Mapping DMN to PDM to Enable Optimizations

Konstantinos Varvoutas[1], Anastasios Gounaris[1] and Georgia Kougka[1]

[1]Department of Informatics, Aristotle University of Thessaloniki, Greece

Abstract

Decision modeling is a key aspect in modern BPM complementing and working alongside process models, such as BPMN. DMN is the main standard for decision modeling and in this work we aim to tackle a main drawback of DMNs, namely the lack of optimization techniques in terms of minimizing the execution time and execution cost. We address this limitation through mapping DMNs to PDMs. PDMs are declarative and emphasize on the data input requirements for performing operations to derive additional data elements, one of which is the final target of the process. Moreover, effective optimization heuristics have been proposed for PDMs. We also present two illustrative examples showing the benefits stemming from our approach.

Keywords

data-centric processes, decision modelling, process optimization, DMN, PDM

1. Introduction

Organizations are interested in executing their processes efficiently, in an attempt to remain competitive. To this end, they make use of *Business Process Management (BPM)*. BPM is defined as a body of methods, techniques and tools to discover, analyze, redesign, execute and monitor the business processes of an organization [1]. BPM does not focus only on the control flow of processes, e.g., through employing the Business Process Model and Notation (BPMN) standard. Given that most processes involve decisions of various kinds, there is an additional focus on the decision-making aspect of processes. To address this need, the *Decision modeling and Notation (DMN)* standard has been introduced. According to [2], DMN covers the decisions that are enacted through a flow of processes. It is a declarative approach, with the purpose of segregating the decision logic from business processes, as the decisions are separated from the other process information, such as the flow of tokens across activities.

Decisions in DMN are usually based on a number of input data, which can be the outcome of one or more other decisions upstream. As such, the decision inputs, especially of intermediate decision tables, are often not available at the beginning of the process; moreover, there is some cost and/or time overhead associated with their acquisition, e.g., a costly operation needs to be performed to obtain them. This gives rise to the problem of defining the optimal order of acquiring input data and checking the corresponding rules in terms of execution time and execution cost. As explained later, this problem generalizes operator and task ordering optimization, which is a key and persistent challenge in query processing and data-intensive workflows [3, 4].

The *Product Data Model* is a data-centric approach, tailored to information-intensive processes, which are used by an array of industries, including insurance companies[5, 1], banking[6] and manufacturing[7]. It places emphasis on modeling the requirements for the production of its output product rather than on the exact way of producing it. To this end, it is accompanied by a set of decision strategies, e.g., through a method referred to as *Product Based Workflow Support (PBWS)* [5]. PBWS features a set of decision strategies that aim to produce the end-product step-by-step, in a cost efficient manner and is recently extended by data management-inspired optimization strategies [8]. More specifically, the work in [8] shows how the rationale of defining the order of joins in database queries and the tasks in data-intensive workflows under arbitrary precedence constraints can be transferred to optimizing PDM execution.

Due to the similarities between the two standards, namely DMN and PDM, in terms of structure and use-cases, PDM could be used to represent the decision logic of processes, originally captured in DMN. Such a conversion would make DMN models compatible with the dynamic optimization techniques that have been developed for PDMs. To this end, we propose an approach to converting a DMN model into a PDM one. Our approach produces a PDM given the Decision Requirement Diagram (DRD) graph and decision table of a process; the resulting PDM is then amenable to optimizations.

In summary, our contribution is twofold: (i) the introduction of an approach to map DMN decisions and input data to a PDM and (ii) the provision of example cases to show the benefits that can stem from optimizing the order in which DMN decisions are taken leveraging the same techniques that optimize the operation ordering in PDMs.

The remainder of this paper is structured as follows.

BICOD 2021: The British International Conference on Databases 2021, December 09–10, 2021, London, UK

kmvarvou@csd.auth.gr (K. Varvoutas); gounaria@csd.auth.gr (A. Gounaris); georkoug@csd.auth.gr (G. Kougka)

Section 2 provides the background and the motivating examples. Next, we introduce our approach followed by discussion of examples. We discuss the additional related work in Sec. 5 and we conclude in Sec. 6.

2. Background and main rationale

2.1. Decision Model and Notation (DMN) basics

The DMN standard complements control flow-oriented initiatives, such as BPMN. It is a declarative approach that models decisions on two levels, the *requirements* level and the *decision logic* level. On the first level, the information requirements of the decisions are represented by a *decision requirements diagram (DRD).* These diagrams consist of four types of elements: *decisions, input data, business knowledge models and knowledge sources.* The decision logic is most commonly expressed with the help of decision tables. In a nutshell, the role of a DMN decision model is to contribute to the business process execution and provide the requirements that must be preserved taking into account the input data [9]. Figure 1 presents the DRD graph of an example claim assignment DMN process, where the decisions nodes are *Determine Employee*, *Employee appropriateness score* and *Experience of people*, while the input data are represented by the DRD nodes *Region of customer and employee*, *Claims expenditure*, *Number of open claims of employee* and *Approval authority.* In this work, we focus on the case where the whole decision process is captured by the DMN model. As such, in our approach, it suffices to focus on decisions and input data solely.

On the second level, decision tables are used to represent decisions (nodes) that have been specified in the DRD in detail; an example is presented in Figure 2. Each row of a decision table represents a rule, which specifies how a decision is taken. The top-left cell specifies the *hit policy.* This policy defines whether one or all rules need to be triggered, whether rules need to be checked in a specified order and so on.

2.2. Optimization in DMN: a motivation example

In many cases, the execution of specific decisions can be optimized for a specific decision model. There are cases where the decision inputs can be available with a specific cost and the need arises to define the most cost efficient execution plan for specific input values. Additionally, there are multiple process models that can be generated based on a decision model, which implies the demanding need to introduce a method to *"evaluate"* the process models for selecting the best business strategies [9].

In Figure 2, we present the decision table of the claim assignment example. Without explaining the full details at this stage, the outcome of a decision is the rightmost column and depends upon a set of input values, which are represented in the blue-colored fields. Often, these decision inputs are not available upfront, while acquiring each one of them incurs a certain cost. Therefore, aside from finding the correct outcome for a specific set of values, a cost-efficiency issue is encountered. The research question that motivates this work is: *what is the optimal order of acquiring the input values and executing the sub-decisions (i.e., rules) of a decision model?*

Let us assume for the moment that the hit policy in Figure 2 is "U", which means that only one rule may match as the rule overlapping may lead to an error and the costs of acquiring the four input fields are 3, 2, 2 and 5 cost units, respectively. Given these costs, it seems that acquiring the first input regarding the region of the employee and the customer with cost 3 is the most efficient choice. But what if, with probability 30%, when acquiring the first input, its value is "no", which implies that no rule in the decision table can be triggered and additional rules need to be checked? To make the case even more complex, we also need to account for the fact that checking each rule (i.e., each row in the table) comes with a cost that potentially differs between rows. To complete the example, if we assume that in most of the cases the experience of the employee is high, the open claims are always more than 10 and each rule has the same cost to execute, then, in the average case, it is beneficial first to extract the number of the open cases and then define the final outcome (i.e., the score) based on the retrieved value. This is despite the fact that extracting the number of open cases comes with the highest cost of input acquiring, i.e., 5 cost units.

In the example above, we essentially had to determine the most beneficial ordering of operations. However, this problem has been investigated in depth in works such as [5, 8], when the model is PDM rather than DMN. Therefore, our main contribution is to capitalize on the optimization techniques for PDMs and apply them to DMN models after translating them.

2.3. Product Data Model

According to [1], the Product Data Model (PDM) is the key mechanism to define the process structure of an informational product, where example informational products include a decision as to whether to grant approval to a specific admission request, approval of a mortgage application, and so on. PDM is used to represent the structure of a workflow product in a rooted graph-like manner. One of the most important phases during the

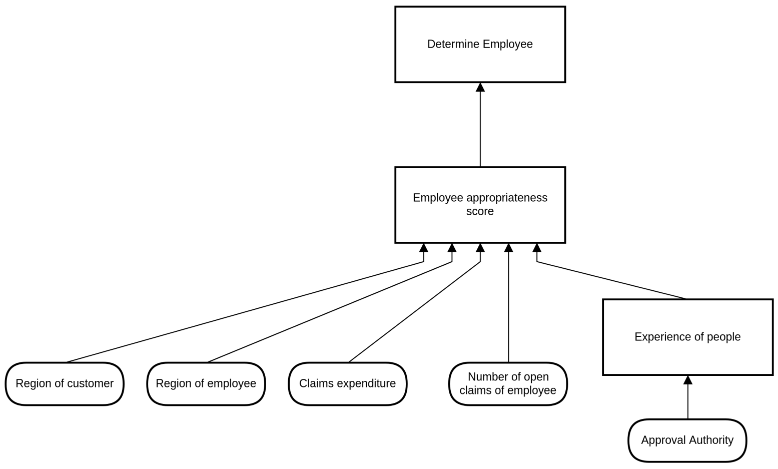
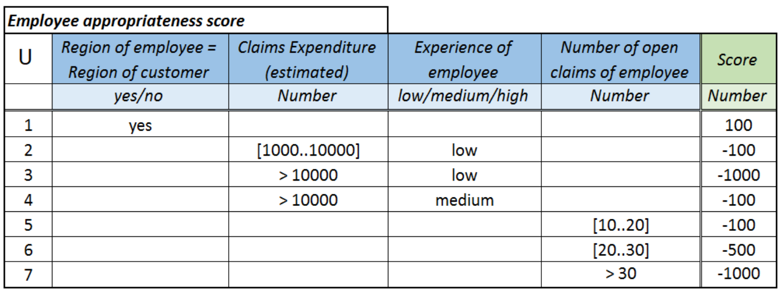


Figure 1: The DRD graph for the claim assignment example (taken from [10])



Employee appropriateness score

U	*Region of employee = Region of customer*	*Claims Expenditure (estimated)*	*Experience of employee*	*Number of open claims of employee*	*Score*
	yes/no	*Number*	*low/medium/high*	*Number*	*Number*
1	yes				100
2		[1000..10000]	low		-100
3		> 10000	low		-1000
4		> 10000	medium		-100
5				[10..20]	-100
6				[20..30]	-500
7				> 30	-1000

Figure 2: The decision table for the *Employee appropriateness score* decision of the claim assignment example (adapted from [10])

PDM design is the analysis, where the information elements, dependencies and production logic are identified [1]. PDMs describe the required elements for generating the end product of a workflow and a PDM mainly consists of connected data elements (nodes), which represent the information that is processed in the workflow. The data element values are produced by executing *operations*, which are represented by the graph edges on the data elements. Each operation requires a set of input data elements and produces exactly one output data element. Additionally, the dependencies between the PDM elements define which data are required to produce other data.

A PDM does not specify per se how the end product (i.e., the root element) is produced, but allows multiple sequences of operations to derive the root information product. In other words, there may be multiple paths to the production of the root element. Usually, each of these paths has a different cost of execution, giving 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?* To answer this question, which relates to the execution of PDMs, the PDM is accompanied by a set of decision strategies in a method referred to as Product Based Workflow Support (PBWS) [5]. PBWS entails both local and global decision strategies, however, in this work we will focus only on local decision strategies and benefit from improvements over [5], as these are presented in [8]. 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. Such cost-based decisions are enabled because PDM operations are typically annotated with quantitative metadata regarding the cost and duration of their execution and the probability of failure to produce a data element.

Note that optimizing PDM execution is at least as difficult as detecting the optimal order of tasks in data-intensive workflows under arbitrary constraints; it is beyond the scope of this work to provide more details, but this is due to the fact that task ordering in workflows can be mapped to a PDM, in which each element can be produced via a single operation. As explained in [3], based on the analysis of [11], the task ordering problem in workflows is not only NP-hard but it is unlikely any polynomial algorithm to manage to approximate the optimal solution within a polynomial factor.

2.4. Decision Strategies in DMNs

DMN and PDM models share some inherent similarities with regards to their structure and use-cases. Both of them are used to represent information-intensive processes, centered around some type of decision making. Similarities are also apparent when it comes to their execution. Both of these standards feature models that can be directly executed. In such cases, the execution is approached in a dynamic step-by-step manner, which aims to produce the final outcome optimizing a particular objective, i.e. cost of execution. However, while, for PDMs, there is a set of decision strategies available to solve this problem, this is not the case for DMNs. Therefore, by providing an approach that converts DMN models into PDM, we also render the existing decision strategies that are available for PDMs applicable to DMNs.

Algorithm 1 Mapping DMNs to PDMs

```
Require: (1) DRD graph of the DMN as depicted in figure 1.
    (2) Decision tables of the DMN as depicted in figures 2
    and 3.
 1: for every element $e$ in DRD do
 2:     if $e$ is the top element of DRD then
 3:         make $e$ root element of PDM
 4:     else if $e$ is an input data element of DRD then
 5:         make $e$ leaf element of PDM
 6:     else
 7:         make $e$ regular element of PDM
 8:     end if
 9: end for
10: for every decision node $e$ in DRD with a decision table
    do
11:     convert decision logic of table into PDM operations.
12: end for
13: for every decision node $e$ in DRD without a decision table
    do
14:     extract decision logic from DRD.
15: end for
16: for every input data element $e$ in DRD do
17:     insert a corresponding leaf operation in the PDM.
18: end for
19: return $PDM$
```

3. Mapping DMNs to PDMs

In this section, we present our approach for converting a DMN model into a PDM. Our approach takes as input both the DRD graph and the decision table of a DMN model. We present a high-level algorithmic outline of our approach in Algorithm 1, while also presenting a more detailed outline of each step below.

The first step relates to the graph structure of the model. Each element of the DRD graph, either a decision or an input node, is mapped to a PDM node. The top element of the DRD is set as the root element of the PDM, while DMN input data elements are mapped to leaf data elements.

The next step is to extract the decision logic from the decision tables, when such tables are available, and the DRD, and convert this logic into the format of PDM operations. Each decision table corresponds to a different decision node, and therefore to a different PDM node based on the output variable in the table. Each row of a decision table corresponds to a PDM operation, but a single PDM operation may cover multiple decision table rows. Each of these operations produces as output the same output node as defined in the decision table. The input elements of each operation are determined based on the values of their respective column. For example, we present the decision table of the *Employee appropriateness score* decision node in Figure 2. The second row represents an operation that takes as input the nodes *Claims Expenditure* and *Experience of employee* (this is because empty columns signify that the respective input data do not play any role in this particular rule). In the corresponding PDM, there would be an operation connecting these two inputs to the output node, where the output node would correspond to the appropriateness score data element.

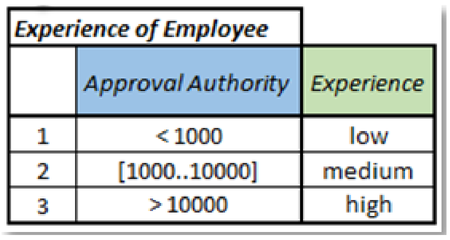
In addition, when a decision node does not have a corresponding decision table, the decision logic used for its production is captured by the DRD exclusively. In such a case, we take into account the incoming edges of the decision node. The decision logic then is transformed into an operation that produces the decision node as output and takes as input the nodes which are connected to it (via the DRD edges).

Finally, for each input data node in the DRD, a leaf operation (i.e operation with no input elements) is added to account for the production (acquisition) of its corresponding data element in the PDM.

This process is summarized in Algorithm 1 and is polynomial in the size of DRD and the entries of DMN decision tables.

4. Examples

In this section, we present two DMN models that will be used as examples to showcase our approach. The first DMN model has already been introduced in Section 2.1 and its DRD graph is presented in Figure 1. It repre-



Experience of Employee		
	Approval Authority	Experience
1	< 1000	low
2	[1000..10000]	medium
3	> 10000	high

Figure 3: The decision table for the *Experience of Employee* decision.

sents a claim assignment process and the DRD consists of three decision nodes and five input data nodes. The second DMN model represents a complex decision management problem of tax regulations [12].

4.1. Mapping to PDM and optimization of execution

4.1.1. First example.

Figure 2 presents the decision table of the *Employee appropriateness score* decision node as already explained. In addition, Figure 3 presents the decision table of the *Experience of Employee* decision node.

The resulting PDM model after applying our approach is presented in Figure 4. It contains eight data elements and ten operations, presented in detail in the upper and lower tables of Figure 4. The operation costs in the bottom table in the figure are assumed to be either extracted from logs or provided by a domain expert.

Regarding the operations of the PDM, we take into account both the decision table and the DRD of the DMN. The three operations *Op05, Op06* and *Op07* that produce data element *i5*, which corresponds the decision of the *Employee appropriateness score*, are derived from the decision table presented in Figure 2. The first row of the table corresponds to an operation that takes as input data elements *i1* and *i2*. Rows 2 to 4 are bundled (i.e., knotted) together into a single operation that takes as input elements *i3* and *i7*. Finally, rows 5 to 7 are bundled together into a single operation that takes as input element *i4*. The operation *Op08* that produces element *i7*, which represents the decision node *Experience of Employee*, is derived from the corresponding decision table in Figure 3. The table's three rows are reduced into a single operation that takes as input element *i6*. Decision node *Determine Employee*, which is represented by data element *i8*, does not have a corresponding decision table. Therefore, the operation that produces this element is derived from the DRD graph exclusively, taking into account the incoming edges of the node. The remaining operations relate to the production of either the leaf data elements of the PDM (*Op01-Op04, Op09*) or the root element out of the appropriateness score (*Op010*). The rationale above is not specific to this specific example but generalizes to any case.

What if the hit policy was different? In Table 2, the hit policy denoted by the leftmost cell in the top row is "U" meaning that any of the rules triggered can yield the output element in its own right. In the original version of this table, the hit policy was "C" meaning that all rules need to be evaluated and contribute to the final outcome. This case corresponds to a simpler PDM, where, instead of *Op05-Op06*, there is a single operation knotting $i1 - i4$ and *i7* to produce *i5*. Finally, we do not target "P (priority)" policies, since they strictly define the execution order thus leaving no space for optimization through choosing among alternatives.

4.1.2. Second example.

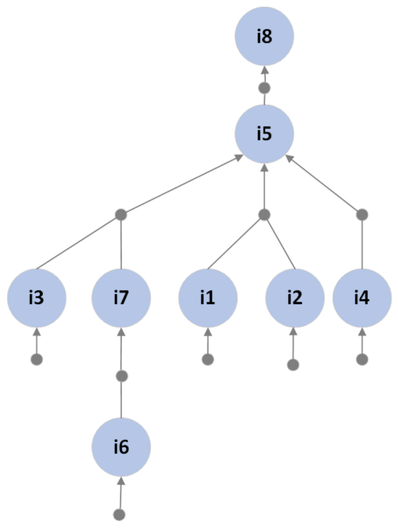
Regarding the tax management example from [12], the DRD graph is presented in Figure 5 and it consists of eight decision nodes and a single input data node. Figure 6 presents the decision table of the *Income Tax* decision node, which constitutes the outcome (top-decision) of the model. As previously we assume that it suffices a single rule to be triggered for the final decision to be taken. The resulting PDM model is presented in Figure 7. It contains nine data elements and thirteen operations.

Op01 aims to produce *i9*, which, through *Op02-Op04* and *Op06,Op07* produces the data elements *i6-i8* and *i3,i4*, respectively. *Op05* derives *i2* through combining *i6,i7*. Similarly, *Op8* produces *i5* through combining *i8,i9*.

The five remaining operations all produce data element *i1*, which represents the decision node *Income Tax*. These operations are derived from the decision table in Figure 6. More specifically, we group the rows that share the same attributes in the table into common operations. Therefore, the first two table rows yield *Op13*, which takes *i2* as inputs and produces *i1*. The third, fourth and last row of the table correspond to three distinct operations, namely, *Op09, Op10* and *Op11*, respectively. Finally, the fifth and sixth table rows are mapped to *Op12*.

4.2. Optimization of the execution

The main motivation behind mapping DMNs to PDMs is to enable optimizations, which refer to the ordering in which operations are executed and thus data elements are produced. In this section, as a proof of concept, we execute the two example PDMs discussed above. The optimized execution is based on the heuristic decision strategies discussed in *PBWS* [5] and [8]; the latter work introduced the notion of *rank* to prioritize operations inspired by optimization techniques in database queries and data-intensive analytics. Here, we demonstrate the applicability of such heuristics, which act as decision



Description of elements

i1	Region of Employee
i2	Region of Customer
i3	Claims Expenditure
i4	Number of open claims of employee
i5	Employee appropriateness score
i6	Approval authority
i7	Experience of Employee
i8	Determine Employee

	Output	Input	Cost
Op01	i1	-	5.0
Op02	i2	-	4.0
Op03	i3	-	5.0
Op04	i4	-	5.0
Op05	i5	i1, i2	5.0
Op06	i5	i3, i7	4.0
Op07	i5	i4	8.0
Op08	i7	i6	8.0
Op09	i6	-	6.0
Op10	i8	i5	7.0

Description of operations

Figure 4: The product data model for the claim assignment example

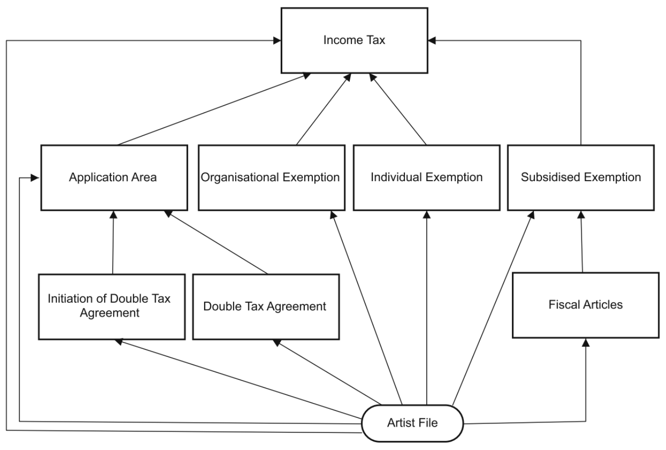
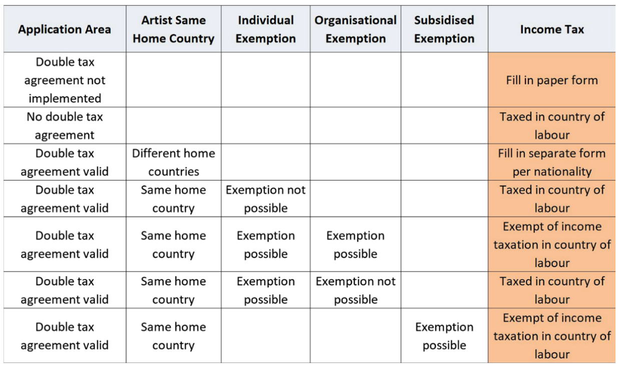


Figure 5: The DRD graph for the tax management example from [12]

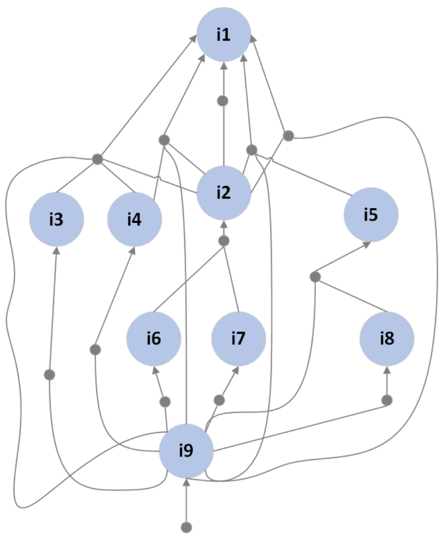
strategies in a case-by-case manner; i.e., they may derive a different operation ordering even for the same PDM but different instances. mentioned in [9], which assumes that not all input of the process is available upfront. Each input element is produced by an operation, which has a cost of execution. The cost of execution is used as a criterion to assess the different operations, using the aforementioned decision strategies. In addition to the cost, we could use other quantitative attributes, such as operation time duration and failure probability. Such attributes are omitted here for simplicity, but the interested reader can refer to [5, 8], whereas some summary information is provided in the appendix.

To demonstrate the impact on the total cost of the decision strategies and the potential for optimization, we present a detailed step-by-step execution instance of the claim assignment example. We assume a PDM case, for which the cost metadata are presented in the lower table of Figure 4. We employ three heuristics, i.e., three different decision strategies to choose the next operation to execute. These are (i) *Random*, which makes a random choice between all operations for which the input is ready, (ii) *Lowest Cost*, which chooses the next available operation with the lowest cost and *Ranked-Cost*, which



Application Area	Artist Same Home Country	Individual Exemption	Organisational Exemption	Subsidised Exemption	Income Tax
Double tax agreement not implemented					Fill in paper form
No double tax agreement					Taxed in country of labour
Double tax agreement valid	Different home countries				Fill in separate form per nationality
Double tax agreement valid	Same home country	Exemption not possible			Taxed in country of labour
Double tax agreement valid	Same home country	Exemption possible	Exemption possible		Exempt of income taxation in country of labour
Double tax agreement valid	Same home country	Exemption possible	Exemption not possible		Taxed in country of labour
Double tax agreement valid	Same home country			Exemption possible	Exempt of income taxation in country of labour

Figure 6: The decision table for the *Income Tax* decision of the tax management example.



Description of elements

i1	Income Tax
i2	Application Area
i3	Organizational Exemption
i4	Individual Exemption
i5	Subsidized Exemption
i6	Initiation of Double Tax
i7	Double Tax Agreement
i8	Fiscal Articles
i9	Artist File - Artist Same Home Country

	Output	Input
Op01	i9	-
Op02	i6	i9
Op03	i7	i9
Op04	i8	i9
Op05	i2	i6, i7
Op06	i3	i9
Op07	i4	i9
Op08	i5	i8, i9
Op09	i1	i2, i9
Op10	i1	i2, i4, i9
Op11	i1	i2, i5, i9
Op12	i1	i2, i3, i4, i9
Op13	i1	i2

Description of operations

Figure 7: The product data model for the tax management example

is detailed in [8] and takes into account the full path from an operation to the production of the root element without performing an exhaustive search of alternatives. The execution paths (i.e., operation orderings) of the three strategies are presented below:

1. *Random*: $Op_{02}, Op_{03}, Op_{04}, Op_{01}, Op_{07}, Op_{05}, Op_{09}, Op_{10}$. Total Cost: 45
2. *Lowest Cost*: $Op_{02}, Op_{01}, Op_{03}, Op_{04}, Op_{05}, Op_{09}, Op_{10}$. Total Cost: 37
3. *Ranked-Cost*: $Op_{02}, Op_{01}, Op_{05}, Op_{10}$. Total Cost: 21

From the example above, we see that making informed decisions can yield cost reduction by a factor more than 2X even in simple cases. Mapping DMNs to PDMs allows us to benefit from the existing state-of-the-art optimizations regarding operation ordering in PDMs. These optimizations include more heuristics, but based on the evidence in [8], on average, the rank-based ones are the better performing ones.

We now go a step further and we conduct a bigger experiment, where we simulate 10,000 random cases, where each operation is randomly assigned a cost value in the range of [0,10] following a uniform distribution. Figure 8 presents the results of the execution in terms of average cost of execution per case. The left chart corresponds to the claim assignment example, while the right chart corresponds to the tax management example. In the former

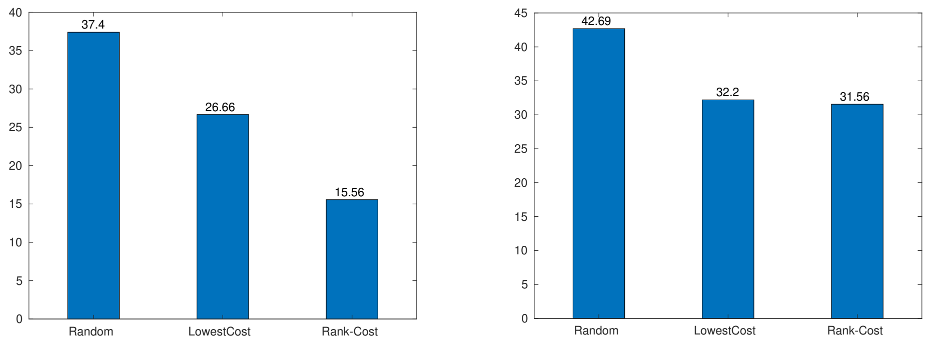


Figure 8: The average cost per case that each strategy achieved for the claim assignment (left) and tax management (right) PDMs respectively.

case, a random choice after each operation execution incurs 2.4X higher cost compared to the rank-based heuristic in the average case; the difference shrinks in the second example but is still significant (> 35%). It should be noted that such a performance gain is in line with the results of former optimization use cases of PDM found in industry [13], and are significant enough to yield tangible benefits [1].

5. Related Work

In the area of decision modeling, a lot of attention has been placed on the *separation of (decision) logic* from business processes. The work in [14] presents a formalization of decision requirements with the aim of achieving integration between a decision and process model. In [15], an approach that extracts the decision logic from an existing BPMN process model and obtains the corresponding DMN model is presented. In a similar context, the work in [16] presents a DMN-based approach that proposes to separate consideration of decisions and processes. An automated approach to mining decision rules from event logs has also been proposed in [17].

Additionally, there is also research that focuses on decision making with the aim of achieving process improvement. The work in [18] presents an approach that relies on data from past executions to make dynamic decisions related to resource assignment and process utilities. We differ in that we make dynamic decisions regarding the order of acquiring input data and execute rules. Finally, our work relates to the proposal in [19], which discusses an automated methodology that derives a BPMN/DMN model from an input PDM workflow. However, we focus on the reverse mapping from DMN to PDM.

6. Summary

In this work, we discuss mapping DMN decisions and input data to a PDM model, the execution of which can be optimized in a cost-based manner. Apart from showing the connection between different decision models, we enable optimizations regarding the ordering in which data elements are produced and rules leading to the final decision are fired. Through illustrative examples, we show that the cost gains can be significant; thus mapping DMNs to PDMs amenable to optimizations aspires to open new directions in the manner business processes are optimized. In the future, we aim to explore these directions in more depth, covering also BPMN models.

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). We would also like to thank Prof. Hajo Reijers for his insightful comments when discussing the main idea of this paper.

References

[1] M. Dumas, M. L. Rosa, J. Mendling, H. A. Reijers, Fundamentals of Business Process Management, Second Edition, Springer, 2018.

[2] OMG: Decision model and notation (DMN), 2015. URL: https://www.omg.org/dmn/.

[3] G. Kougka, A. Gounaris, A. Simitsis, The many faces of data-centric workflow optimization: a survey, Int. J. Data Sci. Anal. 6 (2018) 81–107.

[4] A. Rheinländer, U. Leser, G. Graefe, Optimization of complex dataflows with user-defined functions, ACM Comput. Surv. 50 (2017) 38:1–38:39.

[5] I. T. P. Vanderfeesten, H. A. Reijers, W. M. P. van der Aalst, Product-based workflow support, Inf. Syst. 36 (2011) 517–535.

[6] H. Reijers, S. Mansar, W. Aalst, Product- based workflow design., J. of Management Information Systems 20 (2003) 229–262.

[7] H. van der Aa, H. A. Reijers, I. T. P. Vanderfeesten, Designing like a pro: The automated composition of workflow activities, Comput. Ind. 75 (2016) 162–177.

[8] K. Varvoutas, A. Gounaris, Evaluation of heuristics for product data models, in: Business Process Management BPM Workshops, volume 397, 2020, pp. 355–366.

[9] L. Janssens, J. D. Smedt, J. Vanthienen, Modeling and enacting enterprise decisions, in: Advanced Information Systems Engineering Workshops - CAiSE 2016 Int. Workshops, volume 249, 2016, pp. 169–180.

[10] B. Rücker, Decision model and notation (dmn) – the new business rules standard. an introduction by example, 2015. URL: http://www.bpm-guide.de/2015/07/20/dmn-decision-model-and-notation-introduction-by-example/.

[11] J. Burge, K. Munagala, U. Srivastava, Ordering Pipelined Query Operators with Precedence Constraints, Technical Report 2005-40, Stanford InfoLab, 2005.

[12] F. Hasic, J. Vanthienen, From decision knowledge to e-government expert systems: the case of income taxation for foreign artists in belgium, Knowl. Inf. Syst. 62 (2020) 2011–2028.

[13] H. Reijers, Design and Control of Workflow Processes: Business Process Management for the Service Industry, volume 2617, 2003. doi:10.1007/3-540-36615-6.

[14] L. Janssens, E. Bazhenova, J. D. Smedt, J. Vanthienen, M. Denecker, Consistent integration of decision (DMN) and process (BPMN) models, in: Proc. of the CAiSE'16 Forum, volume 1612, 2016, pp. 121–128.

[15] K. Batoulis, A. Meyer, E. Bazhenova, G. Decker, M. Weske, Extracting decision logic from process models, in: Advanced Information Systems Engineering - 27th Int. Conf., CAiSE 2015, volume 9097, 2015, pp. 349–366.

[16] R. Song, J. Vanthienen, W. Cui, Y. Wang, L. Huang, A dmn-based method for context-aware business process modeling towards process variability, in: Business Information Systems - BIS, volume 353, 2019, pp. 176–188.

[17] J. D. Smedt, F. Hasic, S. K. L. M. vanden Broucke, J. Vanthienen, Holistic discovery of decision models from process execution data, Knowl. Based Syst. 183 (2019).

[18] K. Batoulis, A. Baumgraß, N. Herzberg, M. Weske, Enabling dynamic decision making in business processes with DMN, in: Business Process Management BPM Workshops, volume 256, 2015, pp. 418–431.

[19] H. van der Aa, H. Leopold, K. Batoulis, M. Weske, H. A. Reijers, Integrated process and decision modeling for data-driven processes, in: Business Process Management BPM Workshops, volume 256, 2015, pp. 405–417.

A. PDM Optimization Techniques

In this appendix, we provide some background information about the heuristic decision strategies [5, 8] for PDMs that were used in Section 4.2 of our work. The first two strategies, *Random* and *Lowest Cost* are presented in [5] as part of the *PBWS* methodology. The third strategy, referred to as *Rank-Cost*, relies on an approach that treats the operations of a PDM as knockout activities and their optimal ordering is similar to the ordering of data analytics operators ad database joins [8].

An activity is traditionally classified as knockout when its execution leads directly to the completion of its process. This approach takes into account the probability that an activity (i.e., a PDM operation) produces the root element, either directly or indirectly. In the latter case, it is treated as a sequence of operations, starting from that particular operation. More specifically, it relies on a *rank* function to select the next operation for execution. The rank value of an operation *Op* is defined as follows:

$$rank(Op_i) = \frac{\prod_{Op' \in \pi(Op_i)} 1 - prob_{Op'}}{\sum_{Op' \in \pi(Op_i)} Cost_{Op'}} \tag{1}$$

where $\pi(Op_i)$ is the path from Op_i (including) to an operation directly producing the root. It should be noted that, in the context of this work, we considered PDM operations to be always successful, therefore the numerator is equal to 1 for every operation.

Example: In the PDM in Figure 4, let us assume a state where leaf elements *i4* and *i6* have already been produced. Therefore, operations Op_{07} and Op_{08} are available for execution and a choice must be made for either of them to be executed. The path that that corresponds to Op_{07} is: Op_{07}, Op_{10}, which has an aggregate cost of: $Cost_{Op_{07}} + Cost_{Op_{10}} = 8 + 7 = 15$. While, the path corresponding to Op_{08} is: Op_{08}, Op_{05},Op_{10}, which has an aggregate cost of: $Cost_{Op_{08}} + Cost_{Op_{06}} + Cost_{Op_{10}} = 8 + 4 + 7 = 19$. Consequently the rank value of the two operations are: $rank(Op_{07}) = 1/15 = 0.0666$ and $rank(Op_{08}) = 1/19 = 0.0526$ respectively. Based on these values, Op_{07} is selected for execution.