Optimizing Business Processes Through Parallel Task Execution

Konstantinos Varvoutas
Aristotle University of Thessaloniki
Greece
kmvarvou@csd.auth.gr

Georgia Kougka
Aristotle University of Thessaloniki
Greece
georkoug@csd.auth.gr

Anastasios Gounaris
Aristotle University of Thessaloniki
Greece
gounaria@csd.auth.gr

ABSTRACT

Optimization of business processes is a persistent topic and a key goal in business process management (BPM). In this work, we investigate how a given resource allocation in business processes can drive optimizations in an underlying BPMN diagram. More specifically, the main contribution is a proposal to leverage a variant of representation of processes as Refined Process Structure Trees (RPSTs) with a view to enabling novel resource allocation-driven task re-ordering in a principled manner. The re-orderings targeted enforce the parallelism redesign heuristic that allows for multiple resources operating concurrently, which yields improvements in the process cycle time.

CCS CONCEPTS

• **Applied computing** → **Business process management systems**; • **Information systems** → *Enterprise information systems.*

KEYWORDS

business process optimization, process models, resequencing

ACM Reference Format:
Konstantinos Varvoutas, Georgia Kougka, and Anastasios Gounaris. 2022. Optimizing Business Processes Through Parallel Task Execution. In *International Conference on Management of Digital EcoSystems (MEDES '22), October 19–21, 2022, Venice, Italy.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3508397.3564842

1 INTRODUCTION

Digital ecosystems and business processes are tightly coupled in practice, given that the latter are largely facilitated by the former.[1] 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, is a key aspect in modern BPM. However, up to date, there is no automated optimization technique for BPs that can take advantage from the overlapping task execution in order to improve latency (a.k.a. cycle time) and is generally applicable. In general, automated solutions for BP optimization have not been explored as deeply as process modelling, as discussed in several places, e.g., [3, 13, 19].

We target scenarios where a BP is modelled with the help of a procedural approach, such as BPMN[2], and the optimizations on which we focus fall under the BP behavior heuristics according to the taxonomy in [3]. 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 is different than the initial one. The latter is modified so that certain objectives, such as cycle time or total cost, are improved. 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 [8, 15, 18]; knock-out activities are the activities that can lead to immediate process termination, such as automatically reject an application if it does not meet certain criteria. 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 this work, we introduce a novel combination of resequencing 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 [4] of representation of processes as Refined Process Structure Trees (RPST) [17]. This representation allows us 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. We illustrate and validate the effectiveness of our approach through a use-case example.

The remainder of this work is structured as follows. Next, we present our exemplary use case. In Section 3, we provide the background regarding RPST. In Section 4, we discuss our proposal. We continue with a discussion of the related work in Section 5. We conclude in Section 6.

2 OUR CASE STUDY

Our case study is shown in Figure 1 and refers to common real-world BP regarding an employee expense reimbursement request[3]. Briefly, the BP consists of eight 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

[1]https://www.cleo.com/blog/digital-ecosystems

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

[3]https://www.businessprocessincubator.com/

MEDES '22, October 19–21, 2022, Venice, Italy
© 2022 Association for Computing Machinery.
ACM ISBN 978-1-4503-9219-8/22/10…$15.00
https://doi.org/10.1145/3508397.3564842

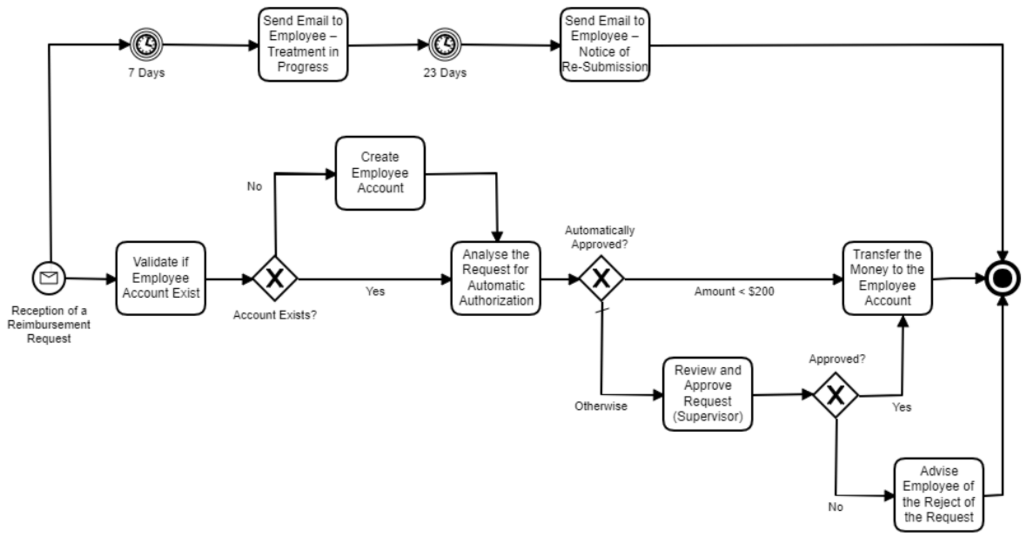

7 Days
Send Email to Employee – Treatment in Progress
23 Days
Send Email to Employee – Notice of Re-Submission
Create Employee Account
No
Reception of a Reimbursement Request
Validate if Employee Account Exist
Account Exists?
Yes
Analyse the Request for Automatic Authorization
Automatically Approved?
Amount < $200
Transfer the Money to the Employee Account
Otherwise
Review and Approve Request (Supervisor)
Approved?
Yes
No
Advise Employee of the Reject of the Request

Figure 1: The process model of an Employee Expense Reimbursement Request.

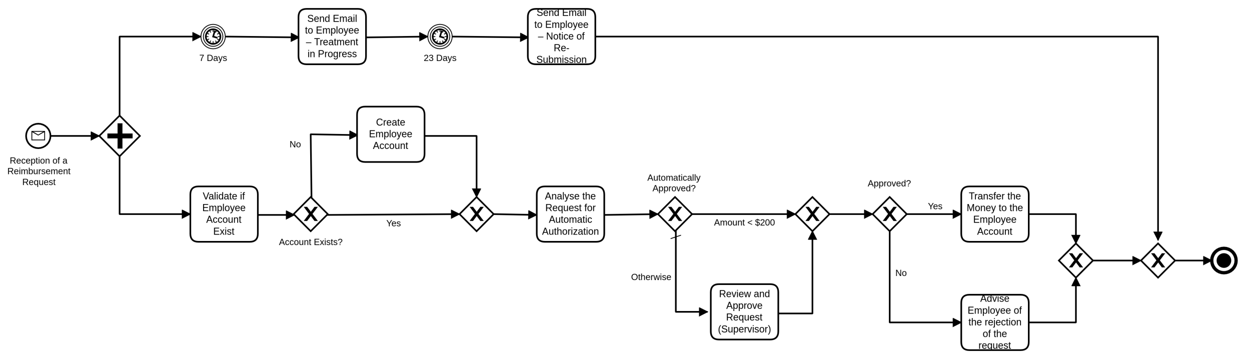

7 Days
Send Email to Employee – Treatment in Progress
23 Days
Send Email to Employee – Notice of Re-Submission
Reception of a Reimbursement Request
Create Employee Account
No
Validate if Employee Account Exist
Account Exists?
Yes
Analyse the Request for Automatic Authorization
Automatically Approved?
Amount < $200
Otherwise
Review and Approve Request (Supervisor)
Approved?
Yes
No
Transfer the Money to the Employee Account
Advise Employee of the rejection of the request

Figure 2: The well-structured process model of an Employee Expense Reimbursement Request

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. [12]. 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., [8, 15]. 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 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 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

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 1: Case Study Activity Cycle Times and Costs.

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

Table 2: Case Study Path Probabilities.

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., [3]. 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)[17] and more specifically a specific variant of RPST, called Task Based Process Structure Tree (TPST) [4]. 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 [4], where a business process model is separated into blocks, which are then 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, 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.

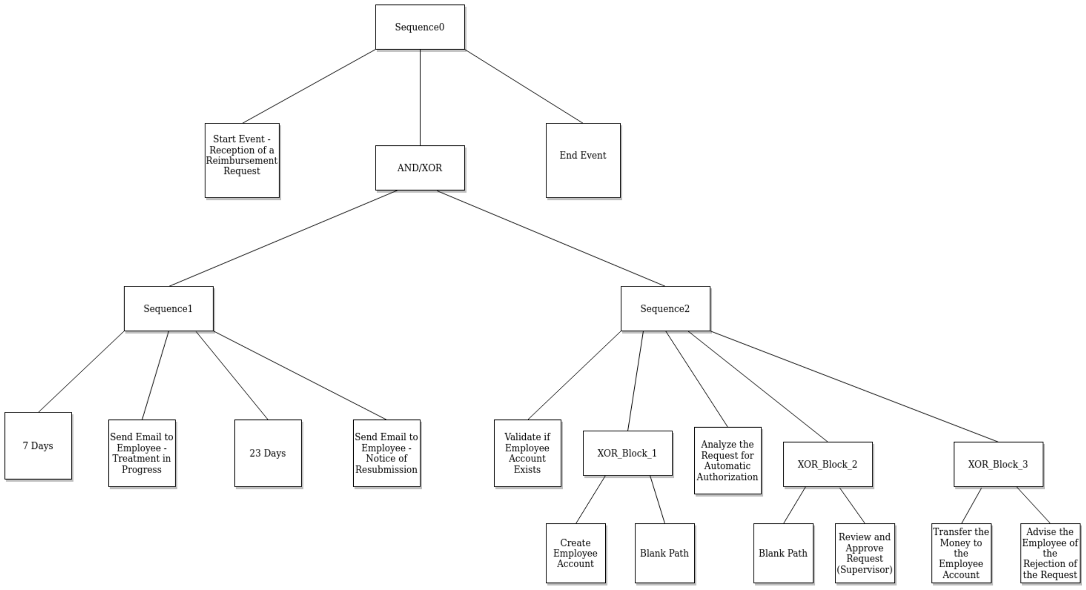

Sequence0
Start Event - Reception of a Reimbursement Request
AND/XOR
End Event
Sequence1
Sequence2
7 Days
Send Email to Employee - Treatment in Progress
23 Days
Send Email to Employee - Notice of Resubmission
Validate if Employee Account Exists
XOR_Block_1
Analyze the Request for Automatic Authorization
XOR_Block_2
XOR_Block_3
Create Employee Account
Blank Path
Blank Path
Review and Approve Request (Supervisor)
Transfer the Money to the Employee Account
Advise the Employee of the Rejection of the Request

Figure 3: The TPST of our case study BP model.

3.2 Decomposition of our scenario and cost modelling

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 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.

Let $ct(a_i)$ denote the cycle time of activity a_i, where i is the activity id as defined in Table 1. Then, based on the above, the cycle time of the case study process is the minimum of the two sequences in Figure 3. These are computed as follows based on the classical formulas in textbooks, such as [3] while assuming the metadata in Tables 1 and 2 and that a day is equal to 8 hours.

$ct(Sequence1) = ct(Timer7) + ct(a_1) + ct(Timer23) + ct(a_2) = 30\ days\ and\ 2\ mins$

$ct(Sequence2) = ct(a_3) + 0.2ct(a_4) + a_5 + 0.2ct(a_6) + 0.6ct(a_7) + 0.4ct(a_8) = 2\ days\ and\ 6.2\ hours$

The total cycle time of the TPST model is calculated by the equation: $ct(TPST) = \min\{ct(Sequence1), ct(sequence2)\}$

The total cost can be calculated using a similar rationale, but we omit details, since, in this case study, we target the reduction in cycle time.

3.3 Limitations of this approach and discussion

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.

Essentially, our resequencing technique is independent of any model to compute the process cycle time and cost. During resequencing, several alternative model 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[4] or digital twins [2], can be also employed.

4 TPST-BASED TASK RE-ORDERING

Our methodology accepts as input the TPST representation of the BPMN model and aims to produce an optimized model. To this

[4]https://bimp.cs.ut.ee/simulator/

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

Table 3: Behavioral constraints considered.

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 approach is summarized below.

- $A = \{a_1, ..., a_n\}$ defines a set of n activities that appear in the BPMN business process model. The cycle time (resp. cost) of each activity $a_i,\ i = 1...n$ is denoted as $ct(a_i)$ (resp. $c(a_i)$)
- $R = \{r_1, ..., r_m\}$ 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 $i \in 1...n$ and $j \in 1...m$.

In addition, we consider a subset of the constraints defined by DECLARE [10], 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., [1], 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 different branches but executed by the same resource are treated as executing sequentially in terms of their cycle time.

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. 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 [3].

(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 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.

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 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
```

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 [6, 7], 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.[5] The algorithm accepts as input the TPST representation of the input model as shown in Figure 3. The resource annotation (not

[5]Our prototype implementation can be found at https://github.com/kmvarvou/bpmn_tpst_optimization

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)*

Table 4: Case Study Behavioral Constraints

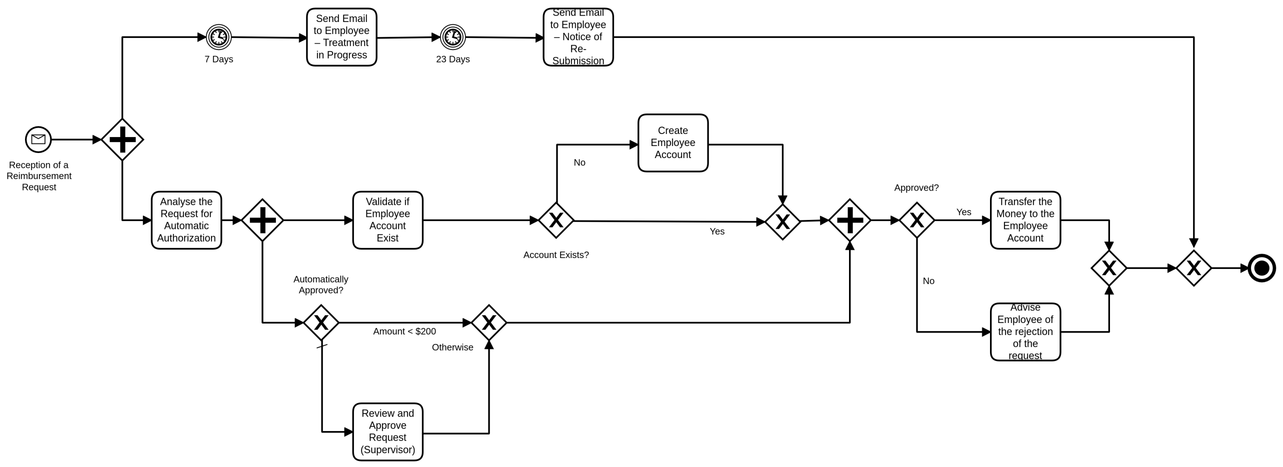
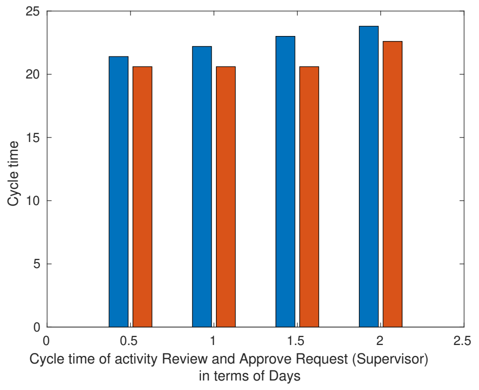

7 Days
Send Email to Employee – Treatment in Progress
23 Days
Send Email to Employee – Notice of Re-Submission
Reception of a Reimbursement Request
Analyse the Request for Automatic Authorization
Validate if Employee Account Exist
Create Employee Account
No
Yes
Account Exists?
Approved?
Yes
Transfer the Money to the Employee Account
No
Advise Employee of the rejection of the request
Automatically Approved?
Amount < $200
Otherwise
Review and Approve Request (Supervisor)

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



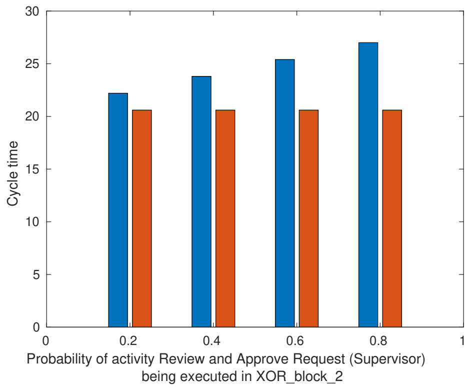


Figure 5: Process cycle times for different values in terms of cycle time of the *Review and Approve Request (Supervisor)* activity (left) and the probability of the activity being executed in *XOR_block_2* (right). The blue bars refer to the original model and the orange bars to the optimized one.

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.[6] The resulting BPMN is presented in Figure 4.

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.

4.4 Further Issues

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 loosely coupled role of the cost model already discussed, we focus also on two additional aspects below.

4.4.1 Metadata Acquisition. The proposed technique, similarly to any cost-based technique, relies on accurate quantitative and qualitative metadata. We expect that the individual cycle times, possibly dependent on the resource allocation and allowing for uncertainty, can be provided by process mining techniques applied on previous logs or domain experts or both. This also applies to constraints derivation, which is a topic already considered in depth in process mining. Finally, we assume that for the specific BP instance, the resource allocation is known. However, a more complete solution would consider a batch of process instances, for which the resource allocation could be looser and referring to resource types rather than individual resources.

4.4.2 Knock-out activities. Our claim is that we do not neglect but extend task resequencing techniques considering knock-out activities, as these are discussed in [8]. 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.

5 RELATED WORK

The main cost-based techniques that perform activity reordering in BPMN diagrams leverage the existence of knock-out activities, as explained in [15]. These techniques are extended with recent advances in dataflow and query processing optimization [7], as explained in [8]. Similar techniques based on heuristics have been also proposed for declarative models, such as PDM (e.g., [16]). As already stated, we differ in that we perform cost-based resequencing without relying on the existence of knock-outs.

Up to date, a plethora of objective functions and cost models have already proposed and applied both for dataflows and business processes, but there are have not examined in parallel and distributed environments sufficiently. Therefore, there is a need to adopt a cost model that will take 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 [2] and are orthogonal to our proposal; the optimization we described relies on a good and realistic cost model but is not tightly coupled with any specific one. Similarly, the authors of [14] 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.

Our work also relates to resource optimization proposals but differs in that it leverages an existing, potentially optimized resource allocation for activity resequencing, rather than deciding on the resource allocation in its own right. Examples of resource allocation appear in [9], where the trade-off between cycle time and resource cost is examined. Additionally, the proposal in [5] 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 [11, 20]. All these proposals are orthogonal to our work as well.

6 CONCLUSIONS

In this work, we advocate 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 and their cycle times to overlap; to this end, we automatically modify the BPMN diagram through inserting AND blocks and moving (blocks of) activities to other places. We employ TPST as an intermediate representation format, while we respect all relevant behavioral constraints. Example results in a real-world case study reveal that there can be tangible benefits in the process cycle time, e.g., reductions of above 20%.

In the future, we aim to extend our work in two main direction. Firstly, to optimize both the model structure and the resource allocation, and secondly 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).

REFERENCES

[1] Johannes De Smedt, Galina Deeva, and Jochen De Weerdt. 2020. Mining Behavioral Sequence Constraints for Classification. *IEEE Transactions on Knowledge and Data Engineering* 32, 6 (2020), 1130–1142. https://doi.org/10.1109/TKDE.2019.2897311

[6] 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.

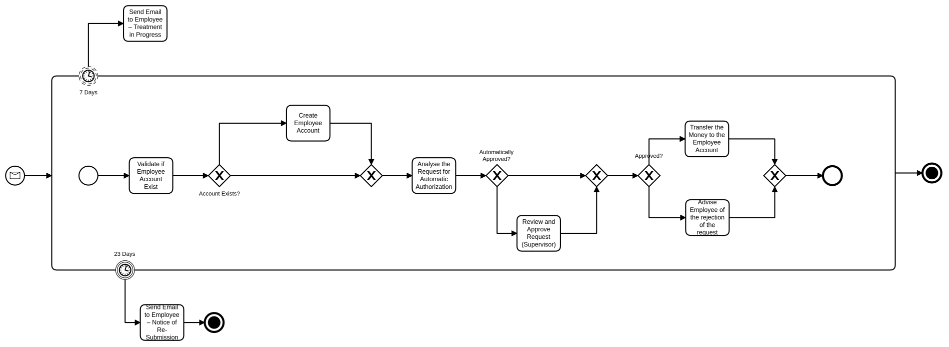

Send Email to Employee – Treatment in Progress
7 Days
Validate if Employee Account Exist
Account Exists?
Create Employee Account
Analyse the Request for Automatic Authorization
Automatically Approved?
Review and Approve Request (Supervisor)
Approved?
Transfer the Money to the Employee Account
Advise Employee of the rejection of the request
23 Days
Send Email to Employee – Notice of Re-Submission

Figure 6: The well-structured process model of an Employee Expense Reimbursement Request

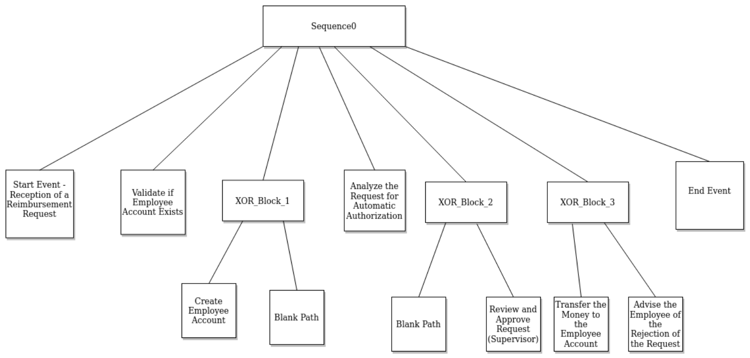

Sequence0
Start Event - Reception of a Reimbursement Request
Validate if Employee Account Exists
XOR_Block_1
Analyze the Request for Automatic Authorization
XOR_Block_2
XOR_Block_3
End Event
Create Employee Account
Blank Path
Blank Path
Review and Approve Request (Supervisor)
Transfer the Money to the Employee Account
Advise the Employee of the Rejection of the Request

Figure 7: The TPST of our case study BP model.

[2] Marlon Dumas. 2021. 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)*. 23–27.

[3] Marlon Dumas, Marcello La Rosa, Jan Mendling, and Hajo A. Reijers. 2018. *Fundamentals of Business Process Management, Second Edition.* Springer.

[4] Jing Fan, Jiaxing Wang, Weishi An, Bin Cao, and Tianyang Dong. 2017. Detecting Difference between Process Models Based on the Refined Process Structure Tree. *Mob. Inf. Syst.* 2017 (2017), 6389567:1–6389567:17.

[5] Vincenzo Ferme, Ana Ivanchikj, and Cesare Pautasso. 2016. Estimating the Cost for Executing Business Processes in the Cloud. In *Business Process Management Forum*, Marcello La Rosa, Peter Loos, and Oscar Pastor (Eds.). 72–88.

[6] Georgia Kougka and Anastasios Gounaris. 2019. Optimization of data flow execution in a parallel environment. *Distributed Parallel Databases* 37, 3 (2019), 385–410.

[7] Georgia Kougka, Anastasios Gounaris, and Alkis Simitsis. 2018. The many faces of data-centric workflow optimization: a survey. *Int. J. Data Sci. Anal.* 6, 2 (2018), 81–107.

[8] Georgia Kougka, Konstantinos Varvoutas, Anastasios Gounaris, George Tsakalidis, and Kostas Vergidis. 2020. On Knowledge Transfer from Cost-Based Optimization of Data-Centric Workflows to Business Process Redesign. *Trans. Large Scale Data Knowl. Centered Syst.* 43 (2020), 62–85.

[9] Orlenys López-Pintado, Marlon Dumas, Maksym Yerokhin, and Fabrizio Maria Maggi. 2021. Silhouetting the Cost-Time Front: Multi-objective Resource Optimization in Business Processes. In *Business Process Management Forum.* 92–108.

[10] Maja Pesic, Helen Schonenberg, and Wil M. P. van der Aalst. 2007. 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.* 287–300.

[11] S. P. F. Peters, R. M. Dijkman, and P. W. P. J. Grefen. 2021. Resource Optimization in Business Processes. In *2021 IEEE 25th International Enterprise Distributed Object Computing Conference (EDOC).* 104–113.

[12] Artem Polyvyanyy, Luciano García-Bañuelos, and Marlon Dumas. 2012. Structuring acyclic process models. *Inf. Syst.* 37, 6 (2012), 518–538.

[13] Hajo A. Reijers, Irene T. P. Vanderfeesten, Marijn G. A. Plomp, Pieter Van Gorp, Dirk Fahland, Wim L. M. van der Crommert, and H. Daniel Diaz Garcia. 2017. Evaluating data-centric process approaches: Does the human factor factor in? *Software and Systems Modeling* 16, 3 (2017), 649–662.

[14] Mahmoud Shoush and Marlon Dumas. 2021. Prescriptive Process Monitoring Under Resource Constraints: A Causal Inference Approach. *CoRR* abs/2109.02894 (2021).

[15] W. M. P. van der Aalst. 2001. Re-engineering Knock-out Processes. *Decis. Support Syst.* 30, 4 (2001), 451–468.

[16] Irene T. P. Vanderfeesten, Hajo A. Reijers, and Wil M. P. van der Aalst. 2011. Product-based workflow support. *Inf. Syst.* 36, 2 (2011), 517–535.

[17] Jussi Vanhatalo, Hagen Völzer, and Jana Koehler. 2009. The refined process structure tree. *Data Knowl. Eng.* 68, 9 (2009), 793–818.

[18] Konstantinos Varvoutas and Anastasios Gounaris. 2020. Evaluation of Heuristics for Product Data Models. In *Business Process Management BPM Workshops.* 355–366.

[19] K. Vergidis, A. Tiwari, and B. Majeed. 2008. Business Process Analysis and Optimization: Beyond Reengineering. *Trans. Sys. Man Cyber Part C* 38, 1 (2008), 69–82.

[20] Mehdi Yaghoubi and Morteza Zahedi. 2018. Tuning Concurrency of the Business Process by Dynamic Programming. 1–5.

A ALTERNATIVE MODELLING

A question may arise as to whether our technique is tailored to the example case study. Furthermore, the inclusion of and AND/XOR node in the TPST diagram may raise suspicions that we deal with a very rare case. Although we do not provide formalities regarding BPMN and TPST, the proposed technique is generic and independent of any specific well-structured model. For example, in Figure 6, we provide a different but equivalent well-structured BPMN diagram, which corresponds to a modified TPST (see Figure 7). All optimizations would be the same.