Evaluation of Heuristics for Product Data Models

Konstantinos Varvoutas and Anastasios Gounaris

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

Abstract. Product Data Model (PDM) is an example of a data-centric approach to modelling information-intensive business processes, which offers flexibility and facilitates process optimization. It is declarative, and as such, there may be multiple workflow designs that can produce the end product. To this end, several heuristics have been proposed. The contributions of this work are twofold: (i) we propose new heuristics that capitalize on established techniques for optimizing data-intensive workflows; and (ii) we extensively evaluate the existing solutions. Our results shed light on the merits of each heuristic and show that our proposal can yield significant benefits in certain cases. We provide our implementation as an open-source product.

Keywords: data-centric processes · process optimization · PDM

1 Introduction

Data-centric approaches have been emerging in the last two decades as an alternative to the more mainstream activity-oriented modelling approaches for business processes [15, 11, 8]. We quote from [15] that *"the central idea behind data-centric approaches is that data objects/elements/artifacts can be used to enhance a process-oriented design or even to serve as the fundament for such a design. This has certain advantages, varying from increasing flexibility in process execution and improving reusability to actually being able to capture processes where data play a relevant role."*

In this work, we focus on a particular data-centric modelling approach, namely a *Product Data Model (PDM)*-oriented one, for which the main driver, as also reported in [15], is process optimization apart from flexibility; this approach is tailored to information-intensive processes and it is declarative. As such, it focuses on describing what is needed in order to deliver an information product rather than the exact way to achieve this goal. To fulfill the latter aspect, the declarative model is accompanied by a method to generate workflow designs, which is referred to as *Product Based Workflow Support (PBWS)* [19]. PBWS presents a set of heuristics for PDMs with a view to enhancing the performance on a case-by-case manner. PBWS improves upon a previous method, called *Product Based Workflow Design (PBWD)* [14], where the burden of defining the sequence of actions rests with the workflow designer, while PBWD merely assists this task through presenting the alternatives.

Heuristic solutions are intuitive in their rationale, easy to implement and are of low computational complexity. However, the existing solutions fail to benefit from established techniques in the areas of optimization for workflows for data analytics and database query execution plans, which adopt principled cost-based approaches [9]. Inspired by such techniques, in our heuristics, we suggest to consider both the time/cost of each operation in a PDM model and the probability of this operation to lead to an early termination of the process, thus saving time and resources. More specifically, we make a twofold contribution:

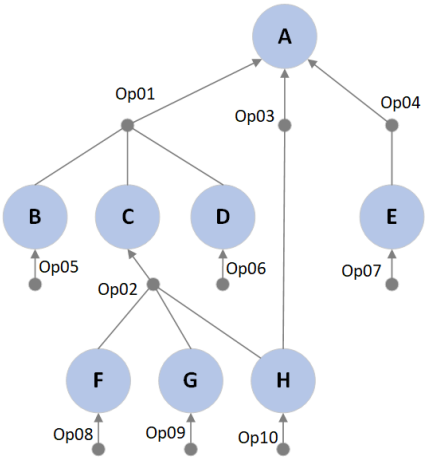
1. We propose a new heuristic for choosing the next operation to be performed in a PDM for a specific case to optimize time duration and/or cost. Our proposal comes in three flavors and is based on established query processing and data-centric workflow technology, and is of low computational complexity.
2. We perform an extensive experimental evaluation of the available heuristics and we show that our proposal yields benefits in terms of time, cost or a combination of both compared to previous heuristics, on the average case. However, there is no globally dominant solution, in the sense that in specific cases, existing heuristics may behave better. We provide the open source of all the heuristics and the experiments, so that interested third parties can repeat our work and extend the set of heuristics and/or test cases.[1]

The remainder of this paper is structured as follows. In Section 2, we present the PDM underpinnings of the techniques evaluated. Section 3 discusses existing solutions from PBWS and introduces our proposal along with implementation details. We evaluate the candidate techniques in Section 4. The next section deals with the related work and we conclude in Section 6, where we also briefly discuss limitations and future work.

2 Background: the Product Data Model

A PDM is used to represent the structure of a workflow product in a rooted graph-like manner, similar to a Bill of Material [19, 12]. PDMs describe the required elements for yielding the desired product in the root, where example (informational) products include decision on whether to grant an approval to a specific admission request, approval of a mortgage application, and so on. More specifically, the vertices (or equivalently nodes) in this structure correspond to data elements, that is the information that is processed in the workflow. Each node has a value assigned to it, which typically differs between process instances (cases). In Figure 1, we present the PDM for a classical mortgage example, which will also be used in the comparison section. The final product of the process is to determine the value of the root (or top or end product) node. Values are determined from the bottom to the root as specified by the arcs (graph edges), which are called operations. These arcs represent actions that are applied on the valued data elements to produce values for nodes downstream. Each operation

[1] https://github.com/kmvarvou/pdm_heuristics



Description of elements

A	Maximum mortgage
B	Percentage of interest
C	Annual budget to be spent on the mortgage
D	Term of mortgage
E	Previous offer (within the period of validity of the offer)
F	Percentage of income to be spent on the mortgage
G	Gross income per year
H	Credit registration

	Output	Input	Cost	Time	Probability	Conditions
Op01	A	B, C, D	5.0	1.0	0.05	-
Op02	C	F, G, H	5.0	4.0	0.05	-
Op03	A	H	9.0	3.0	0.05	Value(H) = "negative"
Op04	A	E	2.0	2.0	0.00	-
Op05	B	-	0.0	0.0	0.00	-
Op06	D	-	0.0	0.0	0.00	-
Op07	E	-	1.0	1.0	0.50	-
Op08	F	-	0.0	0.0	0.00	-
Op09	G	-	0.0	2.0	0.00	-
Op10	H	-	3.0	10.0	0.15	-

Operations and their attributes

Fig. 1. The PDM mortgage example from [19]

can have zero or more input data elements while producing the value for exactly one output data element. An operation is represented by a tuple, which consists of the output element and a set of input elements, e.g. $(A, \{B, C, D\})$ for $Op01$ in the figure, which means that $Op01$ can be applied only if B, C and D have been produced and *may* lead to the generation of the value of A. A data element may be determined through multiple operations, e.g., in the figure, A can be determined in three manners and, for a specific case, the process terminates as soon as one of them manages to complete. In the case where an element has zero input elements, it is called a leaf element, and commonly, it is provided as input to the process; in the figure, elements such as B, F and E are leaf elements.

In summary, the PDM describes the operations that can be performed to produce the top element, along with their inter-dependencies. Not all operations need to be executed. To allow for cost-based decisions, each operation has the following metadata (an example is in the table at the bottom-right of Figure 1):

- *Cost*, which represents the cost associated with executing the operation.
- *Time*, which represents the time required for the complete execution of the operation.
- *Probability*, represents the probability that an operation is executed unsuccessfully, therefore not producing its output element.
- *Conditions*, which represent requirements regarding the value of the input elements. These requirements must be met, if the process is to be executed, meaning that the existence of all input elements of an operation is not sufficient.

3 Deriving workflow designs

PDM does not specify per se how the end information product is created but allows multiple workflow designs to produce the desired information product. As mentioned above, there are cases where multiple workflow designs lead to the production of an output element. Usually, in these cases, the alternative paths have different execution costs and time durations. This gives rise to the following optimization problem: *which paths of operations to choose for a specific case in order to optimize given quantitative objectives of cost and time?*

There are two high-level strategies for the calculation of an optimal execution path of a workflow, namely a global and a local one. A global strategy considers the effect of each decision on future steps. It takes into account the complete set of alternative paths that produce the end product to optimize the execution performance of each case. Instead, a local strategy adopts a step-by-step approach, meaning that, at each step, it examines the set of operations available for execution and chooses the best one, according to a particular metric, e.g. cost of execution. As explained in [19], a global strategy does not scale. For this reason, in this work, we exclusively deal with low-polynomial local strategy heuristics.

More specifically, the local strategies used by the PBWS method in [19] comprise the following heuristics (the last one is not explicitly mentioned in [19] but it is trivial to include it):

1. *Random*: the operation is randomly selected from the set of executable operations.
2. *Lowest Cost*: the operation with the lowest cost is selected.
3. *Shortest Time*: the operation with the shortest time is selected.
4. *Lowest Failure Probability*: the operation with the lowest probability of not being executed successfully is selected.
5. *Shortest Distance to Root Element*: the operation with the shortest distance to the root element (measured in the total number of operations) is selected.
6. *Shortest Remaining Process Time*: the operation with the shortest remaining processing time (measured as the sum of the processing times of the operations on the path to the root element) is selected.
7. *Shortest Remaining Cost*: the operation with the shortest remaining cost (measured as the sum of the costs of the operations on the path to the root element) is selected.

We discuss implementation details at the end of this section.

3.1 Rank-based Heuristics

Our approach relies on treating productions rules in a manner that resembles knock-out activities, and their optimal ordering, which also bears similarities to the way data analytics operators and database joins are ordered [2, 10, 9]. A knock-out activity is an activity whose execution leads directly to the completion of the process. For example, the execution of $Op03$ in Figure 1, which produces

the root element A is a knock-out activity/production rule. Then, the optimal ordering needs to take into account the probability of an operation to produce the end element, either directly or indirectly as a sequence of operations starting with that operation, and the corresponding cost or execution duration.

More specifically, at each step, we consider all the operations ready for execution (i.e., those with all inputs present) and we choose the one with the highest *rank* value. The rank value of an operation Op is defined as follows:

$$rank(Op) = \frac{\prod_{Op' \in \pi(Op)} 1 - Probability(Op')}{\sum_{Op' \in \pi(Op)} Cost(Op')}$$

where $\pi(Op)$ is the path from Op (including) to the root.

Example: in the example in Figure 1, assume that in the current state all leaf elements have been produced already except element E, for which $Op07$ was not executed successfully. Thus, in the next step, there are two available production rules for execution, namely $Op02$ and $Op03$. Based on their attributes they have the following ranking value: $rank(Op02) = 0.9025/10 = 0.09025$. While $Op02$ may not lead to a process termination directly, we consider $Op02$ as part of a path that leads indirectly to the root, that is the path: $Op02 \rightarrow Op01 \rightarrow A(end\ state)$. Therefore, we use as probability of this knock-out path, the probability of *success* of the operations in the whole path, which is $(1 - Probability(Op02)) * (1 - Probability(Op01)) = 0.95 * 0.95 = 0.9025$ and as cost, the aggregate cost of the whole path, which is $Cost(Op01) + Cost(Op02) = 5 + 5 = 10$. On the other hand, $rank(Op03) = 0.95/9 = 0.105556$. The probability 0.95, in the nominator, is the probability of the *successful* execution of $Op03$ because it is the successful execution of $Op03$ that produces the root element A and therefore, completes the workflow execution. Based on these values, $Op03$ is selected for execution. □

This heuristic comes in three flavors. The first one uses the formula above. The other two modify the denominator in the rank formula and employ (i) the sum of the operation duration times and (ii) the sum of the product of the cost and times, respectively. As such, they focus more on time duration and a combination of both metrics, respectively.

3.2 Implementation Issues

All the heuristics conform to a generic template, shown in Algorithm 1, which produces, for each case, the sequence of steps (operations) chosen; this sequence is captured in the variable WF. The operations metadata are mapped to a `HashMap` variable, where the key is the operation id and the value is nested and consists of all attributes in the table of the example in Figure 1. Based on such a structure, the *executableList* can be found through a simple traversal of the hashmap, taking into account the contents of the *availableList*. This occurs once at the beginning and then, *executableList* keeps being updated. Then the *availableList* is scanned to choose the *nextOp* operation according to the chosen local strategy (line 4). The time complexity of this algorithm, for the first 4 heuristics, is $O(n(n+V))$, where n is the number of operations and V the number

Algorithm 1 PDM heuristics template

Require: (1) Operations metadata as depicted in the table in Figure 1. (2) A set of the already produced data elements named "*availableList*"; if none this set is initialized to $\emptyset$.

```
1: WF ← {}
2: executableList ← operations that can be executed
3: while executableList ≠ ∅ and !availableList.contains(root_element) do
4:    nextOp ← select operation for execution from executableList
5:    WF ← {WF, nextOp}
6:    execute nextOp
7:    if execution is successful then
8:       add output elements of executed operation to availableList
9:       update executableList based on new available elements
10:   end if
11: end while
12: return WF
```

of nodes in the graph. This is because, in each step at most n operations are examined, and there are n steps at most. Also, for an operation to be inserted in the *executableList*, up to V elements checks need to be performed. A fast implementation employs a priority queue for supporting the choice of *nextOp* in each iteration, but it is beyond the scope of this work to discuss such details.

However, a more important point is that the three last existing heuristics along with our proposal need to process the path from a given operation to the root. For this, we need to employ another auxiliary structure, where the PDM model is seen as a typical graph with as many vertices as the data elements and directed edges for each data element in the input of another data element pointing to that element. Since finding the shortest path from a root element is at most $O(nlogV)$ using an algorithm such as Dijkstra, the complexity of the relevant techniques is the previous complexity multiplied by this factor, i.e., $O(n^2(n+V)logV)$. In addition to the PDM's data elements, this graph also contains an artificial starting vertex. This vertex represents the initial state of execution where no elements have been produced. It covers operations, such as $Op08$, $Op09$ and $Op10$ in the example, which, otherwise, cannot be represented as edges connecting vertices.

However, in the rank-based solutions more problems arise due to the product in the fraction nominator. In addition, we have to deal with the cases, where multiple paths from a data element to the root element exist, as is the norm. We distinguish between two cases. First, if there exists an operation that directly leads to the root element, we consider this edge as the complete path. Second, if such an operation does not exist, we cannot rely on shortest paths with edges either non weighted or weighted according to the cost or time as we do for calculating the denominator, which is calculated using Dijkstra's algorithm in a straightforward manner. Our procedure is as follows with regards to the nominator of the rank fraction. We use the probability of failure of data element

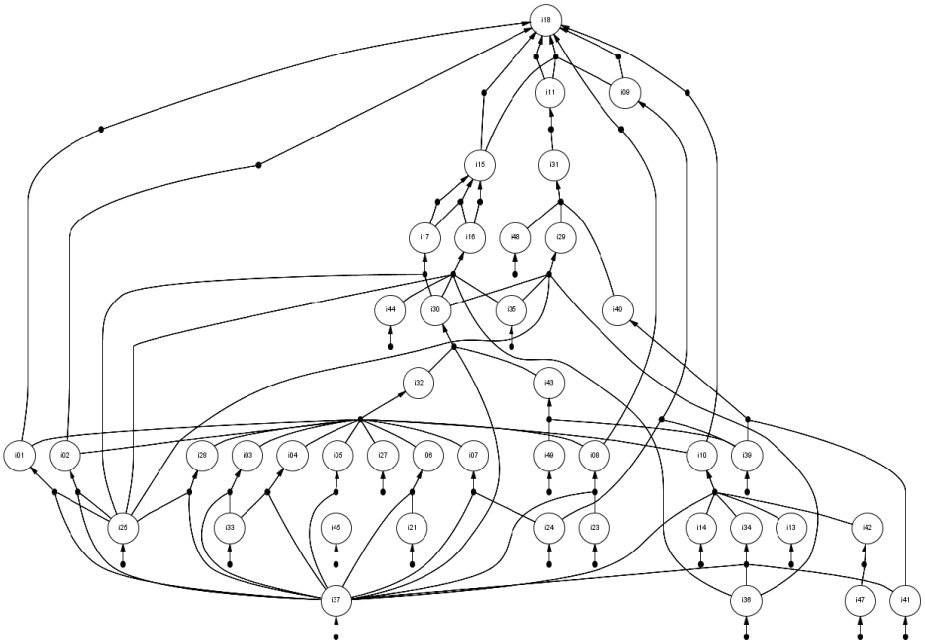


Fig. 2. The PDM social insurance example from [14]

production, exactly as provided in the metadata table to assign weights to the graph edges. Then, as the representative path of the operator Op, we choose the path for which the sum of these probabilities is the smallest one, i.e., we have again reduced the problem to a shortest path one. For this path, we compute the product of the success probability values; the success probability of each operation (or graph edge) is 1 minus the failure probability.

4 Evaluation

In this section, we evaluate our proposed approach using two PDMs. The first one is the mortage example that was presented in Section 2, while the second one, shown in Figure 2, represents a larger process from a social insurance company [14]. To evaluate the 7 existing heuristics and our 3 rank-based flavors, we created 100K random cases for each of the two PDMs. The cost and time attributes were assigned (integer) values in the [1,10] range, i.e., these attributes may differ up to an order of magnitude. The probability of failure was assigned values in the [0.0, 0.999] range, i.e., we consider the complete range from guaranteed success to almost certain failure. All distributions are uniform. We do not explicitly consider the probability of meeting conditions, since this probability can be seen as covered by the failure probability.

The results are summarized in Figures 3, 4, 5 and 6. Each figure corresponds to one of the four combinations of the two objectives of cost and execution time

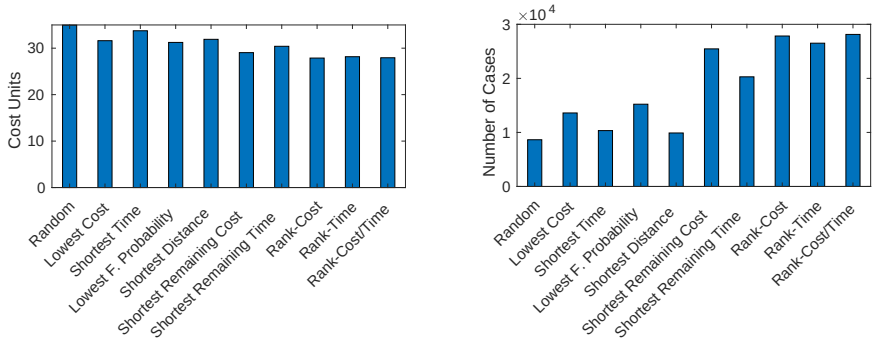


Fig. 3. The average cost per case (left) and number of cases where each heuristic achieved the best result (right) regarding the mortgage PDM.

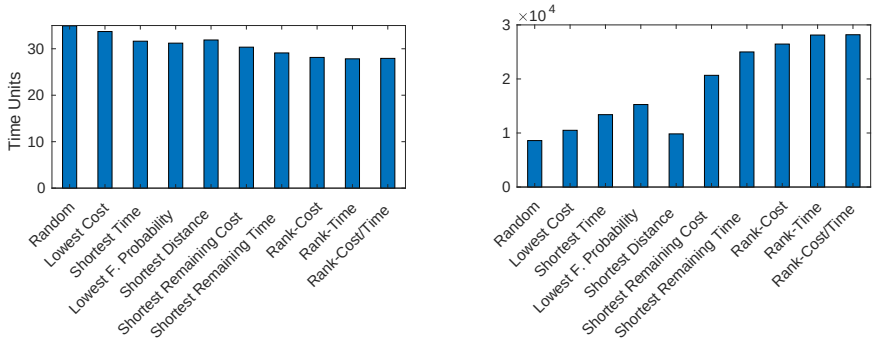


Fig. 4. The average execution time per case (left) and number of cases where each heuristic achieved the best result (right) regarding the mortgage PDM.

of the process and the two PDMs. We start our discussion with the first two result figures that refer to the smaller PDM of Figure 1. In all figures, the left barchart depicts average behavior, while the right one shows the number of cases where a heuristic exhibited the best performance for that specific case. The main observations are as follows:

1. The rank-based heuristics proposed in this work are the best performing ones both when cost and when time is the optimization objective. Their relative difference is small and does not exceed 1.1%, which means that the rank function effectively covers both objectives in all three flavors.
2. Choosing randomly the next operation incurs approximately 25% higher cost and 25% higher time compared to our solutions. The best performing heuristics from the existing ones are *Shortest Remaining Cost* for the cost objective and *Shortest Remaining Time* for the time objective. These heuristics are on average only 4.2% and 4.6% worse than the rank-based ones, respectively.

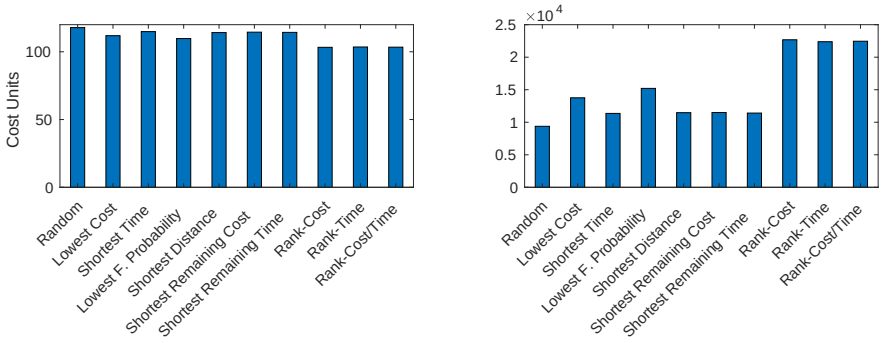


Fig. 5. The average cost per case (left) and number of cases where each heuristic achieved the best result (right) regarding the social insurance PDM.

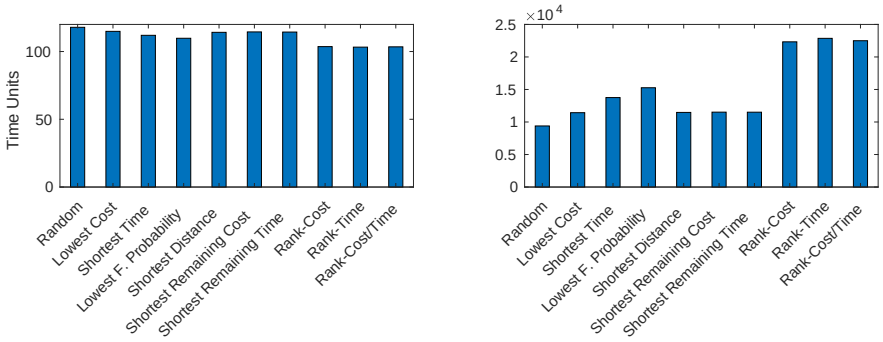


Fig. 6. The average execution time per case (left) and number of cases where each heuristic achieved the best result (right) regarding the social insurance PDM.

3. There is no globally dominant solution. As the two right barcharts show, even the random heuristic yields the best performance in some cases. On average, for each case there are 1.86 heuristics that yield the best performance regardless of the exact objective (it can be observed in the figures that the barcharts do not sum to 100K). The most common winners are the rank-based heuristics, but each individual flavor is the best in no more than 28.5% of the cases.

Next, we move our attention to Figures 5 and 6, which refer to the largest PDM in Figure 2. In this PDM, in summary, the rank-based heuristics are still the best ones in the average case, with even smaller relative differences between the three flavors (less than 0.4%). The random heuristic is only 14% worse than the best heuristic on the average case. But the best performing heuristic from the ones in [19] has now become the one that chooses the operation with the lowest failure probability; this heuristic is 6.3% worse that the rank-based solutions. Finally, in each case, 1.52 heuristics achieve the top performance on average.

Similar and even better results are observed when the optimization objective is the product of cost and time (no details are provided due to space limitations). Two additional significant points are: (i) Our proposals are superior to the best performing existing heuristic by up to more than 20 times. (ii) The overhead time to run the heuristics is extremely low: on a Ryzen 5 3600x CPU with 16GB RAM, our rank-based heuristics take less than 1.34 milliseconds for each case of the social insurance PDM; the other heuristics are even faster and the times are negligible for the small PDM.

5 Related Work

As stated in the introduction, an increasing amount of data-centric approaches have been developed as part of a general trend in the area of Business Process Management (BPM). Despite this recent interest, Business Process Improvement or Redesign, one of the key areas of BPM remains relatively undeveloped in terms of automated algorithmic solutions. In a recent survey that aims to evaluate several data-centric process approaches, this lack of focus on process optimization or redesign is highlighted [15]. Out of the 14 methods examined, only 2 of them identify the objective of business process optimization as a motive for their development. These two methods are Product Based Workflow Design (PBWD) [14] and its extension, Product Based Workflow Support (PBWS) [19], upon which we build our work.

A significant part of recent research in BPs targets variability between process models aiming at the same high-level objectives [16]. For example, the work in [16] is motivated by the fact that the same goal in different municipalities is performed using different equivalent processes and, to manage such variability, it introduces the configurable process trees. This methodology allows a specific set of process models to be selected according to several criteria. This bears some similarity to the way PBWS exploits the existence of alternative paths in order to optimize each case's performance. The main difference lies in the fact that these alternatives are different paths of the same, already existing PDM model, while in [16], there is an attempt to create a model that contains alternative paths to cover all rationales. In such a context, proposals like [5] deal with the problem of extracting alternative models, whereas the issue of assessing the quality of different process model configurations [4] has also been explored.

Additionally, there are proposals considering variant optimization objectives, such as the techniques in [1], where a set of heuristics is introduced for optimizing the metrics of resource utilization, maximal throughput and execution (cycle) time. These heuristics consider changing the relative ordering of activities, enforcing parallel execution and activity merging, but they cannot be applied to PDMs (at least in a straightforward manner). Finally, our work relates to declarative process models [13, 6]; e.g., our workflow design solution can be seen as a promising means to derive executable model structures out of such declarative models although providing a complete methodology to achieve this remains an open issue.

Regarding data-centric workflows, a lot of effort has been put towards finding the best sequential order of flow tasks for objectives such as minimization of the sum of the costs of these tasks or the bottleneck cost, or the maximization of the utilization of each execution processor, and so on [9, 3, 7]. All these proposals aim to optimize a single criterion, but there are also proposals that target multi-objective data flow optimization, such as the algorithms in [18, 17]. Despite some initial efforts in [10], transferring the results of data analytics workflow optimization to business process workflows is still a topic in its infancy.

6 Conclusions and Future Work

This work focuses on processes modelled according to the declarative PDM paradigm and aims to evaluate both existing and novel heuristics for yielding workflow designs on a case-by-case basis. Inspired by data analytics, we use the notion of rank, which combines the probability to produce the root element and the cost to achieve this in a single metric. In our experiments, we show that rank-based heuristics exhibit the best performance on average, but in specific cases, each of the 10 heuristics examined in this work may be the dominant one.

Our work suffers from the same limitations as the heuristics in [19]: we optimize on a case-by-case basis without seeing the process as a whole, e.g., in terms of resource utilization and without considering parallel task execution. Apart from addressing these limitations, we aim to follow three directions as future work: (i) to better handle the information that a data element may need input by multiple elements, when computing path costs (which is now implicitly ignored); (ii) to devise hybrid methodologies that switch between heuristics in a specific case, motivated by our key observation that there is no globally dominant solution; and (iii) to transfer similar techniques to other declarative modelling approaches, such as [13].

Acknowledgment. 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). We would like also to thank Dr. Georgia Kougka for her comments and help.

References

1. van der Aalst, W.M.P.: Re-engineering knock-out processes. Decis. Support Syst. **30**(4), 451–468 (2001)
2. van der Aalst, W.: Re-engineering knock-out processes. Decision Support Systems **30**(4), 451 – 468 (2001). https://doi.org/https://doi.org/10.1016/S0167-9236(00)00136-6
3. Agrawal, K., Benoit, A., Dufossé, F., Robert, Y.: Mapping filtering streaming applications. Algorithmica **62**(12), 258308 (2012)

4. Buijs, J., Dongen, van, B., Aalst, van der, W.: Discovering and navigating a collection of process models using multiple quality dimensions. In: Business Process Management Workshops (BPM 2013 International Workshops, Beijing, China, August 26, 2013, Revised Papers). pp. 3–14. Springer (2014)
5. Buijs, J.C.A.M., van Dongen, B.F., van der Aalst, W.M.P.: Mining configurable process models from collections of event logs. In: Daniel, F., Wang, J., Weber, B. (eds.) Business Process Management - 11th International Conference, BPM 2013, Beijing, China, August 26-30, 2013. Proceedings. Lecture Notes in Computer Science, vol. 8094, pp. 33–48
6. Chawla, N., King, I., Sperduti, A.: User-guided discovery of declarative process models (2011)
7. Deshpande, A., Hellerstein, L.: Parallel pipelined filter ordering with precedence constraints. ACM Trans. Algorithms **8**(4) (2012)
8. Henriques, R., Silva, A.R.: Object-centered process modeling: Principles to model data-intensive systems. In: Business Process Management Workshops - BPM 2010 International Workshops and Education Track, Hoboken, NJ, USA, September 13-15, 2010, Revised Selected Papers. pp. 683–694 (2010)
9. 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)
10. Kougka, G., Varvoutas, K., Gounaris, A., Vergidis, K., Tsakalidis, G.: On knowledge transfer from cost-based optimization of data-centric workflows to business process redesign. Trans. Large Scale Data Knowl. Centered Syst. **to appear** (2020)
11. Künzle, V., Reichert, M.: Philharmonicflows: towards a framework for object-aware process management. Journal of Software Maintenance **23**(4), 205–244 (2011)
12. Orlicky, J.A., Plossl, G.W., Wight, O.W.: Structuring the bill of material for mrp. Operations management: critical perspectives on business and management **58** (2003)
13. 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). pp. 287–300 (2007)
14. Reijers, H.A., Limam, S., van der Aalst, W.M.P.: Product-based workflow design. J. Manag. Inf. Syst. **20**(1), 229–262 (2003)
15. 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)
16. Schunselaar, D.: Configurable process trees : elicitation, analysis, and enactment. Ph.D. thesis, Department of Mathematics and Computer Science (10 2016), proefschrift
17. Simitsis, A., Wilkinson, K., Dayal, U., Castellanos, M.: Optimizing etl workflows for fault-tolerance. In: 2010 IEEE 26th International Conference on Data Engineering (ICDE 2010). pp. 385–396 (2010)
18. Simitsis, A., Wilkinson, K., Castellanos, M., Dayal, U.: Optimizing analytic data flows for multiple execution engines. In: Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data. p. 829840 (2012)
19. Vanderfeesten, I.T.P., Reijers, H.A., van der Aalst, W.M.P.: Product-based workflow support. Inf. Syst. **36**(2), 517–535 (2011)