Computer Science and Information Systems 20(4):1661–1685 https://doi.org/10.2298/CSIS230401057V

Systematic Exploitation of Parallel Task Execution in Business Processes*

Konstantinos Varvoutas, Georgia Kougka, and Anastasios Gounaris

Department Of Informatics, Aristotle University of Thessaloniki
Thessaloniki, Greece
{kmvarvou,georkoug,gounaria}@csd.auth.gr

Abstract. Business process re-engineering (or optimization) has been attracting a lot of interest, and it is considered as a core element of business process management (BPM). One of its most effective mechanisms is task re-sequencing with a view to decreasing process duration and costs, whereas duration (aka cycle time) can be reduced using task parallelism as well. In this work, we propose a novel combination of these two mechanisms, which is resource allocation-aware. Starting from a solution where a given resource allocation in business processes can drive optimizations in an underlying BPMN diagram, our proposal considers resource allocation and model modifications in a combined manner, where an initially suboptimal resource allocation can lead to better overall process executions. More specifically, the main contribution is twofold: (i) to present a proposal that leverages a variant of representation of processes as Refined Process Structure Trees (RPSTs) with a view to enabling novel resource allocation-driven task re-ordering and parallelisation in a principled manner, and (ii) to introduce a resource allocation paradigm that assigns tasks to resources taking into account the re-sequencing opportunities that can arise. The results show that we can yield improvements in a very high proportion of our experimental cases, while these improvements can reach a 45% decrease in cycle time.

Keywords: business process optimization, process models, resequencing, parallelism, resource allocation

1. Introduction

Business Processes (BPs) have nowadays become quite complex as the business requirements are increasing, e.g. to accommodate multiple and evolving customer needs. This situation renders the significance of Business Process Management (BPM) even higher. BP optimization, also covered by terms such as BP reengineering and redesign, has persisted as a key aspect in BPM since the emergence of BPM as a scientific area.

In general, automated solutions for BP optimization have not been explored as deeply as process modelling, as discussed in several places, e.g., [19],[27],[4]. In such a context, this work is motivated by the more specific observation that, up to date, there is no automated optimization technique for BPs that can benefit from the overlapping task execution in order to improve latency (a.k.a. cycle time or duration) and is generally applicable. Moreover, concurrent task execution is typically addressed separately from task

* An early version of this work has appeared in [26].

resequencing, whereas the latter tends to focus solely on cases where there are tasks that may lead to immediate process termination [11],[4]. In addition, task allocation is also considered independently of task parallelisation and re-sequencing, both of which modify the structure of the business model whereas task allocation does not impact the model structure.

We aim to address the afore-mentioned limitations and more specifically, we target scenarios where a BP is modelled with the help of a procedural approach, such as BPMN[1], and the optimizations on which we focus fall under the BP behavior heuristics according to the taxonomy in [4]. This category of heuristics includes activity resequencing and parallelism, and the impact of their application is reflected on the model diagram. In other words, the optimized model's structure is different than the initial one. The latter is modified so that certain objectives, such as cycle time or total cost, are improved and our main novelty is that we apply these heuristics, not only in a resource-aware manner, but through leveraging resource allocation so that opportunities to enhance the model structure arise. To date, resequencing has been explored in a manner that it is tightly coupled with the existence of knock-out activities either directly or indirectly [1],[11],[25]; knock-out activities are the activities that can lead to immediate process termination, such as automatically rejecting an application if it does not meet certain criteria. Here, we depart from such a narrow consideration of resequencing. In addition, principled parallelism, where different activities overlap in the time domain and are executed concurrently, is an overlooked area in BP in the sense that although it is a well-recognized heuristic, to date, no algorithmic technique has been proposed to leverage it.

In our previous work[26], we have introduced a novel combination of re-sequencing and parallelism enforcement, with the aim of reducing the cycle time of the process in question. To this end, we leverage the task-based variant [6] of representation of processes as Refined Process Structure Trees (RPST) [24]. This representation allows us to check valid resequencing actions systematically, while it is more amenable to cycle time computations. A key aspect in our solution is that we annotate the tree vertices with the resource allocated, i.e., we take into account both the control flow and the resource perspective of the process. This part of our solution has appeared in [26].

The main novelty in this more complete proposal is that we extend the initial conference version in [26] by combining the cost-based task resequencing and parallelism with a reorderability-aware resource allocation mechanism that may take suboptimal task assignments decisions in order to create room for resequencing. Our rationale is not tightly coupled with a specific resource allocation technique, provided that such a technique assigns a suitability score for each resource-task pair. In our proof-of-concept implementation, the Realistic, Investigative, Artistic, Social, Enterprising and Conventional (RIASEC) dimensions are specified to quantify the suitability of the resources to execute specific tasks in line with the proposal [14].

Reorderability potential has already been discussed in [22]. In this work, we depart from simply assessing the reorderability based on model complexity measures and we do perform process redesign. However, another major difference between our work and the work in [22] is that we take into account resource allocation rather than model features in order to reason about the applicability of reordering tasks and putting branches in parallel.

[1] https://www.bpmn.org/

The results of this work are particularly encouraging. We show that we can improve a very high proportion of cases with the maximum improvements in cycle time being 45%. Moreover, we explain that our proposals can be combined so that the best performing flavor in each case can be selected.

The remainder of this work is structured as follows. Next, we present an exemplary use case. In Section 3, we provide the background regarding RPST and its task-based variant along with cost modeling. In Section 4, present a cost-based task-ordering algorithm that is extended in Section 5 by the proposal of an innovative resource allocation algorithm that is reorderability-aware. We continue with the experimental evaluation results of the discussed techniques that shows their benefits in Section 6. Finally, in Sections 7 and 8 the related work and conclusions are discussed, respectively.

2. Our Case Study

Our case study is shown in Figure 1 and refers to a common real-world BP regarding an employee expense reimbursement request[2]. Briefly, the BP consists of 8 activities that are required to analyse, approve and pay an expense statement submitted by an employee of a business, while accounting for essential steps, such as money transfer, notifications and validation that an account exists. Despite its simplicity, this process is amenable to optimizations, where the relative order of some parts can change, e.g., the initial check regarding the account existence can be performed in parallel with other activities. Furthermore, the activities are performed by different actors (automated services, ordinary employees and supervisors), which may result in configurable execution ordering and therefore, lower waiting times at the expense of higher human effort cost. As such, this type of business process forms an excellent candidate to benefit from the advances in automated cost-based flow optimization that we aim to introduce.

More specifically, we handle the example case study as follows. In our approach, we start with the modelling quality and we consider only well-structured models. It is out of our scope to advocate specific automated transformations, but there exist several proposals in the literature, e.g. [17]. Therefore, the model we process is transformed to the well-structured form as shown in Figure 2.

Next, a closer examination of the activities reveals that the review and approval of claims above $200 can be seen as a knock-out activity because one of its outcome can lead to immediate process termination under an additional assumption that the task of advising employees of the rejection has zero cost and can be replaced by a message. Therefore, it makes sense to move the knock-out activity as early as possible using a rank formula that considers both activity cycle time and cost. This is already covered by previous works, e.g., [1],[11]. Our approach can encapsulate these proposals, but, to better show the novelty of this work, in our case study, we treat every activity, including this specific one, as *not* being a knock-out one, i.e., as if all claims are approved. So, the question that arises is: *"If there are no knock-out activities, what type of resequencing is beneficial?"*.

Our answer to this question is to move the block with the review and approval of claims above $200, which is performed by the supervisor, in parallel with the early blocks

[2] `https://www.businessprocessincubator.com/`

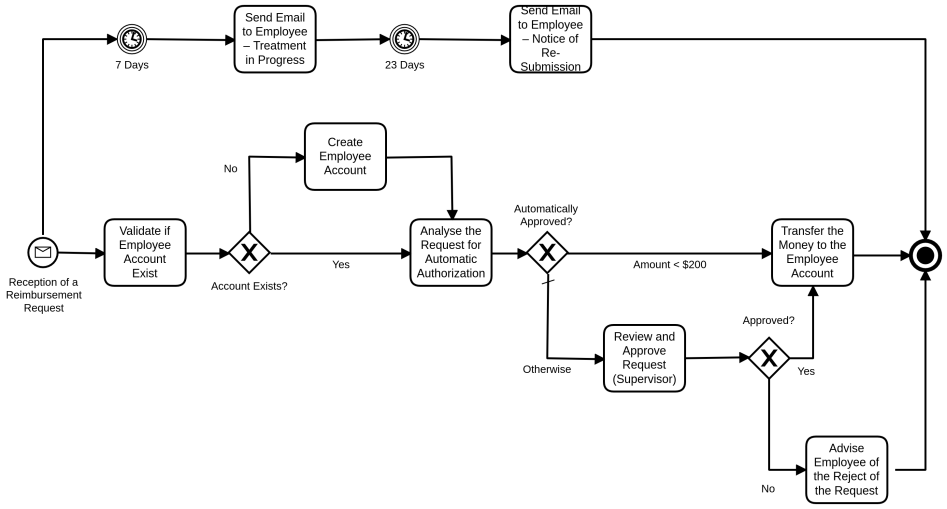

Fig. 1. The process model of an Employee Expense Reimbursement Request

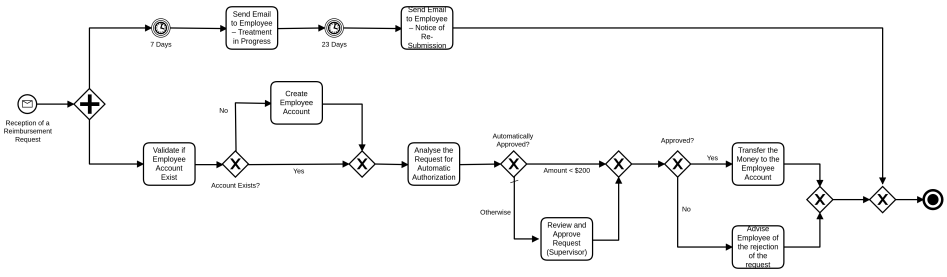

Fig. 2. The well-structured process model of an Employee Expense Reimbursement Request

Table 1. Case Study Activity Cycle Times and Costs

ID	Path	Activity Name	Cost	Time
1	1	Send Email to Employee - Treatment in Progress	1	1 Minute
2	1	Send Email to Employee - Notice of Re-Submission	1	1 Minute
3	2	Validate if Employee Account Exists	2	1 Day
4	2	Create Employee Account	4	1 Day
5	2	Analyze the Request for Automatic Authorization	3	1 Hour
6	2	Review and Approve Request (Supervisor)	8	1 Day
7	2	Transfer the Money to the Employee Account	2	2 Days
8	2	Advise Employee of the rejection of the request	0.1	1 Hour

Table 2. Case Study Path Probabilities

Name of Gate	Path	Probability
XOR_block1	*Path_Account*	0.8
XOR_block1	*Path_No_Account*	0.2
XOR_block2	*Path_Amount*	0.8
XOR_block2	*Path_Otherwise*	0.2
XOR_block3	*Path_Transfer*	0.6
XOR_block3	*Path_Advise*	0.4

of account existence validation and possible creation of an employee account. This allows two distinct types of resources, namely both the supervisor and the supervisee employee, to operate in parallel, so that their cycle times are overlapped. However, it is valid to claim that such resequencing modifications cannot happen because the activity performed by the supervisor should follow the activity for the analysis of the request for the automatic authorization. Therefore, the latter task needs to be moved earlier as well. In our solution, we deal with these issues and in a nutshell, we propose a principled technique that puts blocks of activities in parallel. This movement of activities in the diagram leads to lower cycle times, and entails the incorporation of AND gateways in the model, while ensuring that precedence constraints are met through also moving the necessary activities upstream. I.e., the validity of the optimized model is always guaranteed.

2.1. Statistical Metadata

The solution that we propose is principled in two senses: (i) we follow a cost-based approach, according to which the alternative models are quantitatively annotated in terms of their cycle time and cost; and (ii) we cast our solution as an algorithm that can be easily followed (and re-implemented) by third parties in arbitrary scenarios.

To support the first point above, it is necessary to obtain statistical metadata for the activities that are present in the model. If we are interested in cycle time and cost, there are at least three types of statistical metadata required, namely (a) the activity cycle times; (b) the activity costs and (c) the probability to follow a specific path after (X)OR gateways. These are adequate to compute the process cycle time, as is recorded in several textbooks, e.g., [4]. Tables 1 and 2 present such example metadata for our case study.

3. Process Model Decomposition

We employ a convenient representation of a process, where convenience means that the representation should naturally lend itself to resequencing operations, and process total cycle time and cost can be easily computed. The cost (resp. cycle time) of the entire process is calculated by combining the costs (resp. cycle times) of the individual fragments using appropriate cost functions, e.g., sums, minimum, maximum and so on. We advocate the usage of the Refined Process Structure Tree (RPST) [24] and more specifically a specific variant of RPST, called Task Based Process Structure Tree (TPST) [6]. Both can be deemed as decomposition techniques separating a business process into its individual fragments exactly as we desire.

3.1. Task Based Process Structure Tree (TPST)

The construction of TPST entails a decomposition approach that is based on RPST [6], where a business process model is separated into blocks, which are organised in a hierarchical way. The main differences between TPST and RPST are the following:

- The leaf nodes of a TPST represent a node (i.e., a BPMN activity) of its corresponding process model instead of an edge. This allows us to compute total times and costs based on the activity times and costs, respectively, and also to reorder (blocks of) activities.
- There are multiple types of process fragments into which a process may be separated instead of one generic type. These types include $Sequence$, $Loop$, XOR and AND. These are the same types that are typically employed in flow analysis-based cost computation.
- The leaf nodes of a TPST are ordered, thus making the TPST a semi-ordered tree.
- The internal (i.e., non-leaf) nodes of a TPST represent the control flow.

A subtle point regarding the model in Figure 2 is that, although it is well-structured, an AND split gateway is paired with an XOR merge gateway. Normally, such a situation may lead to erroneous lack of synchronization, but in our case, only a single token is guaranteed to arrive at the merge XOR gateway, as required in valid BPMN models. However, we need to employ a specific *AND/XOR* fragment type to cover this case. In the appendix in [26], we present an alternative modelling of the same process, which employs boundary events in a sub-process in order to show that our approach is not specific model-dependent as long as the model to be optimized is in a well-structured form.

3.2. Decomposition of Our Scenario

The TPST of the model in Figure 2 is depicted in Figure 3. All the leaf nodes of the TPST correspond to BP activities and the waiting events. The root of the TPST represents the complete BP of Figure 2 and it is a *Sequence*. The children of this sequence are (i) a starting event, (ii) an *AND/XOR* gateway node and (iii) an end event. At the next level, there are two other sequences that are children of the *AND/XOR* gateway node. The left sequence represents the top path of the BPMN model, while the right sequence represents

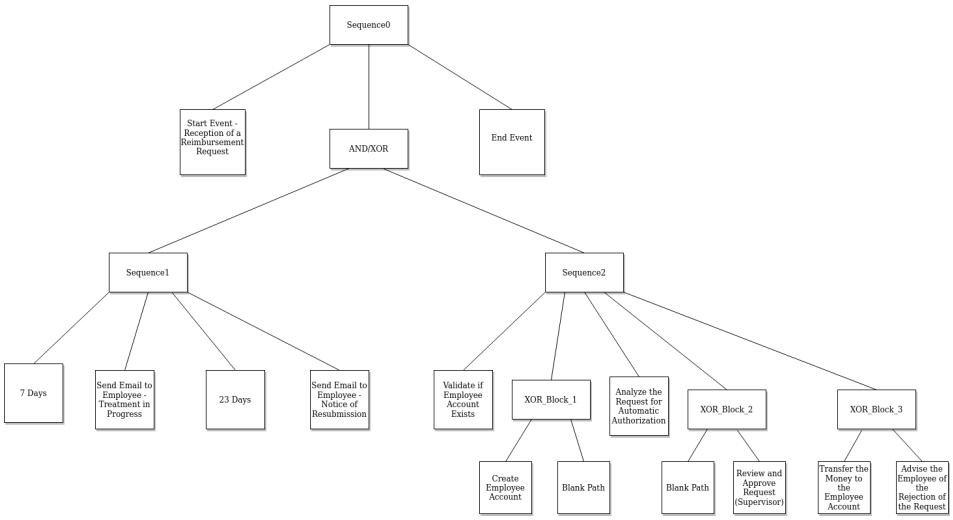

Fig. 3. The TPST of our case study BP model.

the bottom path. Similarly, each of these sequences is connected with its children activities and/or *XOR* blocks comprising activities.

After the decomposition of the BP model to its fragments using the TPST approach, the total cost and cycle time of the process can be calculated in a straightforward manner. When a token arrives at the *AND/XOR* gateway, both upper and bottom paths are initiated. In the upper path, due to the timer activities, the token proceeds when each timer runs out. The two parallel paths are executed independently and the cycle time of the *AND/XOR* subprocess block is the *minimum* of the two paths, while the cost is the cost of the path with the minimum cycle time plus the costs of all TPST nodes that have completed in the other path.

Based on the metadata in Tables 1 and 2, assuming that a day is equal to 8 hours, we can easily compute that the cycle time of the first sequence in Figure 3 is 30 days and 2 minutes, while, for the second sequence it is 2 days and 6.2 hours. The total cycle time is the minimum of these two values. Details are provided in [26]. Using classical formulas in textbooks, such as [4], can help as to compute the cycle time and cost directly from the BPMN diagram, but the TPST representation renders this computation trivial.

However, this cost model is limited to reflect the cost of the execution paths without considering the available resources and the advantage of executing in parallel independent paths. For example, in our case study, if there were more reimbursement requests than request handlers, there would be resource contention that leads to increases in the cycle time. Under such conditions, some instances and their corresponding tasks are put on hold until the required resources become available. Also, the statistics in Table 1 reflect expected times, which lead to a situation that the cycle time computations for all instances will always consider that *Sequence2* is the fastest one for *all* instances. Obviously, it would be more realistic to consider time variations; however these improvements in the cost modelling do not affect our solution.

Table 3. Behavioral constraints considered

Activity constraints	Description
precedence(a_1, a_2)	Whenever activity a_2 is executed, the execution of a_1 must precede
not co-existence(a_1, a_2)	Either activity a_1 or a_2 can be executed, but not both
chain succession(a_1, a_2)	Activity a_2 directly follows a_1

Essentially, our resequencing technique is independent of any model to compute the process cycle time and cost. During resequencing, several alternative models are implicitly generated and checked as to whether they lead to cycle time improvements. Instead of using the technique described above for cycle time computation, advanced simulators, e.g., BIMP[3] or digital twins [3], can be also employed.

4. TPST-based Task Re-ordering with Fixed Resource Allocation

Our methodology accepts as input the TPST representation of the BPMN model and aims to produce an optimized model. To this end, it takes into account (i) the metadata of the input model's tasks, (ii) any behavioral constraints that may apply; and (iii) the given resource allocation. The former item has already been introduced (see also Tables 1 and 2), thus, in this section we focus on the constraints and the resource allocation.

4.1. Notation and Cycle Time Computation Considerations

The main notation required to explain our optimization approach is summarized below.

- $A = \{a_1, \ldots, a_n\}$ defines a set of n activities that appear in the BPMN business process model. The cycle time (resp. cost) of each activity $a_i,\ i = 1n$ is denoted as $ct(a_i)$ (resp. $cost(a_i)$)
- $R = \{r_1, \ldots, r_m\}$ denotes a set of m resources, where the set of activities A are allocated to.
- $A \rightarrow R$ defines the mapping of activities to resources; a_i^j denotes that a_i is mapped to r_j, where $i \in 1 \ldots n$ and $j \in 1 \ldots m$.

In addition, we consider a subset of the constraints defined by DECLARE [15], as depicted in Table 3. Without loss of generality, we assume that the *precedence* constraint subsumes the *chain succession* one, and their existence prohibits the existence of the *not co-existence* constraint. These constraints are used in many works, e.g., [2], and can be accompanied by additional ones, such as existence of alternating ordering, which, however, add no further knowledge in our case and thus are not required by our technique.

The main implication of the resource allocation regarding the computation of the process cycle time is when considering AND blocks: instead of returning the maximum of all branches always, we do so only if the resources are different. Activities belonging to

[3] `https://bimp.cs.ut.ee/simulator/`

Algorithm 1 TPST-based re-ordering

```
1: Annotate TPST with resource allocation info
2: Perform BFS on TPST
3: if node.type is "Sequence" then
4:     for each node_pair in the sequence do
5:         check pair validity for parallel execution
6:         assess impact on cycle time
7:         resolve additional constraint violations
8:     end for
9: end if
```

different branches but executed by the same resource are treated as executing sequentially in terms of their cycle time. Note that the resource-oriented notation is kept as simple as possible at this stage and in the next section, we will include extensions and provide full details.

4.2. Cost-based Task-reordering Algorithm

We annotate each leaf node of the TPST with the resource allocated. Each parent node with all its children annotated with the same resource is annotated accordingly as well; if there are multiple different resource annotations among the node children, the resource annotation of the parent is the union of all children resources. The resequencing algorithm is applied to such a resource-annotated TPST representation of the input model. The next step of the algorithm is to traverse the TPST through performing Breadth-First-Search (BFS). For each *Sequence* node encountered, every possible node pair in the specific sequence is considered as a candidate for parallel execution through the following procedure (see also Algorithm 1).

1. First, it is checked whether any precedence constraints are violated. I.e., for any two activities a_1 and a_2 that are considered to be placed in a parallel (AND) block, behavioral constraints must not include both *Precedence(*a_1, a_2*)* and *Precedence(*a_2, a_1*)*. If a node is not an activity one but a complete fragment, e.g., a XOR block, then this check is performed for all activities in the block. In essence, the absence of these two constraints suggests the existence of the relation of potential parallelism [4].
2. The impact on the cycle time and execution cost is examined when moving the downstream node in parallel with the upstream one in the sequence. If the downstream node has already been moved in parallel with another node in an earlier pair consideration, it is checked whether the upstream node in the initial sequence should be added to the branch of the AND block that does not contain the downstream node. Basically, the cycle time is improved if the nodes considered for parallel execution are assigned to different resources, i.e., the intersection of their resource annotations is null. The total cost remains unaffected unless a knock-out activity is executed in parallel instead of as early as possible. If there is no improvement, this pair is not further considered.
3. A final check whether creating an AND block leads to violation of precedence constraints involving one activity other than a_1 or a_2 is performed. If this the case, we need to consider if the violation can be resolved by reordering the other activity just

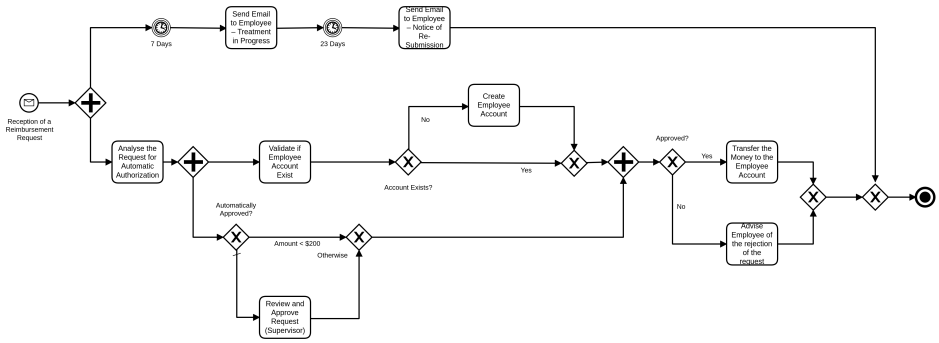

Fig. 4. The result of applying our proposed methodology to the case study BPMN.

before the AND block. If such a reordering is not feasible due to the constraints of the model, then the changes under consideration are rejected and we continue with the next pair.

Complexity Analysis: The complexity of the above solution in $O(n^3)$, where n is the total number of activities. The maximum number of pairs is $O(n^2)$ while each node can participate in $O(n)$ behavioral constraints. Actually, the complexity is lower, since it is cubic in the length of the longest sequence. Apart from the polynomial complexity, it is important to stress that even in large processes, n does not typically grow very large. Finally, due to the same reasoning as in [10],[9], we can characterize the problem in question as NP-hard, thus the polynomial algorithm presented does not aim to find the optimal solution but just to improve the BPMN diagram considered.

4.3. Application in Our Example Case Study

As a proof of concept, we present the application of our proposed methodology to the example case study that was presented in Section 2[4]. The algorithm accepts as input the TPST representation of the input model as shown in Figure 3. The resource annotation (not shown) is based on the activity names: all activities but one are executed by a single resource, while the remaining activity is executed by a different resource named *supervisor*. The constraints are presented in Table 4. The first sequence *Sequence1* has no valid candidate node pairs due to the behavioral constraints in place. However, when applying the algorithm on *Sequence2*, the result is that activity *Validate if Employee Account Exists* and block *XOR_Block_2* are reordered for parallel execution, and then *XOR_Block_1* is placed in the same branch as the former activity[5]. The resulting BPMN is presented in Figure 4.

[4] The prototype implementation can be found at `https://github.com/kmvarvou/bpmn_tpst_optimization`

[5] According to the example metadata, actually this last movement does not lead to improvements (the cycle time remains the same) but we include it for completeness, since it may yields lower times if the cycle time of the supervisor task was longer.

Table 4. Case Study Behavioral Constraints

Constraint	Activity1	Activity2
Precedence	*7 Days*	*Send Email to Employee - Treatment In Progress*
Precedence	*7 Days*	*23 Days*
Precedence	*7 Days*	*Send Email to Employee - Notice of Resubmission*
Precedence	*Send Email to Employee - Treatment In Progress*	*23 Days*
Precedence	*Send Email to Employee - Treatment In Progress*	*Send Email to Employee - Notice of Resubmission*
Precedence	*23 Days*	*Send Email to Employee - Notice of Resubmission*
Precedence	*Review and Approve Request (Supervisor)*	*Transfer the Money to the Employee Account*
Precedence	*Review and Approve Request (Supervisor)*	*Advise the Employee of the Rejection of the Request*
Precedence	*Validate if Employee Account Exists*	*Create Employee Account*
Precedence	*Create Employee Account*	*Transfer the Money to the Employee Account*
Precedence	*Create Employee Account*	*Advise the Employee of the Rejection of the Request*
Precedence	*Validate if Employee Account Exists*	*Transfer the Money to the Employee Account*
Precedence	*Validate if Employee Account Exists*	*Advise Employee of the rejection of the request*
Precedence	*Analyze the Request for Automatic Authorization*	*Review and Approve Request (Supervisor)*

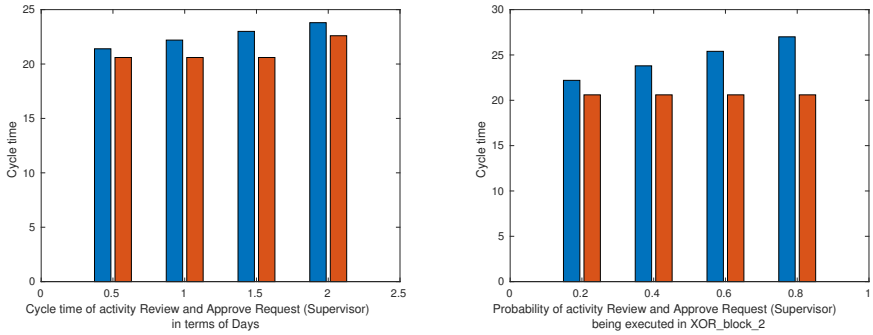

Fig. 5. Process cycle times for different values in terms of cycle time of the *Review and Approve Request (Supervisor)* activity (left) and the probability of the activity being executed in *XOR_block_2* (right). The blue bars refer to the original model and the orange bars to the optimized one.

In the optimized model, the average total cycle time becomes 2 days and 4.6 hours. In other words, the introduction of parallelism leads to a decrease of 7.3 % in terms of cycle time. In Figure 5, we present the results of applying our methodology for a variety of values in terms of cycle time (of the activity *Review and Approve Request (Supervisor)*) and XOR branch probability; all the other values remain the same as the example ones already provided in Section 2. In the two plots in the figure, the improvements are up to 13.5% and 23.8%, respectively.

5. Blending Resource Allocation with Reordering

In the previous section, our approach to task reordering relied on the existence of a resource allocation for a business process model. In this section, we extend our approach to conduct also resource allocation. The components of task reordering and resource alloca-

tion are loosely coupled and the exact resource allocation proposal can be substituted by more sophisticated ones, since the reorderability-aware resource allocation rationale that we propose is easy to be incorporated to additional resource allocation techniques.

5.1. Rationale and Further Notation

We begin with extending the resource modeling specifications, presented in Section 4.1, upon which we build our work. A lot of work has focused on the importance of capturing the differentiated performance of distinct resources when trying to build an accurate process simulation [13],[12]. In this context, we assume a scenario where, to drive the final mapping, there is also a quantified suitability measure when assigning an activity to a resource. More specifically, activities and resources both feature a psychological profile that aims to classify the resource performance per task according to a set of personality traits [14]. These attributes correspond to the *Realistic, Investigative, Artistic, Social, Enterprising and Conventional (RIASEC)* dimensions of resources and tasks. The psychological profile of an activity a_i is defined as a sextuple, one attribute for each RIASEC dimension, denoted as:

$$P_{a_i} =< P_{a_i.1}, P_{a_i.2}, \ldots, P_{a_i.6} >$$

A similar profile characterizes the suitability of each resource as well. Therefore, while a set of resources may be eligible for the execution of a specific activity, each one of them may demonstrate different levels of suitability, depending on its profile, denoted as:

$$P_{r_j} =< P_{r_j.1}, P_{r_j.2}, \ldots, P_{r_j.6} >$$

Furthermore, the cost and duration of the activity execution are differentiated, depending on the allocated resource. In other words, instead of a *single cycle time* for each activity for all resources, $ct(a_i)$, the *cycle time* is dependent on the resource r_j executing a_i, and is denoted as $ct(a_i^j)$. Similarly, for each activity, we assume the existence, apart of the average duration (cycle time), of the cost of execution, $cost(a_i)$, which may be provided by domain experts or through past execution logs. Both these resource allocation-agnostic metrics, namely $ct(a_i)$ and $cost(a_i)$, are indicative of the performance of each activity, without taking into account the resource allocated. When allocating a specific resource for the execution of an activity, these metrics are differentiated according to how suitable this resource is. To this end, the suitability of a resource for a specific activity is quantified using the term *performance coefficient*, denoted as pc.

More specifically, for each activity $a_i \in A$ and for each resource $r_j \in R$ that feature performance psychological profiles P_{a_i} and P_{r_j} respectively, and assuming r_j being a resource that is eligible to be selected for the execution of activity a_i, we calculate the performance coefficient pc_i^j using the following equation:

$$pc_i^j = 1 + \sum_{k=1}^{6} |P_{a_i.k} - P_{r_j.k}|$$

Then, we can derive the resource-aware cycle time and cost metadata as follows: $ct(a_i^j) = ct(a_i) \times pc_{i_j}$, while $cost(a_i^j) = cost(a_i) \times pc_{i_j}$

Additionally, as implied above, we take into account the *eligibility e* of the available resources to execute the corresponding activities. The eligibility is defined as follows:

$$e : A \times R \rightarrow \{0, 1\}$$

As such, the eligibility of each resource r_j with regards to activity a_i is denoted as e_i^j and is a binary variable.

Another factor that affects the resource allocation decisions is the *availability*, denoted as $av(r_j)$. This metric reflects the capacity constraint in the terms of defining the number of activities that can be allocated to a single resource $r_j \in R$.

Finally, a key metric for computing total cycle time closer to reality is the *overhead of intercommunicating* between two different resources in terms of transition cost when activities executed by different resources are connected in the BPMN diagram (ric, standing for resource intercommunication cost):

$$ric(r_i, r_j) : R \times R \rightarrow \mathbb{R}_{\geq 0}$$

5.2. Baseline Allocation

In summary, the allocation approach that is introduced in the following requires as input:

- a TPST structure that consists of a set of activities (A) along with their profiles ($P_{a_i},\ \forall a_i \in A$),
- a set of resources (R) along with their profiles ($P_{r_j},\ \forall r_j \in R$),
- an eligibility mapping of activities to resources (e),
- the baseline cycle time $ct(a_i)$ and cost $cost(a_i)$ of the activity a_i execution, based on which the $ct(a_i^j)$ and $cost(a_i^j)$ metadata can be derived, and
- the capacity of each resource r_j

Based on the key metadata input defined above, the baseline allocation technique is briefly presented as follows.

The first step of this optimization approach is to identify and rank all the activity nodes of the TPST input in terms of their average duration of execution $ct(a_i)$. Then, according to their ranking, each activity is allocated to the most suitable available resource. More specifically, the resource selection depends on the lowest performance coefficient and the resource availability based on the existing resource's capacity constraint. In the case that there is an activity, for which no allocation can be found, this activity is re-ranked and pushed to the top of the ranking. This process is repeated until a valid allocation is found for all activities.

The steps of the technique are presented in Algorithm 2. Basically, after ranking the activities, we choose for an activity a_i the resource r_j with the minimum pc_i^j value, provided that $e(a_i^j) = 1$ and $av(r_j) \geq 1$. After each allocation, the $av(r_j)$ metric is decreased by one. In practice, the re-ranking allows activities that can be executed only by scarce resources to be considered earlier, so that such resources have not reached their capacity.

Algorithm 2 Baseline Resource Allocation Approach

```
1: Rank activity nodes based on average duration of execution
2: for each activity node a_i do
3:    for each resource r_j that is eligible and available for its execution do
4:       calculate performance coefficient pc_i^j
5:       allocate resource with lowest performance coefficient
6:       if there are is no possible allocation for this activity a_i then then
7:          push a_i to the top of the ranking
8:          go to line 2
9:       end if
10:   end for
11: end for
```

5.3. Reorderability-aware Resource Allocation

In this section, we propose a resource allocation approach, which aims to produce an allocation that promotes/enables the parallelization of activities. To this end, our approach entails a pre-processing step where activities, which may be executed parallel, are identified. This is achieved, by taking into account the resource mapping and the behavioral constraints of the input process model.

More specifically, our approach takes as input a process model represented in TPST form, along with the set of execution-related metadata that was outlined in Section 5.1. The input TPST is traversed using DFS in order to identify sequence nodes/blocks. Then, for each sequence block identified, every node pairing is checked on whether it is eligible for parallel execution.

The check entails the following step. The behavioral constraints that apply to the input model are checked, to identify which nodes may be executed in parallel. As already mentioned in Section 4, for each pair of activities a_1 and a_2, the behavioral constraints must not include both *precedence(*a_1, a_2*)* and *precedence(*a_2, a_1*)*. The absence of these two constraints suggests the existence of the relation of potential parallelism [4]. A subsequent step is to ensure that there is at least one possible assignment of resources to tasks so that the two activities are executed by different resources. We also note that in the generic case, a node in the tree is not a leaf, thus it covers a set of activities.

Through these steps, activity pairings which may be placed in parallel are identified and are referred to as *prioritized* activities. Then these node pairings are pushed to the top of the ranking, so that an allocation for them is prioritized. Our approach aims to provide a resource allocation which, by taking into account the prioritized pairings, performs such a resource allocation that enables the parallelization of the aforementioend nodes, even though the initial allocation may be deemed as a suboptimal one.

Our approach comes in two different flavors, with regard to how the allocation is produced for these nodes. In the first flavor, the node pairings are directly assigned to resources, in a way that ensures that nodes of a single pairing are executed by different resources. In the second flavor, the allocation of different resources to each pairing is produced indirectly.

More specifically, in the first flavor, the *prioritized* activities are assigned resources in a direct manner in a way that allows for their placement in parallel. For each activity pair included in the *prioritized* set, their resource mappings are scanned to identify all possi-

ble allocations. The first allocation that satisfies the condition for parallel placement of the two activities, i.e. the intersection of the allocations of the two is null, is chosen. The remaining, non-*prioritized* activities are allocated using the baseline approach presented in Section 5.1. This flavor is termed as *Advanced_Enforced*, since it enforces parallelization.

In the second flavor, termed as *Advanced_Promoted*, the allocation of resources facilitates the placement of $eligible$ activities in parallel in an indirect and configurable manner. For each activity pair $a1, a2$ included in the *prioritized* set, each resource that is present in the resource mapping of only one of the pair's activities has its performance coefficient (for that respective activity) reduced by a factor of p. Then, activities are allocated to resources using the approach presented in Section 5.1. [6]

5.4. Discussion

As implied by the description of the advanced flavors above, their coupling with the baseline resource allocation techniques is a loose one. In other words, the baseline technique, which is a list scheduling one, is replaceable and the reorderability-aware rationale is more generically applicable. The prerequisite is that any base resource allocation technique to be capable of assigning suitability values for each (eligible) resource-activity pair. Overall, the technique is loosely coupled with both the cost model and the base resource allocation technique employed.

Also, this work aims to pave the way for a new line of research work that revisits cost-based activity reordering and the parallelism redesign heuristic in BPMN diagrams. Apart from the aspects discussed above, we focus also on two additional one.

Firstly, the proposed technique, similarly to any cost-based technique, relies on accurate quantitative and qualitative metadata. We expect that the base activity cycle times, possibly allowing for uncertainty, can be provided by process mining techniques applied on previous logs or domain experts or both. This also applies to constraint derivation, which is a topic already considered in depth in process mining. In addition, our claim is that we do not neglect but extend task resequencing techniques considering knock-out activities, as these are discussed in [11]. More specifically, in the algorithm provided, before examining each pair in the sequencing, we can apply node reordering and then to proceed to parallelism investigation. Moving knock out activities upstream is guaranteed to yield lower cycle times and execution costs. Whereas, our parallilized resequencing reduces the cycle times at the expense of an increase in the cost in the generic case. This increase is due to the deliberate suboptimal resource assignment. As shown in the experiments in the next section though, the increase is insignificant compared to the benefits in terms of cycle time.

6. Evaluation

We begin this section by outlining the experimental setting that was used to evaluate our proposal. Then, we proceed to the results' presentation and discussion.

[6] The prototype implementation can be found at `https://github.com/kmvarvou/bpmn_tpst_allocation_optimization`

Table 5. Experimental settings parameters regarding TPSTs

Setting	Nodes	Tree Depth	Tree Breadth
1	18	9	3
2	17	4	8
3	20	5	4

6.1. Experimental Setting

Our software prototype generates process models in the form of TPST trees, and also generates the required metadata. To allow for a comprehensive and wide-scope evaluation, we explore numerous random metadata values. These values cover psychological profiles, behavioral constraints, eligibility and performance metadata, i.e. base cost and duration of execution for each process activity. For each activity, the cost and duration of execution attributes was assigned (integer) values in the [1,10] range. This allows for differences on the order of a magnitude. Additionally, the psychological profiles were assigned random values for each of their RIASEC dimension in the [0,1] range.

Regarding resource eligibility, each activity may be executed by a subset of the full set of resources. We cover two scenarios regarding the eligibility of resources: in the first scenario there are 4 resources: *Clerk, Supervisor, Administrator, Intern*, while in the second scenario there are 8, which are : *Clerk, Supervisor, Administrator, Intern, Automated, Manager, Consultant, Temp*. In the first scenario, all resources have a capacity of 6, while in the second one, they feature a capacity of 3. In other words, both scenarios feature the same amount of distinct resource allocation slots, but in the second scenario, parallelization is inherently easier to achieve due to the larger number of distinct types available. The probability that a node pair features a precedence constraint is 50%. An exception is made for node pairings in the form of *start,a* or *a,end* where *start* is the starting event of the process and *end* is the ending event of the process, where a constraint must always be featured. Lastly, regarding the *ric* values, we again experiment with two scenarios. In the first scenario there is no *ric* overhead. In the second one, *ric* is set to 1 when two consecutive activities are allocated to different resources.

Regarding the process model types, we conducted our experiments by focusing on three distinct TPST cases, with each model featuring different characteristics in terms of depth and breadth. As a reference point, the TPST of the second setting is presented in Figure 6 while the others are in the appendix. The parameters of the experimental evaluation are presented in Table 5 and they cover cases where the tree is (i) narrow and deep, thus leaving little room for reordering improvements; (ii) wide and shallow; and (iii) balanced, with moderate width and depth.

For each of the three TPST designs, we generated 1000 random cases, each one entailing different quantitative and qualitative metadata as described above. We applied all three allocation approaches for each TPST case. Then, our cost-based resequencing approach was applied to each TPST for each of the three allocation approaches. For each approach, we present the number of cases where parallelization was achieved, and also the average and maximum improvement achieved in terms of percentage. The improvements are over the baseline approach without any resequencing.

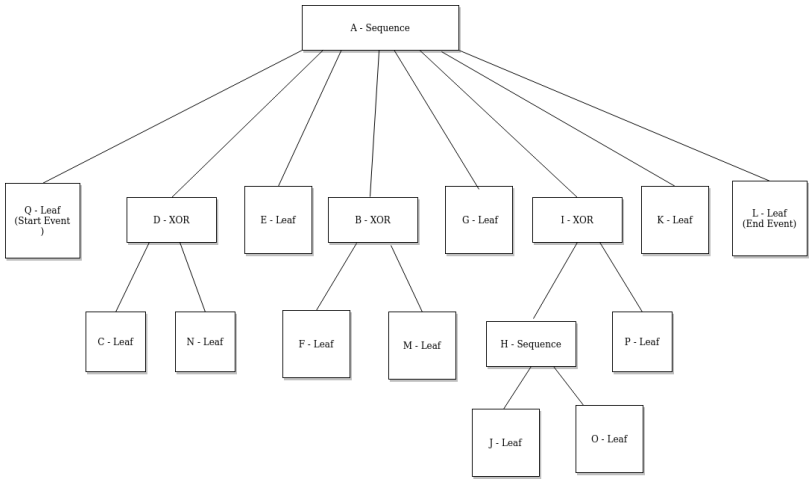

Fig. 6. The TPST of the second experimental setting

Table 6. Results for the 3 experimental settings in terms of the number of cases improved (out of 1000), and the average and maximum improvement in these cases (no *ric*, 4 resources, capacity equal to 6 each)

Setting	Resequencing only			*Advanced_Enforced*			*Advanced_Promoted* (p = 1.0)			*Advanced_Promoted* (p= 0.5)		
	Cases	Average	Maximum	Cases	Average	Maximum	Cases	Average	Maximum	Cases	Average	Maximum
1	248	4.2%	19.9%	502	4.1%	17.35%	556	4.1%	22.76%	510	4.0%	18.3%
2	514	10.7%	29.0%	722	8.7%	35.38%	756	8.5%	36.42%	770	8.4%	31.6%
3	438	12.7%	37%	631	8.8%	44.8%	691	8.2%	39.8%	648	8.4%	36.02%

6.2. Results

We start from the scenario with fewer resources and no *ric* overhead. The results are summarized in Table 6. For each of the three settings (aka TPST types) summarized in Table 5, there is a different row. We compare the effect of resequencing on top of the baseline, the *Advanced_Enforced* proposal and the *Advanced_Promoted* one. For the latter, we investigate two flavors, one with $p = 1$ and another one with $p = 0.5$. For each solution, we mention the number of cases that they lead to improvements in each random set of 1000 instances and the average and maximum improvements in cycle time compared against the cycle time of the baseline resource allocation solution without any resequencing. Note that in terms of total cost, there might be some degradation, but is negligible, i.e., less than 2% at most.

At first glance, the three allocation approaches appear to have complementary benefits with no solution dominating each other in all dimensions. More specifically, in all three settings, applying resequencing on top of the baseline allocation yields higher improvements on average albeit in fewer cases. *Advanced_Enforced* manages to improve more cases than resequencing only, but *Advanced_Promoted* improves even more. Also, there is a trend the more the improvements the less the average improvement.

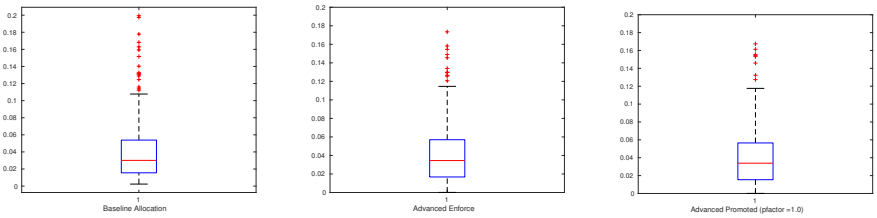

Fig. 7. Results for the first setting

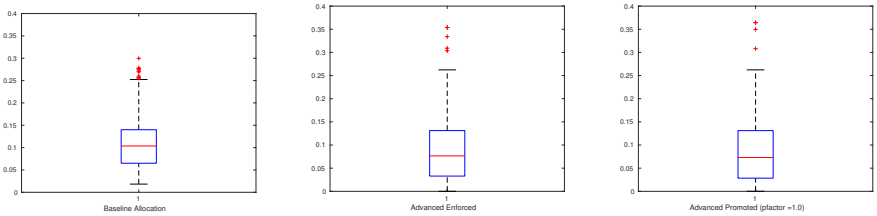

Fig. 8. Results for the second setting

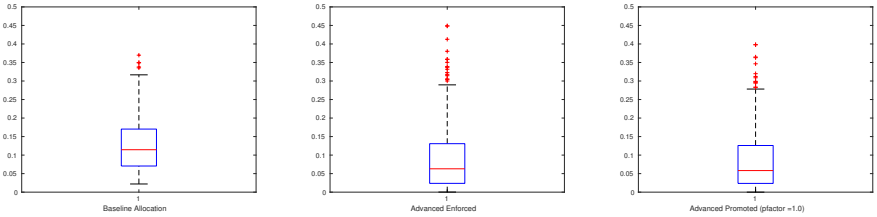

Fig. 9. Results for the third setting

Also, on average the third setting, which corresponds to moderately wide and deep trees, benefits from our solutions the most. This is important, because such trees are encountered more frequently in practice. Another remark is that *Advanced_Promoted* approach is capable of consistently outperforming the other two approaches in all three TPST settings in terms of the cases improved. Compared to the resequencing approach, it was able to yield improvements in 40% more cases (and over 120% in the case of narrow and deep TPSTs). Overall, we can improve more than 75% of the cases for the wide TPSTs and 69.1% of the moderately wide and deep trees, for which the maximum improvements can reach 44.8%.

Based on the observations above, in case we have an efficient simulator at our disposal, we can employ in practice a hybrid solution that first checks the estimates (predictions) if we perform just resequencing over the baseline allocation. Then, if there are no or not satisfactory improvements, to check *Advanced_Enforced*, and finally, to check *Advanced_Promoted*. If we follow this approach, we can combine the highest improvements with the highest number of cases improved.

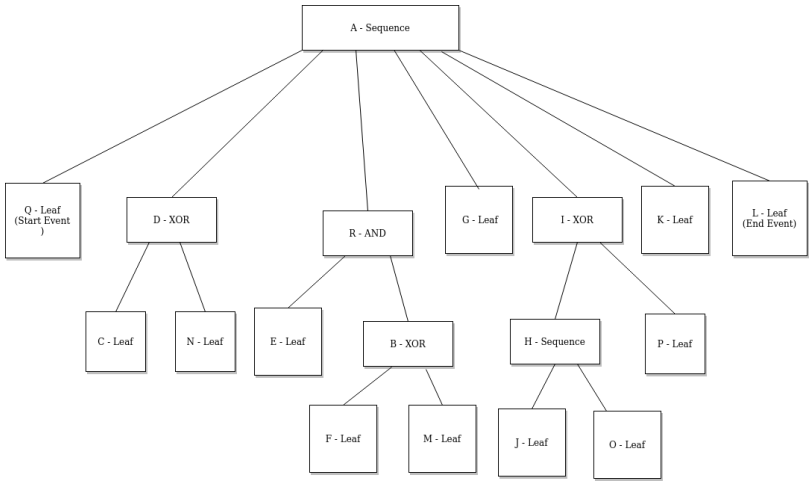

Fig. 10. The result of applying our re-sequencing approach to the TPST of the third experimental setting.

In Table 6, we have presented coarser-grained statistics. In Figures 7,8 and 9, we present the boxplots of improvements for each setting, respectively. We can see that a significant portion of the cases are improved significantly more than the average improvement.

To demonstrate in greater detail the benefits of our proposed approach, we focus in a particular case of the TPST presented in Figure 6. In this specific scenario, nodes E and B do not feature any precedence constraints, meaning that they may be placed in parallel. Additionally, node E has a resource mapping of $\{Clerk, Supervisor\}$. For node B, the resource mapping of its two children nodes, F and M is: $\{Supervisor\}$ and $\{Clerk, Intern, Administrator\}$, respectively. Based on these parameters, the allocations produced by the *Baseline Allocation* approach and *Advanced_Enforced* approach are:

1. $Baseline : E \Rightarrow Supervisor, F \Rightarrow Supervisor, M \Rightarrow Clerk$
2. $Advanced\_Enforced : E \Rightarrow, Clerk, F \Rightarrow Supervisor, M \Rightarrow Intern.$

The allocation produced by the *Advanced_Enforced* approach, enables the placement of nodes E and B in parallel, after applying our resequencing approach, the result of which is presented in Figure 10, leading to a reduction of duration of execution of about 10%. On the contrary, the allocation produced by the baseline allocation approach prevents the placement of the two nodes in parallel.

6.3. Sensitivity Analysis

Next, we aim to discuss the impact that each of the different parameters of the experimental setting has on the performance of the presented allocation approaches. To this end, we continue our experiments testing with *ric* enabled (see Table 7). Then, we repeat both experiments when we increase the resources to 8 (see Tables 8 and 9, respectively). We also assess the impact of the volume of precedence constraints.

Table 7. Results with *ric*, 4 resources, capacity equal to 6 each)

Setting	Resequencing only			*Advanced_Enforced*			*Advanced_Promoted* (p = 1.0)			*Advanced_Promoted* (p= 0.5)		
	Cases	Average	Maximum	Cases	Average	Maximum	Cases	Average	Maximum	Cases	Average	Maximum
1	279	4.4%	23.95%	504	4.4%	25.89%	558	4.36%	26.76%	518	4.31%	25.84%
2	525	10.6%	32.2%	725	8.35%	29.3%	742	8.45%	35.3%	746	8.45%	31.6%
3	431	12.5%	35.8%	665	9.2%	43.4%	680	8.9%	40.8%	662	8.6%	39.8%

Table 8. Results with no *ric*, 8 resources, capacity equal to 3 each)

Setting	Resequencing only			*Advanced_Enforced*			*Advanced_Promoted* (p = 1.0)			*Advanced_Promoted* (p= 0.5)		
	Cases	Average	Maximum	Cases	Average	Maximum	Cases	Average	Maximum	Cases	Average	Maximum
1	347	4.8%	34.8%	494	5.0%	31.7%	580	4.27%	25.8%	532	4.18%	21.2%
2	682	11.58%	31.7%	800	9.73%	32.16%	791	10.22%	36.59%	817	9.78%	32.37%
3	515	13.16%	44.4%	687	9.78%	41.19%	720	9.44%	38.57%	685	9.79%	43.17%

Table 9. Results with *ric*, 8 resources, capacity equal to 3 each)

Setting	Resequencing only			*Advanced_Enforced*			*Advanced_Promoted* (p = 1.0)			*Advanced_Promoted* (p= 0.5)		
	Cases	Average	Maximum	Cases	Average	Maximum	Cases	Average	Maximum	Cases	Average	Maximum
1	337	4.7%	23.1%	482	4.66%	30.0%	585	4.66%	28.4%	531	4.5%	29.5%
2	663	12.24%	35.4%	791	9.87%	41.60%	827	10.24%	35.54%	828	10.39%	36.33%
3	525	13.20%	40.4%	705	10.5%	42.2%	728	9.9%	45.7%	708	10.17%	38.58%

Impact of resource pool size. We begin our discussion with the impact of the size of the resource pool, i.e., the amount of distinct types of resources. Comparing the results of Tables 6 and 7 with the results of Tables 8 and 9, we can observe an increase in the number of cases where parallelization was achieved in all allocation approaches. More specifically, the resequencing approach on top of the baseline allocation exhibited an increase of 34.5% in the number of cases, while the advanced approaches exhibit an average increase of 5.6%, 6.3% and 6.4%, respectively. It should be noted that despite this increase in performance, the simple resequencing approach still trails all advanced approaches by more than 29%. As shown, the advanced allocation approaches exhibit resilient performance, while the resequencing approach with baseline allocation was heavily impacted by the less favorable for parallelization parameters in the first scenario with fewer resources. Overall, the improvements are similar with the highest improvement exceeding 45%, but for wide TPSTs more than 80% of the cases can now be improved; for the 3rd setting, the proportion of improved cases was increased to over 72%.

Impact of resource intercommunication overhead. Here, we focus again on the four same tables but in different combinations. Tables 6 and 8 are compared against Tables 7 and 9, respectively. As we can see, in both settings the incorporation of resource intercommunication overhead did not seem to impact performance in a significant way, with all approaches remaining unaffected in all three dimensions.

Table 10. Results for the TPST of the first experimental setting for three different probabilities of constraint existence with *ric*, 4 resources, capacity equal to 6 each

Probability	Resequencing only			*Advanced_Enforced*			*Advanced_Promoted* (p = 1.0)			*Advanced_Promoted* (p= 0.5)		
	Cases	Average	Maximum	Cases	Average	Maximum	Cases	Average	Maximum	Cases	Average	Maximum
40	349	4.5%	25.4%	513	5.0%	27.1%	586	4.59%	28.5%	564	4.67%	23.9%
50	272	4.2%	23.9%	485	4.15%	28.3%	564	4.3%	24.3%	529	4.4%	19.6%
60	202	3.7%	15%	461	3.8%	22.3%	544	3.83%	17.7%	486	3.9%	21.16%

Impact of precedence constraints. Finally, we would like to examine the impact of different values of probability of a node pair featuring a precedence constraint on performance. In theory, the higher the amount of constraints there are in a model (i.e. higher probability) then the lower the possibility that some of its activities may be parallelized. For this experiment, we focused on the narrow and deep TPST, using probability values of 40%. 50% and 60%. The results are presented in Table 10. As we can see, the resequencing on top of the baseline allocation approach exhibits a 26 % decrease in the number of cases identified when the probability increases from 50% to 60%. On the other hand, it exhibits a 28 & increase in the number of cases identified, when the probability decreases to 40 %. Again, this approach displays a high sensitivity, being affected by the setting's parameters. On the contrary, both advanced allocation approaches displayed higher resiliency in terms of the number of cases they managed to improve. Finally, the average improvements in the cycle time decrease as there are more constraints.

6.4. Summary of observations

Here, we provide a summary of the key observations.

- There is no clear winner between our proposals. We see that we can achieve highest improvements with approaches that manage to improve fewer cases in general.
- The observation above is not actually a limitation. The solutions need not be considered as competitors and can be combined to form a hybrid ensemble solutions, where all are tested at the beginning and the best performing one can be chosen.
- We also note the high insensitivity of the proposed methods in terms of resource pool size, intercommunication overhead and extent of precedence constraints.

7. Related Work

The main cost-based techniques that perform activity reordering in BPMN diagrams leverage the existence of knock-out activities, as explained in [1], or utilize one of the redesign patterns outlined in [18]. The work in [18] presents a set of redesign heuristics, including parallelization, which is partially aligned with the focus of our work, in an attempt to identify the best practices in the field of Business Process Redesign (BPR). In a similar context, a subset of these heuristics have been implemented in [7] as part of an assisted BPR approach. However, contrary to our work, this approach focuses mainly on the control-flow aspect of the process. These techniques are extended with recent advances in

dataflow and query processing optimization [10], as explained in [11]. Similar techniques based on heuristics have been also proposed for declarative models, such as PDM (e.g., [23]). As already stated, we differ in that we perform cost-based resequencing without relying on the existence of knock-outs.

Additionally, there are proposals considering issues of automatic Business Process Redesign. The work in [5] presents an approach that aims to redesign processes in an automatic way by utilizing insights provided by process mining. However, this approach does not take into account the resource perspective and also does not include any evaluation. Furthermore, the work in[20] focuses on configurable process models. Their approach aims to simulate automatically generated variants of the input model and identify the best-performing ones, including previously undiscovered variants.

Up to date, a plethora of objective functions and cost models have already been proposed and applied both for dataflows and business processes, but there are have not been examined in parallel and distributed environments sufficiently. Therefore, there is a need to adopt a cost model that will take into account the parallel execution of the BP activities. The BP execution requires to take into consideration the probability distributions of the input data tokens, the waiting times or the cost of the occupied resources in a realistic manner; these challenges are aligned to the effort to construct digital twins for BPs [3] and are orthogonal to our proposal; the optimization we propose relies on a good and realistic cost model but is not tightly coupled with any specific one. Similarly, the authors of [21] propose building predictive and prescriptive models. The former model estimates the undesired case outcome probability. The latter one refers to a causal model that estimates the impact of a given intervention. Our resequencing proposal can benefit from advances in cost models for BPs to better assess the impact of resequencing.

A significant portion of recent research has highlighted the importance of resource allocation in the context of BPR and BP optimization. The work in [14] presents an allocation approach that aims to represent more accurately the unique characteristics of workers (i.e. resources) by utilizing a personality assessment framework. The framework used is *Holland's person–job fit theory (HPJFT)* upon which we also build our work. In a similar context, the work in [12] proposes an allocation approach that focuses on maximizing cooperation between resources. We differ in that we leverage a resource allocation approach to raise more reorderability opportunities without being tightly coupled with a specific resource allocation proposal.

Overall, our work also relates to resource allocation optimization proposals but differs in that it leverages an existing, potentially optimized resource allocation for activity resequencing rather than targeting on resource allocation as its final goal. Examples of resource allocation appear in [13], where the trade-off between cycle time and resource cost is examined. Additionally, the proposal in [8] discusses an allocation technique to minimize the cloud infrastructure costs in the terms of resource (CPU, RAM, Database size) consumption when executing real-world BPs with different number of simulated users. Other examples of resource allocation techniques achieving resource balancing can be found in [16],[28]. All these proposals are orthogonal to our work as well.

8. Conclusions

In this work, we deal with the problem of proposing a systematic approach to reordering BPMN activities and putting them in parallel. We start with advocating to leverage a given resource allocation with regards to a BPMN model in order to reorder activities so that they can be executed in parallel. Then, we move to proposing techniques that modify the resource allocation so that more reorderability opportunities arise. Our solutions modify the BPMN diagram through inserting AND blocks and moving (blocks of) activities to other places. The intermediate representation that we employ are TPSTs, while we respect all relevant behavioral constraints. We show that we can yield improvement in a very high number of cases examined (in some settings, exceeding 80%), while both the average and the maximum decreases in cycle time are important. For example, the maximum observed improvement exceeded 45%. In the future, we aim to extend our work to optimize several process instances together rather than treating each process instance in isolation.

Acknowledgments. The research work was supported by the Hellenic Foundation for Research and Innovation (H.F.R.I.) under the "First Call for H.F.R.I. Research Projects to support Faculty members and Researchers and the procurement of high-cost research equipment grant" (Project Number:1052, Project Name: DataflowOpt) and by the European Commission under the Horizon 2020 Programme, through funding of the Trineflex project (Grant 101058174). We would also like to acknowledge the support of Dr. Christos Bellas during the implementation.

References

1. van der Aalst, W.M.P.: Re-engineering knock-out processes. Decis. Support Syst. 30(4), 451–468 (2001)
2. De Smedt, J., Deeva, G., De Weerdt, J.: Mining behavioral sequence constraints for classification. IEEE Transactions on Knowledge and Data Engineering 32(6), 1130–1142 (2020)
3. Dumas, M.: Constructing digital twins for accurate and reliable what-if business process analysis. In: Proceedings of the International Workshop on BPM Problems to Solve Before We Die (PROBLEMS 2021). CEUR Workshop Proceedings, vol. 2938, pp. 23–27 (2021)
4. Dumas, M., Rosa, M.L., Mendling, J., Reijers, H.A.: Fundamentals of Business Process Management, Second Edition. Springer (2018)
5. Essam, M., Mansar, S.L.: Towards a software framework for automatic business process redesign (2011)
6. Fan, J., Wang, J., An, W., Cao, B., Dong, T.: Detecting difference between process models based on the refined process structure tree. Mob. Inf. Syst. 2017, 6389567:1–6389567:17 (2017)
7. Fehrer, T., Fischer, D.A., Leemans, S.J., Röglinger, M., Wynn, M.T.: An assisted approach to business process redesign. Decision Support Systems p. 113749 (2022)
8. Ferme, V., Ivanchikj, A., Pautasso, C.: Estimating the cost for executing business processes in the cloud. In: La Rosa, M., Loos, P., Pastor, O. (eds.) Business Process Management Forum. pp. 72–88 (2016)
9. Kougka, G., Gounaris, A.: Optimization of data flow execution in a parallel environment. Distributed Parallel Databases 37(3), 385–410 (2019)
10. Kougka, G., Gounaris, A., Simitsis, A.: The many faces of data-centric workflow optimization: a survey. Int. J. Data Sci. Anal. 6(2), 81–107 (2018)

11. Kougka, G., Varvoutas, K., Gounaris, A., Tsakalidis, G., Vergidis, K.: On knowledge transfer from cost-based optimization of data-centric workflows to business process redesign. Trans. Large Scale Data Knowl. Centered Syst. 43, 62–85 (2020)
12. Kumar, A., Dijkman, R., Song, M.: Optimal resource assignment in workflows for maximizing cooperation. pp. 235–250 (08 2013)
13. López-Pintado, O., Dumas, M., Yerokhin, M., Maggi, F.M.: Silhouetting the cost-time front: Multi-objective resource optimization in business processes. In: Business Process Management Forum. pp. 92–108 (2021)
14. Pereira, J.L., Varajão, J., Uahi, R.: A new approach for improving work distribution in business processes supported by bpms. Business Process Management Journal ahead-of-print (03 2020)
15. Pesic, M., Schonenberg, H., van der Aalst, W.M.P.: DECLARE: full support for loosely-structured processes. In: 11th IEEE International Enterprise Distributed Object Computing Conference (EDOC 2007), 15-19 October 2007, Annapolis, Maryland, USA. pp. 287–300 (2007)
16. Peters, S.P.F., Dijkman, R.M., Grefen, P.W.P.J.: Resource optimization in business processes. In: 2021 IEEE 25th International Enterprise Distributed Object Computing Conference (EDOC). pp. 104–113 (2021)
17. Polyvyanyy, A., García-Bañuelos, L., Dumas, M.: Structuring acyclic process models. Inf. Syst. 37(6), 518–538 (2012)
18. Reijers, H., Mansar, S.: Best practices in business process redesign: An overview and qualitative evaluation of successful redesign heuristics. Omega 33, 283–306 (08 2005)
19. Reijers, H.A., Vanderfeesten, I.T.P., Plomp, M.G.A., Gorp, P.V., Fahland, D., van der Crommert, W.L.M., Garcia, H.D.D.: Evaluating data-centric process approaches: Does the human factor factor in? Software and Systems Modeling 16(3), 649–662 (2017)
20. Schunselaar, D.D., Verbeek, H.E., van der Aalst, W.M., Reijers, H.A.: Petra : Process model based extensible toolset for redesign and analysis (2014)
21. Shoush, M., Dumas, M.: Prescriptive process monitoring under resource constraints: A causal inference approach. CoRR abs/2109.02894 (2021)
22. Tsakalidis, G., Vergidis, K., Kougka, G., Gounaris, A.: Eligibility of bpmn models for business process redesign. Information 10(7) (2019)
23. Vanderfeesten, I.T.P., Reijers, H.A., van der Aalst, W.M.P.: Product-based workflow support. Inf. Syst. 36(2), 517–535 (2011)
24. Vanhatalo, J., Völzer, H., Koehler, J.: The refined process structure tree. Data Knowl. Eng. 68(9), 793–818 (2009)
25. Varvoutas, K., Gounaris, A.: Evaluation of heuristics for product data models. In: Business Process Management BPM Workshops. pp. 355–366 (2020)
26. Varvoutas, K., Kougka, G., Gounaris, A.: Optimizing business processes through parallel task execution. In: Proceedings of the 14th International Conference on Management of Digital EcoSystems, MEDES. pp. 24–31. ACM (2022)
27. Vergidis, K., Tiwari, A., Majeed, B.: Business process analysis and optimization: Beyond reengineering. Trans. Sys. Man Cyber Part C 38(1), 69–82 (2008)
28. Yaghoubi, M., Zahedi, M.: Tuning concurrency of the business process by dynamic programming. pp. 1–5 (02 2018)

Konstantinos Varvoutas is a Ph.D. student at the Aristotle University of Thessaloniki. His main research interests lie in the fields of business process optimization and data intensive computing.

Georgia Kougka is a postdoc in Aristotle University of Thessaloniki and her research interests lie in the field of data science for industrial purposes, data-centric flow modeling, optimization and execution, but also in business process optimization. Moreover, she

has also participated in several research programs and initiatives for predictive maintenance purposes in Industry 4.0 scenarios. Finally, she deals with data analysis in smart agriculture and energy disaggregation scenarios.

Anastasios Gounaris is an associate professor at the Aristotle University of Thessaloniki, Greece. A. Gounaris received his PhD from the University of Manchester (UK) in 2005. His main research interests are in the areas of large-scale data management, massive parallelism, workflow and business process optimization, big data analytics, and data mining. More details can be found at `http://datalab.csd.auth.gr/gounaris/`

Received: April 01, 2023; Accepted: July 17, 2023.

A. TPSTs used in the experiments

We have already shown in Figure 6, the TPST corresponding to one of the three experimental settings. In Figures 11 and 12 the remaining two settings are presented.

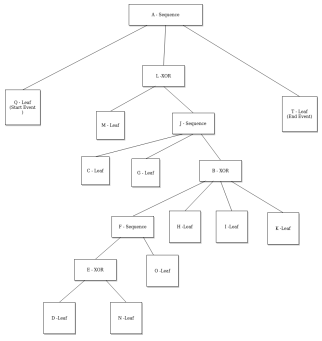

Fig. 11. The TPST of the first experimental setting

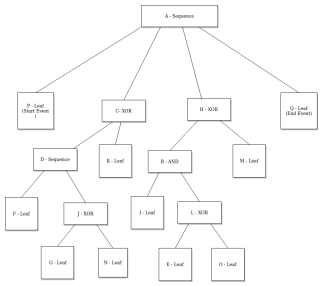

Fig. 12. The TPST of the third experimental setting