		
3.1.1	F:050 - P:050 - Q:050 C:050 - U:050 - O:050 W:050		1		Top-1: R17	2.20	
					Top-2: R19	2.18	0.06
					Top-3: R12	2.17	
					Requests#(c_0 c_1)		
3.1.2	F:050 - P:050 - Q:050 C:050 - U:050 - O:050 W:050		2		Top-1: R17 R19	2.31	
					Top-2: R12 R19	2.30	0.27
					Top-3: R12 R12	2.29	
					Requests#(c_0 c_1 c_0)		
3.1.3	F:050 - P:050 - Q:050 C:050 - U:050 - O:050 W:050	20	3	3	Top-1: R17 R19 R12	2.30	
					Top-2: R12 R19 R17	2.30	1.8
					Top-3: R17 R12 R19	2.29	
					Requests#(c_0 c_1 c_0 c_1)		
3.1.4	F:050 - P:050 - Q:050 C:050 - U:050 - O:050 W:050		4		Top-1: R17 R19 R12 R02	2.27	
					Top-2: R17 R02 R12 R19		8.8
					Top-3: R12 R19 R17 R02		
					Requests#(c_0 c_1 c_0 c_1)		
3.2.1	F:050 - P:050 - Q:050 C:050 - U:050 - O:050 W:050	5			Top-1: R01 R02 R03 R02	2.41	
					Top-2: R01 R02 R01 R02		0,9
					Top-3: R03 R02 R03 R01		
3.2.2	F:050 - P:050 - Q:050 C:050 - U:050 - O:050 W:050	10			Top-1: R01 R02 R03 R02		
					Top-2: R03 R02 R01 R02	2.08	3.6
					Top-3: R03 R02 R01 R07		
3.2.3	F:050 - P:050 - Q:050 C:050 - U:050 - O:050 W:050	15	4	3	Top-1: R12 R02 R01 R03	2.25	
					Top-2: R12 R03 R01 R02		6.7
					Top-3: R01 R03 R12 R02		
3.2.4	F:050 - P:050 - Q:050 C:050 - U:050 - O:050 W:050	20			Top-1: R17 R19 R12 R02	2.27	
					Top-2: R17 R02 R12 R19		8.8
					Top-3: R12 R19 R17 R02		
					Requests#(c_0 c_1 c_0 c_1)		
3.3.1	F:050 - P:050 - Q:050 C:050 - U:050 - O:050 W:050			1	Top-1: R17 R19 R12 R02	2.2777	8.8
3.3.2	F:050 - P:050 - Q:050 C:050 - U:050 - O:050 W:050				Top-1: R17 R19 R12 R02	2.2777	
				3	Top-2: R17 R02 R12 R19	2.2777	8.8
					Top-3: R12 R19 R17 R02	2.2777	
3.3.3	F:050 - P:050 - Q:050 C:050 - U:050 - O:050 W:050				Top-1: R17 R19 R12 R02	2.2777	
					Top-2: R17 R02 R12 R19	2.2777	
		20	4	5	Top-3: R12 R19 R17 R02	2.2777	8.8
					Top-4: R12 R02 R17 R19	2.2777	
					Top-5: R17 R19 R12 R01	2.2772	
3.3.4	F:050 - P:050 - Q:050 C:050 - U:050 - O:050 W:050				Top-1: R17 R19 R12 R02	2.2777	
					Top-2: R17 R02 R12 R19	2.2777	
					Top-3: R12 R19 R17 R02	2.2777	
				7	Top-4: R12 R02 R17 R19	2.2777	8.8
					Top-5: R17 R19 R12 R01	2.2772	
					Top-6: R17 R01 R12 R19	2.2772	
					Top-7: R12 R19 R17 R01	2.2772	
3.3.5	F:050 - P:050 - Q:050 C:050 - U:050 - O:050 W:050				Top-1: R17 R19 R12 R02	2.2777	
					Top-2: R17 R02 R12 R19	2.2777	
					Top-3: R12 R19 R17 R02	2.2777	
					Top-4: R12 R02 R17 R19	2.2777	
				9	Top-5: R17 R19 R12 R01	2.2772	8.8
					Top-6: R17 R01 R12 R19	2.2772	
					Top-7: R12 R19 R17 R01	2.2772	
					Top-8: R12 R01 R17 R19	2.2772	
					Top-9: R17 R19 R12 R17	2.2756	

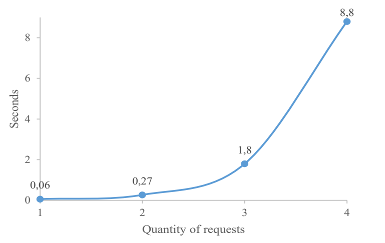
F= Frequency, P= Performance, Q= Quality, C= Cost, U= Underqualified, O= Overqualified, W= Workload and R= Resource

Table 4: Resource allocations for the experiments with ILP method

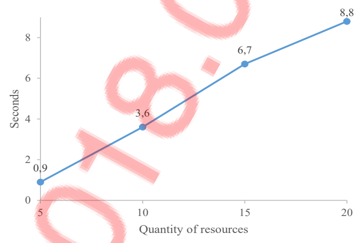
Exp.	Weights (%)	nRes	nReq	Allocation	Overall	Time (sec)
4.1.1	F:050 - P:050 - Q:050 C:050 - U:050 - O:050 W:050		1	**Requests#**(c_0) Top-1: R17	0.315	0.06
4.1.2	F:050 - P:050 - Q:050 C:050 - U:050 - O:050 W:050		2	**Requests#**(c_0 c_1) Top-1: R17 R19	0.314	0.085
4.1.3	F:050 - P:050 - Q:050 C:050 - U:050 - O:050 W:050		3	**Requests#**(c_0 c_1 c_0) Top-1: R17 R19 R17	0.314	0.094
4.1.4	F:050 - P:050 - Q:050 C:050 - U:050 - O:050 W:050	20	4	**Requests#**(c_0 c_1 c_0 c_1) Top-1: R17 R19 R17 R19	0.314	0.145
4.1.5	F:050 - P:050 - Q:050 C:050 - U:050 - O:050 W:050		10	**Requests#**(c_0 c_1 c_0 c_1 c_0 Top-1: R17 R19 R17 R12 R17 c_1 c_0 c_1 c_0 c_1) Top-1: R19 R12 R12 R17 R19	0.312	0.172
4.1.6	F:050 - P:050 - Q:050 C:050 - U:050 - O:050 W:050		20	**Requests#**(c_0 c_1 c_0 c_1 c_0 Top-1: R17 R19 R17 R02 R17 c_1 c_0 c_1 c_0 c_1 Top-1: R19 R12 R20 R17 R19 c_0 c_1 c_0 c_1 c_0 Top-1: R12 R03 R18 R20 R01 c_1 c_0 c_1 c_0 c_1) Top-1: R12 R07 R01 R02 R03	0.306	0.204
4.2.1	F:050 - P:050 - Q:050 C:050 - U:050 - O:050 W:050	5		**Requests#**(c_0 c_1 c_0) Top-1: R01 R02 R01	0.305	0.04
4.2.2	F:050 - P:050 - Q:050 C:050 - U:050 - O:050 W:050	10		**Requests#**(c_0 c_1 c_0) Top-1: R01 R02 R01	0.305	0.06
4.2.3	F:050 - P:050 - Q:050 C:050 - U:050 - O:050 W:050	15	3	**Requests#**(c_0 c_1 c_0) Top-1: R12 R02 R12	0.308	0.07
4.2.4	F:050 - P:050 - Q:050 C:050 - U:050 - O:050 W:050	20		**Requests#**(c_0 c_1 c_0) Top-1: R17 R19 R17	0.314	0.09

F= Frequency, P= Performance, Q= Quality, C= Cost, U= Underqualified, O= Overqualified, W= Workload and R= Resource

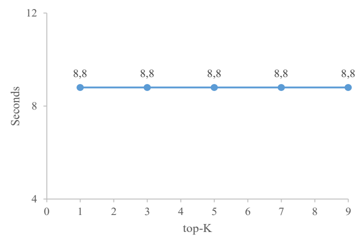
Figure 4: Performance analysis using the BPA2 method



(a) variation in quantity of requests

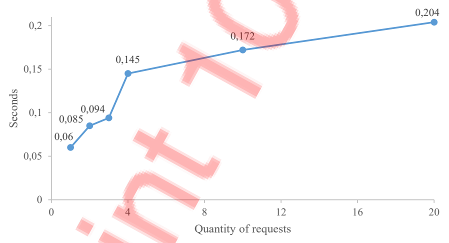


(b) variation in quantity of resources

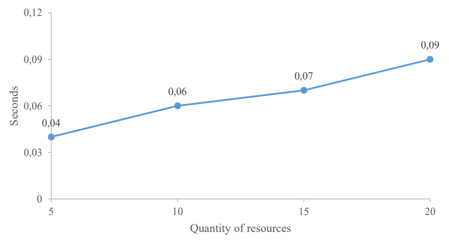


(c) variation in top-k

Figure 5: Performance analysis using the ILP method



(a) variation in quantity of requests



(b) variation in quantity of resources

time and the quantity of resources used to generate the allocation (Figure 5b).

8 Case study

The help-desk process we selected as a case study comes from the Costa Rican company Software & Consulting Group (SCG), a consultancy firm for business software solutions. This company offers an incident-response service to its clients (Figure 6).

Within the company, when an incident is received, it must be allocated to an agent for assessment, response, resolution and closure. Accordingly, an information system registers details relating to the process execution in an event log. For this case study, we considered the following data: Incident ID, Client Description, System Affected, Allocated Agent, Creation Date, End Date, Cost, Priority, Origin of the Problem, and Evaluation of the Service (Quality is evaluated using the following scale: 1: Insufficient; 2: Regular; 3: Good; 4: Very Good; 5: Excellent). We extracted the information in a log with a total of 1,778 cases, registered between August and November 2015. In addition, SCG utilizes contextual information regarding the level of expertise of the agents and the expertise required to resolve the different incidents. For example, it is necessary to have knowledge of the following tools in order to work on the help-desk process at SCG: Java, SQL Server, and .Net. We conducted our case study by considering the activities that correspond to the First Level Contact Center -Level 1- (incidents may be assigned to the other levels, which are viewed as additional sub-processes) of the help-desk. The allocation of resources within the process takes place at this level and is executed manually by a service assistant. We considered incidents reported on the DMS-One system, which is a management system originally designed for the automotive industry. In this system, the origin of the incidents is classified according to the following categories: System, User, SAP, Configuration, Update, and Others.

To perform an evaluation using the case study, the following considerations were taken into account:

- To generate the knowledge base, we used information from August to October 2015 (1,280 cases).
- For validation purposes, we used the requests created on 2 November 2015.
- The Resource Request Characterization includes the following factors: 1) $Unit\ level$: level 1; and 2) $Typologies$: system and incident origin. For example, the characterization $c00003$ relates to an incident type: $Level\ 1, DMS-One, SAP$.
- The expertise matrices were generated using the information related to the expertise of the company (the level of expertise is based on the following scale: 1: No Expertise; 2: Basic Expertise; 3: Intermediate Expertise; 4: Advanced Expertise).
- For the workload information, the historical participation of each resource in the process was considered in order to determine the maximum quantity of cases that can be handled in any given day. The open cases of each resource were computed based on the historical information, in order to assess his/her availability.

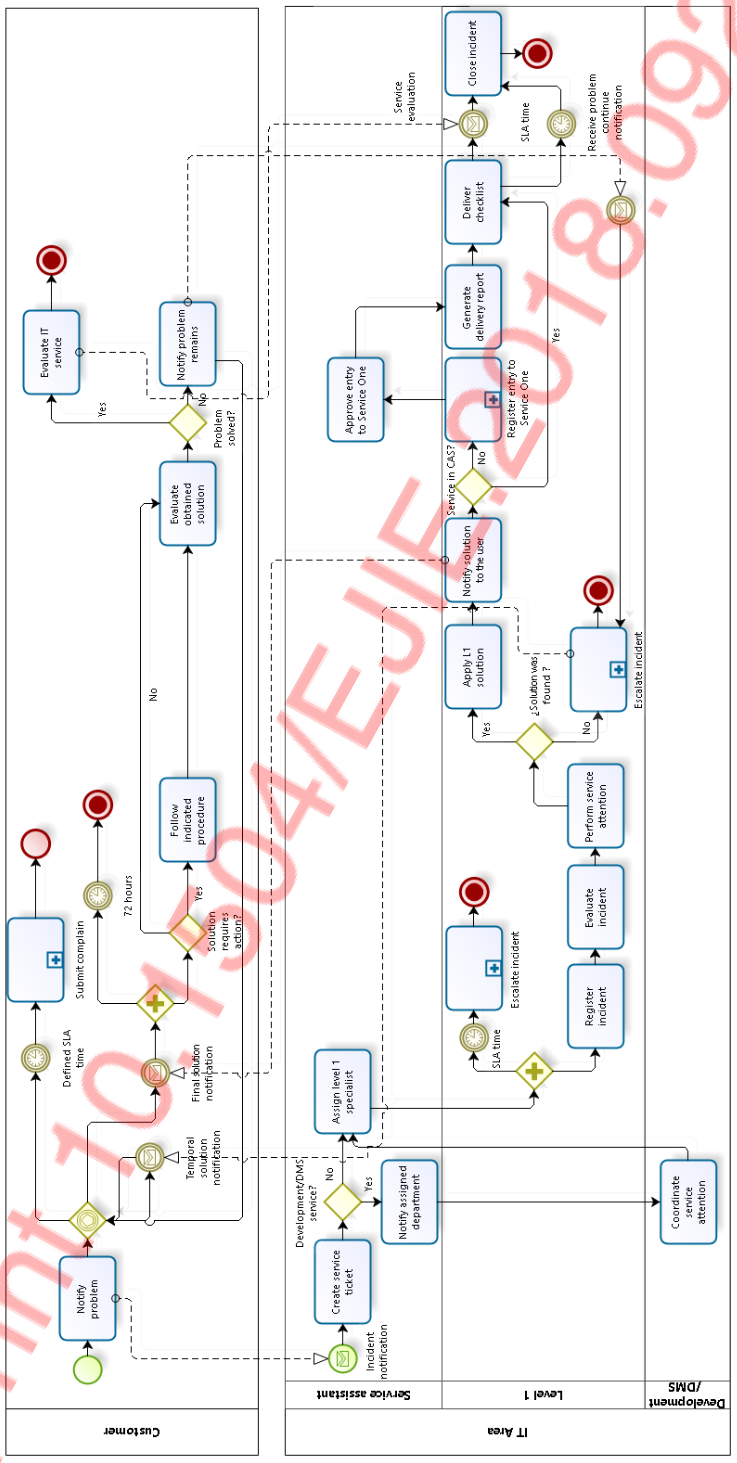
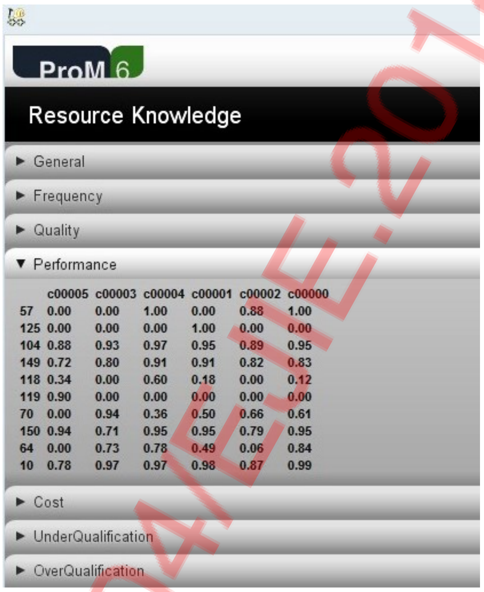


Figure 6: HelpDesk process implemented by the company

8.1 Results and evaluation

Figure 7 shows the results of using the *Generate Resource Knowledge* plug-in. This plug-in was used to process the historical and contextual information by applying the metrics in each dimension and generating the knowledge base used to produce the resource allocation.

Figure 7: Screenshot of the *Generate Resource Knowledge* plug-in upon processing information from the case study



	c00005	c00003	c00004	c00001	c00002	c00000
57	0.00	0.00	1.00	0.00	0.88	1.00
125	0.00	0.00	0.00	1.00	0.00	0.00
104	0.88	0.93	0.97	0.95	0.89	0.95
149	0.72	0.80	0.91	0.91	0.82	0.83
118	0.34	0.00	0.60	0.18	0.00	0.12
119	0.90	0.00	0.00	0.00	0.00	0.00
70	0.00	0.94	0.36	0.50	0.66	0.61
150	0.94	0.71	0.95	0.95	0.79	0.95
64	0.00	0.73	0.78	0.49	0.06	0.84
10	0.78	0.97	0.97	0.98	0.87	0.99

Table 5 Real allocation made on 2 November 2015

ID	Client	System	Agent	Creation date	End Date	Source	Quality
18121	C00208		104	02/11/2015 08:23	10/11/2015 08:51	C00001 User	5
18123	C00029		10	02/11/2015 09:11	02/11/2015 11:24	C00000 System	4
18125	C00231		10	02/11/2015 11:07	02/11/2015 11:09	C00000 System	5
18126	C00231		10	02/11/2015 11:09	02/11/2015 11:11	C00000 System	5
18127	C00231		10	02/11/2015 11:11	02/11/2015 11:12	C00000 System	5
18128	C00231		10	02/11/2015 11:12	02/11/2015 11:14	C00000 System	5
18129	C00231		10	02/11/2015 11:14	02/11/2015 11:15	C00000 System	5
18130	C00231		10	02/11/2015 11:16	02/11/2015 11:18	C00000 System	5
18131	C00171		10	02/11/2015 11:18	02/11/2015 11:19	C00001 User	5
18132	C00231	DMS-One	10	02/11/2015 11:19	02/11/2015 11:21	C00000 System	5
18133	C00229		10	02/11/2015 11:21	02/11/2015 11:22	C00000 System	4
18134	C00231		104	02/11/2015 12:11	02/11/2015 12:12	C00000 System	5
18135	C00231		104	02/11/2015 12:12	02/11/2015 12:13	C00000 System	5
18136	C00231		104	02/11/2015 12:13	02/11/2015 12:14	C00000 System	5
18137	C00231		104	02/11/2015 12:14	02/11/2015 12:14	C00000 System	5
18138	C00231		104	02/11/2015 12:15	02/11/2015 12:15	C00000 System	5
18142	C00231		104	02/11/2015 14:35	19/11/2015 15:28	C00004 Configuration	5
18145	C00231		104	02/11/2015 16:02	04/11/2015 12:25	C00002 Others	4

Table 5 shows partial information relating to an incident handled on 2 November 2015. This information summarizes the case details, including the real allocation undertaken by the company regarding each request, as well as client evaluation of the quality of service received from the company, which is conducted following the conclusion of each case. During the evaluation, the resources involved were available to execute a maximum of ten requests simultaneously.

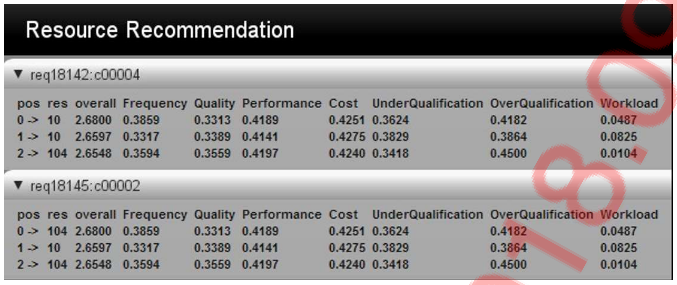
We compared the real allocation undertaken by the company with the results obtained using the two methods proposed by our approach. Results are shown in Table 6. The comparison was realized by separating the requests into four separate blocks of four requests and one block of two requests. Each block was resolved independently, i.e., a set of resources was allocated/recommended for the Batch of requests considered in each block, independently of other requests. As part of the evaluation, SCG established the same weight for all the metrics. Figure 8 shows the results generated by the *Recommend Resources* plug-in, with the recommendation stemming from the method based on BPA2 for two requests. Figure 9 shows the allocation obtained by the method based on ILP. Request 18142 had a characterization $c00004(Level1, DMS - One, Configuration)$, whereas that of request 18145 was $c00002(Level1, DMS - One, Others)$.

Table 6 Real allocations versus recommendations generated by the framework

ID	Real Resource Allocation	BPA2 Resource Recommendation	Time (sec)	ILP Resource Assignment	Time (sec)
18121	(104,10,10,10)	Top-1: (10,10,10,10)		(10,10,10,10)	
18123		Top-2: (125,10,10,10)	1.52		0.060
18125		Top-3: (150,10,10,10)			
18126					
18127	(10,10,10,10)	Top-1: (10,10,10,10)		(10,10,10,10)	
18128		Top-2: (150,10,10,10)	1.52		0.069
18129		Top-3: (10,150,10,10)			
18130					
18131	(10,10,10,104)	Top-1: (10,10,10,10)		(10,10,10,10)	
18132		Top-2: (104,10,10,10)	1.52		0.060
18133		Top-3: (125,10,10,10)			
18134					
18135	(104,104,104,104)	Top-1: (10,10,10,10)		(10,10,10,10)	
18136		Top-2: (150,10,10,10)	1.52		0.062
18137		Top-3: (10,150,10,10)			
18138					
18142	(104,104)	Top-1: (10,104)		(104,104)	
18145		Top-2: (104,104)	0.60		0.040
		Top-3: (104,10)			

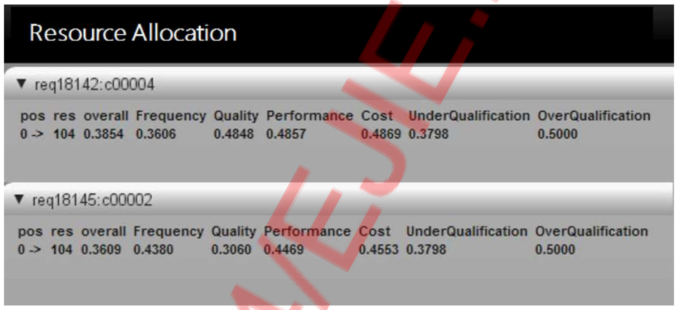
Table 6 shows that the method based on ILP generated results in a shorter period of time than the method based on BPA2. Overall, 66.7% of the allocations recommended by the ILP method concur with the allocations made by the company. This is compared to 61.1% using the BPA2 method. In cases in which there was no concurrence between the company and the results obtained by both our methods (33.3% and 38.9%, respectively), it was observed that the resource allocated by the company was agent 104, whereas the resource recommended by both our methods was agent 10.

Figure 8: Screenshot of the *Recommend Resources* plug-in, using the recommendation method based on BPA2



```
Resource Recommendation
▼ req18142:c00004
pos res overall Frequency Quality Performance Cost   UnderQualification OverQualification Workload
0 -> 10  2.6800 0.3859    0.3313  0.4189      0.4251 0.3624             0.4182            0.0487
1 -> 10  2.6597 0.3317    0.3389  0.4141      0.4275 0.3829             0.3864            0.0825
2 -> 104 2.6548 0.3594    0.3559  0.4197      0.4240 0.3418             0.4500            0.0104
▼ req18145:c00002
pos res overall Frequency Quality Performance Cost   UnderQualification OverQualification Workload
0 -> 104 2.6800 0.3859    0.3313  0.4189      0.4251 0.3624             0.4182            0.0487
1 -> 10  2.6597 0.3317    0.3389  0.4141      0.4275 0.3829             0.3864            0.0825
2 -> 104 2.6548 0.3594    0.3559  0.4197      0.4240 0.3418             0.4500            0.0104
```

Figure 9: Screenshot of the *Recommend Resources* plug-in, using the allocation method based on ILP



```
Resource Allocation
▼ req18142:c00004
pos res overall Frequency Quality Performance Cost   UnderQualification OverQualification
0 -> 104 0.3854 0.3606    0.4848  0.4857      0.4869 0.3798             0.5000
▼ req18145:c00002
pos res overall Frequency Quality Performance Cost   UnderQualification OverQualification
0 -> 104 0.3609 0.4380    0.3060  0.4469      0.4553 0.3798             0.5000
```

We considered two perspectives to evaluate the obtained results. First, we analysed the results from the perspective of the historical and contextual information. From this analysis, it was possible to determine that resource 10 had greater experience in executing similar cases in the past; the cases in which he/she had participated received more positive client evaluations, and his/her evaluation was greater in terms of performance and cost. Regarding expertise, resource 10 had a profile closer to the level of expertise required. As a second perspective, we incorporated the feedback provided by the owner of the company's help-desk process. To do so, we met with the process owner to show him our approach. Having reviewed the results generated by the framework and how they concurred with the real allocation, the process owner judged them to be correct and in line with the context of the allocation. For cases in which there was no concurrence, the allocation generated by our framework was accepted by the process owner. Nevertheless, he deemed that further dimensions could be incorporated within the framework, given its flexibility and extensiveness. For example, the concurrence would be greater if the complexity of the cases was also considered by the framework.

As well as evaluating the quality of the results obtained, the process owner analysed additional aspects of our approach. First, he highlighted the importance of being able to consider different dimensions and of adjusting the level of importance of each of these using the relevant weights. This approach would produce a diverse array of results and could be adjusted to fit with specific business contexts. Second, he stressed that it was possible to interpret and understand the dimensions used and the results produced from the framework in a clear and immediate manner. Third, he noted the ability to be able to undertake individual allocations/recommendations, as well as by blocks of requests, as a particularly relevant functionality of the framework. This is because the framework can be made to fit perfectly with real business scenarios in which there is a need for managing the allocation of resources to multiple cases simultaneously. For example, when the quantity of cases increases during the period of accounting close and at the end of administrative procedures (e.g., 30 cases require simultaneous resolution due to the large-scale use of the systems supported by the help-desk). Fourth, he asserted that the proposed approach allowed for the consideration of only those resources that were available when the allocation or recommendation was made, which was in line with changes recently implemented in the company business model. Nowadays, the pool of resources available varies in different circumstances, e.g., weekends, specific times of year, or the country in which the assistance is provided. Finally, the process owner agreed that the exponential growth of the processing time presented by the method based on BPA2 could be a limitation in scenarios in which the quantity of request increases; an aspect that requires improvement within this approach.

The results obtained in this real business scenario demonstrated the applicability of our framework for the dynamic allocation of resources. Thus, we have introduced an approach that can be adapted and used quickly by process owners across diverse resource allocation contexts. The use of historical and contextual information allows metrics to be calculated that help to measure the suitability of resources to be allocated/recommended to an activity, sub-process or an entire process. This approach can also help to resolve multiple requests simultaneously. Furthermore, the methods proposed considers the actual availability of resources, preventing the recommendation of resources that lack the required availability to resolve the respective cases. The task of allocating resources remains a challenge for many organisations. Nevertheless, the dynamic allocation of resources plays a significant role in optimising the use of resources at the organisational level, reducing costs and improving process performance.

9 The role of the weights

In a multi-criteria problem (Keeney & Raiffa 1993, Marler & Arora 2004), there is no single solution that optimises all the criteria at the same time, and, therefore, a compromise solution must be found. For the sake of simplicity, the method proposed in this paper provides only one solution, which maximises the weighted sum of the metrics considered (Marler & Arora 2010), depending on the weights defined by the person responsible for allocating resources. However, more advanced alternatives are also possible, and compatible with the proposed framework. For example, another person could define other weights, generating different results. The set of solutions that maximise the weighted sum of the metrics for all possible weights is known as Pareto front (Coello et al. 1999). In the Pareto front, no solution is better than the others in all the metrics; on the contrary, a resource allocation that improves one metric will necessarily worsen another one.

To assign the weight that will be given to each metric, it is possible to consider strategies that take into account the desired performance of the process execution or take into account case typologies, using both historical information about the process execution (Kumar et al. 2013) as well as the available literature (e Silva & Costa 2013). For example, a strategy might consider using previously established weights for different scenarios of desired process performance, e.g., when it is desired to ensure the compliance with a service-level agreement in the execution of the process versus when it is desired to reduce the risk on the process execution. This strategy has been used in (e Silva & Costa 2013) to determine the optimal allocation of software developers in various projects that start at the same time, considering different criteria for the assignment of weights (e.g., the required software reliability or the required execution time) and information from the literature on the estimation of required effort (Boehm et al. 2000). Another strategy could consider the assignment of weights according to the characteristics of the cases that are being executed, which is expressed in different resource request characterizations. For example, based on the history of the process execution, it could be established that it is necessary to assign a greater weight to the quality metric when a more complex resource request related to a problem on a server needs to be handled. This strategy has been used in (Kumar et al. 2013) to determine the degree of compatibility between resources that need to be allocated to work groups considering different business contexts.

Implicitly, we are using a compensatory method, i.e., a trade-off is made such that the worst performance in a certain metric is compensated by a better performance in another metric. Optionally, a non-compensatory method (Lee & Anderson 2009) could be used, in which certain metrics have priority over others, so poor performance in the higher priority metrics cannot be compensated by good performance in less relevant metrics. It is possible that a minimum level is required for the the highest priority metrics; or that the highest priority metrics are evaluated first, and only if the same values are obtained in these metrics, the less relevant metrics are considered; or that only the highest priority metrics are considered. Non-compensatory methods also have the advantage that they simplify the decision process by applying heuristics to quickly evaluate existing alternatives with less effort.

10 Conclusions and future work

In this paper, we have presented a framework based on multi-factor criteria that can be used to recommend the most suitable resources for executing different activities, considering both On-demand and Batch requests. There are four main findings presented by this paper. First, the framework introduced allows for the resolution of individual requests (On-demand) or a set of simultaneous requests (Batch). Second, our approach helps to resolve four different use cases that can occur at the moment of allocating resources, therefore providing greater flexibility to the person responsible. He/she is able to decide whether the system must automatically allocate a resource or provide a ranking of resources, so that he/she can make the final decision about what resources to allocate. Third, we implemented two methods that are used in the recommender system to generate the allocations/recommendations. The method based on BPA2 provides a ranking of resources, whereas the method based on ILP seeks to maximize the use of the resources through a direct allocation. Fourth, we implemented our approach in the ProM framework, defining a XES extension to standardize and represent the relevant historical and contextual information. Accordingly, an empirical evaluation was conducted to verify the approach of this work. The results of the experiments demonstrate that our framework is capable of generating allocations/recommendations, simultaneously, for either one or multiple requests defined at run-time. The performance of the methods implemented in the recommender system is efficient. However, the method based on ILP is more scalable than the method based on BPA2, and this is an important benefit to bear in mind in scenarios in which there is a large quantity of resources that can potentially be allocated. Furthermore, we evaluated the framework in a functioning consultancy firm, thereby demonstrating its applicability and adaptability to optimize the task of resource allocation during the execution of a process in a real-world scenario. The findings indicate that the recommender system delivers satisfactory results, according to the judgment of the process owner. For example, by considering contextual and historical information of the process execution, it is sometimes possible to allocate more suitable resources compared to the allocations executed manually by the employees of the consultancy firm.

As future work, we plan to extend the framework to identify the most suitable teams for recommendation to collaboration-intensive processes, as well as to undertake resource allocation in several processes simultaneously. We shall integrate a greedy-based approach as part of the recommender system in order to avoid the problem of exponential growth of the processing time when using BPA2 to solve the *Batch Resource Recommendation* use case; an approach that is only feasible when there are few resources or few tasks in the Batch. Furthermore, it is possible to broaden the knowledge base to improve analysis by incorporating new dimensions in the resource process cube. We shall evaluate the possibility of incorporating the control-flow perspective (van der Aalst et al. 2011), considering the order of activities of the pre-mortem cases and the resources necessary to reach a final state. Finally, we plan to continue to evaluate our approach in a broad spectrum of case studies, and alternative weighting strategies.

Acknowledgement

This project was partially funded by the Ph.D. Scholarship Program of CONICYT Chile (Doctorado Nacional/2014-63140181), and by University of Costa Rica Professor Fellowships. The authors would like to thank the following people for their contributions to this research: Barbara Weber (General Chair), Stefanie Rinderle-Ma and Mathias Weske (Doctoral Consortium Chairs) of the 13th International Conference on Business Process Management (BPM 2015), for their guidance and comments. Reza Akbarinia for the BPA2 implementation support, and Eric Verbeek who gave us valuable support related to the ProM framework.

References

Agrawal, R., Gupta, A. & Sarawagi, S. (1997), Modeling multidimensional databases, *in* 'Proceedings of the Thirteenth International Conference on Data Engineering, April 7-11, 1997 Birmingham U.K.', pp. 232–243.

Akbarinia, R., Pacitti, E. & Valduriez, P. (2011), 'Best position algorithms for efficient top-k query processing', *Inf. Syst.* **36**(6), 973–989.

Arias, M., Rojas, E., Lee, W. L. J., Munoz-Gama, J. & Sepúlveda, M. (2016), Resrec: A multi-criteria tool for resource recommendation, *in* 'Proceedings of the BPM Demo Track 2016 Co-located with the 14th International Conference on Business Process Management (BPM 2016), Rio de Janeiro, Brazil, September 21, 2016.', pp. 17–22.

Arias, M., Rojas, E., Munoz-Gama, J. & Sepúlveda, M. (2015), A framework for recommending resource allocation based on process mining, *in* 'Business Process Management Workshops - BPM 2015', pp. 1–13.

Boehm, B. W., Madachy, R., Steece, B. et al. (2000), *Software cost estimation with Cocomo II with Cdrom*, Prentice Hall PTR.

Braganza, A., Brooks, L., Nepelski, D., Ali, M. & Moro, R. (2017), 'Resource management in big data initiatives: Processes and dynamic capabilities', *Journal of Business Research* **70**, 328–337.

Cabanillas, C., García, J. M., Resinas, M., Ruiz, D., Mendling, J. & Cortés, A. R. (2013), Priority-based human resource allocation in business processes, *in* S. Basu, C. Pautasso, L. Zhang & X. Fu, eds, 'ICSOC 2013', Vol. 8274 of *LNCS*, Springer, pp. 374–388.

Cabanillas, C., Resinas, M., del-Río-Ortega, A. & Cortés, A. R. (2015), 'Specification and automated design-time analysis of the business process human resource perspective', *Inf. Syst.* **52**, 55–82.

Carrera, B. & Jung, J. (2014), Constructing probabilistic process models based on hidden markov models for resource allocation, *in* 'Business Process Management Workshops - BPM 2014 International Workshops, Eindhoven, The Netherlands, September 7-8, 2014, Revised Papers', pp. 477–488.

Coello, C. A. C. et al. (1999), 'A comprehensive survey of evolutionary-based multiobjective optimization techniques', *Knowledge and Information systems* **1**(3), 129–156.

de Leoni, M., Adams, M., van der Aalst, W. M. P. & ter Hofstede, A. H. M. (2012), 'Visual support for work assignment in process-aware information systems: Framework formalisation and implementation', *Decision Support Systems* **54**(1), 345–361.

de Leoni, M. & van der Aalst, W. M. P. (2013), Aligning event logs and process models for multi-perspective conformance checking: An approach based on integer linear programming, *in* 'Business Process Management - 11th International Conference, BPM 2013, Beijing, China, August 26-30, 2013. Proceedings', pp. 113–129.

Dumas, M., Rosa, M. L., Mendling, J. & Reijers, H. A. (2013), *Fundamentals of Business Process Management*, Springer.

e Silva, L. C. & Costa, A. P. C. S. (2013), 'Decision model for allocating human resources in information system projects', *International Journal of Project Management* **31**(1), 100–108.

Fagin, R., Lotem, A. & Naor, M. (2003), 'Optimal aggregation algorithms for middleware', *J. Comput. Syst. Sci.* **66**(4), 614–656.

Havur, G., Cabanillas, C., Mendling, J. & Polleres, A. (2015), Automated resource allocation in business processes with answer set programming, *in* 'Business Process Management Workshops - BPM 2015, 13th International Workshops, Innsbruck, Austria, August 31 - September 3, 2015, Revised Papers', pp. 191–203.

Huang, Z., Lu, X. & Duan, H. (2011), 'Mining association rules to support resource allocation in business process management', *Expert Syst. Appl.* **38**(8), 9483–9490.

Huang, Z., Lu, X. & Duan, H. (2012), 'A task operation model for resource allocation optimization in business process management', *IEEE Transactions on Systems, Man, and Cybernetics, Part A* **42**(5), 1256–1270.

Huang, Z., van der Aalst, W. M. P., Lu, X. & Duan, H. (2011), 'Reinforcement learning based resource allocation in business process management', *Data Knowl. Eng.* **70**(1), 127–145.

Keeney, R. L. & Raiffa, H. (1993), *Decisions with multiple objectives: preferences and value trade-offs*, Cambridge university press.

Kim, A., Obregon, J. & Jung, J. (2013), Constructing decision trees from process logs for performer recommendation, *in* 'Business Process Management Workshops - BPM 2013 International Workshops, Beijing, China, August 26, 2013, Revised Papers', pp. 224–236.

Koschmider, A., Liu, Y. & Schuster, T. (2011), Role assignment in business process models, *in* 'Business Process Management Workshops - BPM 2011', pp. 37–49.

Kuhn, H. W. (1955), 'The hungarian method for the assignment problem', *Naval research logistics quarterly* **2**(1-2), 83–97.

Kumar, A., Dijkman, R. M. & Song, M. (2013), Optimal resource assignment in workflows for maximizing cooperation, *in* 'Business Process Management - 11th International Conference, BPM 2013, Beijing, China, August 26-30, 2013. Proceedings', pp. 235–250.

Kumar, A., van der Aalst, W. M. P. & Verbeek, H. M. W. E. (2002), 'Dynamic work distribution in workflow management systems: How to balance quality and performance', *J. of Management Information Systems* **18**(3), 157–194.

Kumar, A. & Wang, J. (2015), A framework for resource-based workflow management, *in* 'Handbook on Business Process Management 1, Introduction, Methods, and Information Systems, 2nd Ed.', Springer, pp. 507–529.

Lee, L. & Anderson, R. (2009), A comparison of compensatory and non-compensatory decision making strategies in it project portfolio management, *in* 'International Research Workshop on IT Project Management 2009', p. 9.

Liu, T., Cheng, Y. & Ni, Z. (2012), 'Mining event logs to support workflow resource allocation', *Knowl.-Based Syst.* **35**, 320–331.

Liu, X., Chen, J., Ji, Y. & Yu, Y. (2014), Q-learning algorithm for task allocation based on social relation, *in* 'International Workshop on Process-Aware Systems', Springer, pp. 49–58.

Liu, Y., Wang, J. & Sun, J. (2007), A machine learning approach to semi-automating workflow staff assignment, *in* 'Proceedings of the 2007 ACM Symposium on Applied Computing (SAC), Seoul, Korea, March 11-15, 2007', pp. 340–345.

Liu, Y., Wang, J., Yang, Y. & Sun, J. (2008), 'A semi-automatic approach for workflow staff assignment', *Computers in Industry* **59**(5), 463–476.

Ly, L. T., Rinderle, S., Dadam, P. & Reichert, M. (2005), Mining staff assignment rules from event-based data, *in* 'Business Process Management Workshops, BPM 2005', pp. 177–190.

Marler, R. T. & Arora, J. S. (2004), ‘Survey of multi-objective optimization methods for engineering’, *Structural and multidisciplinary optimization* **26**(6), 369–395.

Marler, R. T. & Arora, J. S. (2010), ‘The weighted sum method for multi-objective optimization: new insights’, *Structural and multidisciplinary optimization* **41**(6), 853–862.

McDonald, R. T. (2007), A study of global inference algorithms in multi-document summarization, *in* ‘Advances in Information Retrieval, 29th European Conference on IR Research, ECIR 2007, Rome, Italy, April 2-5, 2007, Proceedings’, pp. 557–564.

Munkres, J. (1957), ‘Algorithms for the assignment and transportation problems’, *Journal of the society for industrial and applied mathematics* **5**(1), 32–38.

Munoz-Gama, J., Carmona, J. & van der Aalst, W. M. P. (2014), ‘Single-entry single-exit decomposed conformance checking’, *Inf. Syst.* **46**, 102–122.

Nakatumba, J. & van der Aalst, W. M. P. (2009), Analyzing resource behavior using process mining, *in* ‘Business Process Management Workshops, BPM 2009 International Workshops, Ulm, Germany, September 7, 2009. Revised Papers’, pp. 69–80.

Oberweis, A. & Schuster, T. (2010), A meta-model based approach to the description of resources and skills., *in* ‘AMCIS’, p. 383.

Obregon, J., Kim, A. & Jung, J. (2013), Dtminer: A tool for decision making based on historical process data, *in* ‘Asia Pacific Business Process Management - First Asia Pacific Conference, AP-BPM 2013, Beijing, China, August 29-30, 2013. Selected Papers’, pp. 81–91.

Rinderle-Ma, S. & van der Aalst, W. M. P. (2007), ‘Life-cycle support for staff assignment rules in process-aware information systems’, *BETA Working Paper Series, WP 213, Eindhoven University of Technology (2007)* .

Rozinat, A. & van der Aalst, W. M. P. (2005), *Conformance Testing: Measuring the Alignment Between Event Logs and Process Models*, BETA Research School for Operations Management and Logistics.

Russell, N., van der Aalst, W. M. P., ter Hofstede, A. H. M. & Edmond, D. (2005), Workflow Resource Patterns: Identification, Representation and Tool Support, *in* O. Pastor & J. F. e Cunha, eds, ‘CAiSE 2005’, Vol. 3520 of *LNCS*, Springer, pp. 216–232.

Schrijver, A. (1998), *Theory of linear and integer programming*, John Wiley & Sons.

Senderovich, A., Weidlich, M., Gal, A. & Mandelbaum, A. (2014), Mining resource scheduling protocols, *in* ‘Business Process Management - 12th International Conference, BPM 2014, Haifa, Israel, September 7-11, 2014. Proceedings’, pp. 200–216.

van der Aalst, W. M. P. (2009), ‘Process-aware information systems: Lessons to be learned from process mining’, *T. Petri Nets and Other Models of Concurrency* **2**, 1–26.

van der Aalst, W. M. P. (2013), ‘Decomposing petri nets for process mining: A generic approach’, *Distributed and Parallel Databases* **31**(4), 471–507.

van der Aalst, W. M. P. (2016), *Process Mining - Data Science in Action, Second Edition*, Springer.

van der Aalst, W. M. P., Adriansyah, A., de Medeiros, A. K. A., Arcieri, F., Baier, T., Blickle, T., Bose, R. P. J. C., van den Brand, P., Brandtjen, R., Buijs, J. C. A. M. et al. (2011), Process mining manifesto, *in* 'Business Process Management Workshops - BPM 2011', Springer, pp. 169–194.

van der Aalst, W. M. P. & Verbeek, H. M. W. (2014), 'Process discovery and conformance checking using passages', *Fundam. Inform.* **131**(1), 103–138.

Vanderbei, R. J. (2001), *Linear Programming: Foundations and Extensions, Second Edition*, Kluwer Academic Publishers, Boston, MA.

Vanderfeesten, I. T. P. & Grefen, P. (2015), Advanced dynamic role resolution in business processes, *in* 'Advanced Information Systems Engineering Workshops - CAiSE 2015 International Workshops, Stockholm, Sweden, June 8-9, 2015, Proceedings', pp. 87–93.

Verbeek, H. M. W., Buijs, J. C. A. M., van Dongen, B. F. & van der Aalst, W. M. P. (2010), Xes, xesame, and prom 6, *in* 'Information Systems Evolution - CAiSE Forum 2010, Hammamet, Tunisia, June 7-9, 2010, Selected Extended Papers', pp. 60–75.

Wibisono, A., Nisafani, A. S., Bae, H. & Park, Y. (2015), On-the-fly performance-aware human resource allocation in the business process management systems environment using naïve bayes, *in* 'Asia Pacific Business Process Management - Third Asia Pacific Conference, AP-BPM 2015, Busan, South Korea, June 24-26, 2015, Proceedings', pp. 70–80.

Xu, J., Liu, C. & Zhao, X. (2008), Resource allocation vs. business process improvement: How they impact on each other, *in* 'Business Process Management, 6th International Conference, BPM 2008, Milan, Italy, September 2-4, 2008. Proceedings', pp. 228–243.

Xu, L., Hutter, F., Hoos, H. H. & Leyton-Brown, K. (2011), 'Satzilla: Portfolio-based algorithm selection for SAT', *CoRR* **abs/1111.2249**.

Zhao, W. & Zhao, X. (2014), 'Process mining from the organizational perspective', *Advances in Intelligent Systems and Computing* pp. 701–708.