Pareto-Optimal Trace Generation from Declarative Process Models

Juan F. Diaz[1], Hugo A. López[2], Luis Quesada[3], and Juan C. Rosero[1⋆⋆]

[1] Universidad del Valle, Cali, Colombia
{juanfco.diaz,juan.camilo.rosero}@correounivalle.edu.co
[2] Technical University of Denmark
Kongens Lyngby, Denmark
hulo@dtu.dk
[3] Insight Centre for Data Analytics, Cork, Ireland
luis.quesada@insight-centre.org

Abstract. Declarative process models (DPMs) enable the description of business process models with a high level of flexibility by being able to describe the constraints that compliant traces must abide by. In this way, a well-formed declarative specification generates a family of compliant traces. However, little is known about the difference between different compliant traces, as the only criterion used for comparison is satisfiability. In particular, we believe that not all compliant traces are alike: some might be sub-optimal in their resource usage. In this work, we would like to support users of DPMs in the selection of *compliant and optimal* traces. In particular, we use the Dynamic Condition Response language as our language to represent DPMs, extending it with a parametric definition of costs linked to events. Multiple types of cost imply that different traces might be optimal, each according to a different cost dimension. We encode cost-effective finite trace generation as a Constraint Optimisation Problem (COP) and showcase the feasibility of the implementation via an implementation in MiniZinc. Our initial benchmarks suggest that the implementation is capable of providing answers efficiently for processes of varying size, number of constraints and trace length.

Key words: Declarative Process Models, DCR graphs, Constraint Optimization Problems, Multi-Objective Optimization

1 Introduction

Starting from the seminal works of Pesic [1], declarative process models have been described as an alternative to provide flexibility in the orchestration of business processes. By describing a process in terms of its constraints, all non-compliant executions can be filtered (see Fig. 1, left-hand side). This flexibility allows knowledge workers to take discretionary decisions based on the nature of the case, knowing that any path they take will be compliant with the requirements of the process, for instance, laws or medical guidelines.

⋆⋆ Alphabetical order, equal author contribution

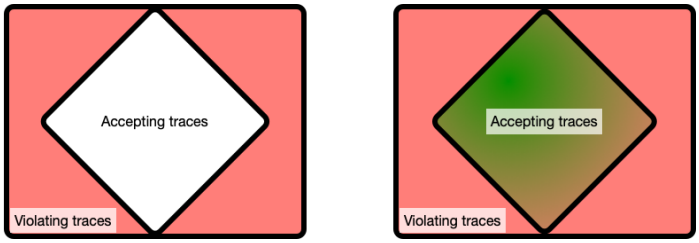

Fig. 1. Optimization of DPMs: constraints define the frame where valid executions live (left graphic), but even within the conforming space, some executions are more desirable than others (where optimality is denoted by colour range in the right-most graphic).

In our previous work [2], we showcased different application cases where discerning between compliant executions is necessary. Our inspiration comes from helping consultants and case workers plan executions using declarative process models. Consider a patient-handling process as an example. A dentist needs to decide whether to apply preventive, interceptive or corrective treatments depending on the impact on the patient, as well as other criteria (for instance, the associated cost). Not all the compliant treatments will be the same: some will be more cost-effective than others (depicted as the green regions in Fig. 1). Moreover, there will be cases in which different notions of costs play against each other. Consider a green-transition legislation process, where users are recommended to change to low-polluting technologies. Shifting from a gasoline car to an electric car will reduce the CO_2 expenses in the commuting process, but it will increase the economic cost for a family that already has a gasoline car.

Our contribution We introduce a method to identify the best traces in a declarative process model according to Pareto optimality. Our method is summarized in Figure 2. In this case, we take a declarative process model (using the DCR graphs [3] modelling notation), extended with multiple cost annotations for each event in the model, and a maximum trace length. The algorithms presented in this paper encode the generation of cost-effective traces as a constraint optimization problem, so, in case the model has a solution within the maximum trace length that dominates one of the cost dimensions, it will be part of the output. For our optimization problem, we only consider finite traces. This means that even if the DCR models considered in the input can generate omega-regular executions, only finite traces will be considered. This restriction obeys practical user needs. In our interactions with industry users of DCR graphs in the public and private sectors, they are less interested in a universe of traces. Instead, they are interested in *value-driven* traces, those that finish a case eventually. Similar considerations have been considered for other declarative process languages [4]. Similarly, finite executions (finished cases) are an assumption commonly used when working in process discovery and conformance checking. To showcase the feasibility of our approach, we have implemented the COP in MiniZinc [5]. Our initial benchmarks suggest that the implementation can provide answers efficiently for processes of varying size, number of constraints and trace length.

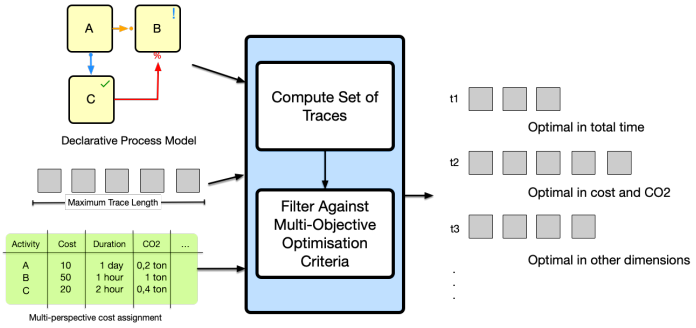

Fig. 2. Visual abstract of our approach. Our inputs (left hand side) include a DCR graph, the maximum trace length and a resource assignation. For our outputs (right hand side) we generate set of traces that are optimal in at least one cost dimension.

Paper structure We finish this section listing our related work. In section 2 We provide background to DCR graphs as well as multi-objective optimization techniques. An extension of DCR graphs with multi-dimensional costs is introduced in Section 3. We solve the discovery of optimal traces according to costs as combination of a Constraint Satisfaction Problem (CSP) and a Constraint Optimization Problem (COP) in Section 4. The benchmark of our solution is presented in Section 5, and we conclude in Section 6.

1.1 Related Work

Several approaches have been used to model multiobjective optimization in business processes. An evolutionary algorithm-based optimization framework is proposed in [6], employing the well-known Non-Dominated Sorting Genetic Algorithm II (NSGAII) [7], to generate optimized business processes by considering search space, fitness function, and optimization constraints. Evolutionary algorithms are also used in [8] to generate optimized business processes, based on predefined requirements, a task library, and input/output resources, providing a process graph, task library, and infeasibility degree. The work in [9] enhances their approach by incorporating a preprocessing phase into the evolutionary algorithm. [10] introduce a genetic algorithm-based approach for optimizing BPMN 2.0 processes and resource consumption using a Matlab simulation. Djedovic et al. propose a genetic algorithm based on colored Petri nets, considering specifications for possible processes, resource types, selection criteria, and elimination criteria [11]. Jiménez et al. [12] introduce a framework for the automatic generation of business processes using the process modelling language SDeclare to produce enactment plans that are optimal by removing Pareto-dominated plans. [13] proposes a method to check trace conformance and compliance in business processes. Additionally, [14] uses an approach involving Pareto efficiency to optimize resource allocation.

In this study, we introduce an innovative approach by leveraging DCR graphs, a declarative modelling language, to generate potential business processes incorporating user-defined cost-associated features to determine the optimality of the generated processes, we use a Pareto optimization algorithm for sorting. Our work distinguishes

itself from previous studies in several key aspects. Firstly, our primary focus lies in discovering optimal business processes based on activities that already have associated resource costs rather than allocating resources to activities for optimization purposes, as observed in [8, 9, 10, 11], and [14]. Moreover, we adopt a declarative approach using DCR graphs, whereas works such as [10] employ the more imperative Oracle Business Process Management software. Our work also diverges from [12] in the approach taken: while [12] involves assigning roles to activities and considering resource availability based on roles, our work focuses on assigning resource consumption to activities and optimizing the overall resource utilization of a process. Another aspect is that we are using a constraint-based model to generate traces and to specify the requirements needed for a trace to be valid and compliant with a given DCR graph, which differs from the algorithm proposed in [13] as it is an algorithm set to check the compliance of an existing trace, as oposed to our work that explores the search space of the graph to generate compliant traces. Finally our prunning method to accelerate reaching the Pareto frontier is different than the ranking method used by [14].

2 Background

In the following section, we will recall the definitions of DCR graphs and multi-objective process optimization. Our work is framed on the classical definitions by Hildebrandt and Mukkamala [3].

2.1 DCR Graphs

Definition 1 (DCR graphs [3]). *A Dynamic Condition Response Graph (DCR Graph) is a tuple* $G = \langle E, M, Act, \rightarrow\bullet, \bullet\rightarrow, \pm, l \rangle$*, where 1.* E *is the set of events. 2.* $M \in \mathcal{M}(G) = \mathcal{P}(E) \times \mathcal{P}(E) \times \mathcal{P}(E)$ *is a marking and* $\mathcal{M}(G)$ *is the set of all markings. 3.* Act *is the set of actions. 4.* $\rightarrow \bullet \subseteq E \times E$ *is the condition relation. 5.* $\bullet \rightarrow \subseteq E \times E$ *is the response relation. 5.* $\pm \colon E \times E \rightharpoonup \{+, \%\}$ *defines the dynamic inclusion/exclusion relations by* $e \rightarrow + e'$ *if* $\pm(e, e') = +$ *and* $e \rightarrow \% e'$ *if* $\pm(e, e') = \%$*. 6.* $l \colon E \rightarrow Act$ *is a labelling function.*

In DCR graphs, the condition and response relations allow for cyclic interactions. The marking $M = (Ex, Re, In) \in M(G)$ comprises three sets of events: executed events (Ex), pending responses (Re) that are yet to be executed or excluded, and currently included events (In). The dynamic exclusion and inclusion relations, denoted as $\rightarrow +$ and $\rightarrow \%$, are represented in the partial map $\pm \colon E \times E \rightharpoonup \{+, \%\}$. These relations enable events to be dynamically included or excluded from the graph. An event e *is enabled* in a marking $M = (Ex, Re, In)$ if 1) $e \in In$ and 2) if $\exists f.(f, e) \in \rightarrow\bullet \implies f \in Ex \vee f \notin In$. When evaluating constraints, only the events currently included are considered. For instance, if an event a has a response event b, but b is excluded, the occurrence of b is not necessary for the graph to be acceptable. Intuitively, the relation $e \rightarrow + e'$ indicates that when event e occurs, event e' is included in the graph. On the other hand, $e \rightarrow \% e'$ signifies that when event e occurs, event e' is excluded from the graph. Moreover, we will simplify our development by assuming l to be a bijective function [15], thus allowing us to compute its inverse.

The execution semantics of DCR graphs is defined in terms of a labelled transition system, where states are defined by markings, and transitions are fired events.

Definition 2 (Transitions [3]). *For a DCR graph* $G = \langle E, M, Act, \rightarrow\bullet, \bullet\rightarrow, \pm, l \rangle$*, the corresponding labelled transition system* $T(G)$ *to be the tuple* $\langle \mathcal{M}(G), M, \rightarrow \subseteq \mathcal{M}(G) \times Act \times \mathcal{M}(G) \rangle$ *where* $\mathcal{M}(G)$ *is the set of markings* G*,* $M \in \mathcal{M}(G)$ *is the initial marking, and* $\rightarrow \subseteq \mathcal{M}(G) \times (E \times Act) \times \mathcal{M}(G)$ *is the transition relation given* $M' \xrightarrow{(e,a)} M''$ *where: 1.* $M' = (Ex', Re', In')$ *is the marking before the transition. 2.* $M'' = (Ex' \cup \{e\}, Re'', In'')$ *is the marking after the transition. 3.* $e \in In'$ *and* $l(e) = a$ *4.* $\{e' \in In' \mid e' \rightarrow\bullet e\} \subseteq Ex'$ *5.* $In'' = (In' \cup \{e' \mid e \rightarrow + e'\}) \backslash \{e' \mid e \rightarrow \% e'\}$ *6.* $Re'' = (Re' \backslash \{e\}) \cup \{e' \mid e \bullet\rightarrow e'\}$

We define a trace $t = (e_0, a_0), (e_1, a_1), ...$ as a (possibly infinite) sequence of transitions $M_i \xrightarrow{(e_i, a_i)} M_{i+1}$ where $M_i = (Ex_i, Re_i, In_i)$ and $M_0 = M$. A trace is *accepting* if $\forall i \geq 0$, $e \in Re_i, \exists j \leq i \colon (e = e_j \vee e \notin In_j)$. This means that there is no event that is both included and pending at the same time, without having been executed first. Given an accepting trace $t = (e_0, a_0), (e_1, a_1), \ldots$, we also define $Actions(t) = \langle a_0, a_1, ... \rangle$.

The definition includes the following key points: (i) Markings before and after transitions, (ii) Execution requirement for an event e to be included and the condition that all currently included condition events for e must have been executed, (iii) Updates to the set of included events and pending responses during a transition.

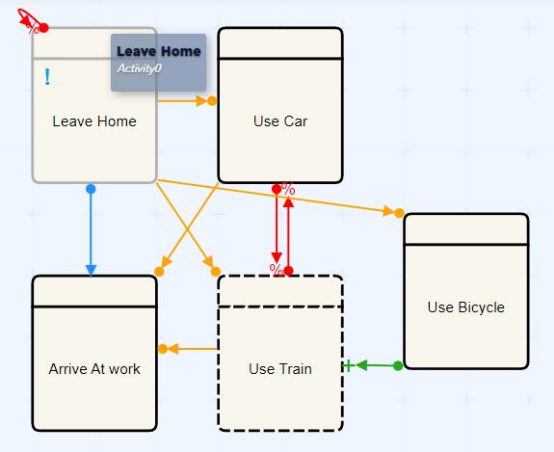

Fig. 3. A simple DCR graph showing the inclusion, exclusion, condition, and response relation

Figure 3 showcases a simple DCR graph representing the process of going to work. The graph consists of 5 events, each event has a label, e.g., $Activity0$ labelled as $Leave\,Home$. The markings indicate the status of each event, where some are included (solid borders), excluded (dashed borders), pending (decorated with a ! symbol) or executed (decorated with a ✓). The graph illustrates in the hovering label the event (Activity0) and the action (Leave Home) associated. The graph shows four types of relations: inclusions, exclusions, conditions, and responses. Executing certain events triggers changes, such as including or excluding other events. For example, executing $Use\,Bicycle$ includes $Use\,Train$ while executing $Use\,Car$ excludes it. Condition relations specify prerequisites: both $Use\,Car$ and $Use\,Train$ must be executed or excluded before $Arrive\,At\,Work$ can occur. The response relation highlights the requirement of executing or excluding $Arrive\,At\,Work$ when $Leave\,Home$ is executed. So for a trace to be valid, we must first start by leaving home ($Leave\,Home$), then we would either use the car

($Use\,Car$) and reach work ($Arrive\,At\,Work$) or use the bicycle ($Use\,Bicycle$) to then take a train ($Use\,Train$) to work ($Arrive\,At\,Work$).

2.2 Pareto Optimization

A multi-objective optimization problem requires the task of maximizing a series of $k > 1$ objective functions, usually in a conflict between them. Let S be a set of decision vectors $\vec{s_1}, \ldots, \vec{s_n}$. An objective vector is the projection of a decision vector consisting of the values of the objective functions $z = (f_1(\vec{s}), \ldots, f_n(\vec{s}))$. The optimization problem is defined as:

$Maximize\{f_1(\vec{s}), \ldots, f_n(\vec{s})\}$, such that $\vec{s} \in S, f_i \colon S \to \mathbb{R}, S \neq \emptyset$. According to [16], Pareto optimization searches for dominant decision vectors. For $\vec{s}, \vec{s'} \in S$ we say that $\vec{s} \in S$ dominates $\vec{s'}$ if:

$$\forall i \in 1, \ldots, k \colon f_i(\vec{s}) \geq f_i(\vec{s'}) \wedge \exists j \in 1, \ldots, k \colon f_j(\vec{s}) > f_j(\vec{s'})$$

Finally, we say that $\vec{s} \in S$ is a Pareto optimal vector if it does not exist $\vec{s'} \in S$ such that $\vec{s'}$ dominates $\vec{s}$. Pareto solutions can be understood as those balance points where it is impossible to better optimize one of the components of the decision vector without compromising the optimization of any other components.

3 DCR graphs with Multi-dimensional Costs

In this section, we extend DCR graphs with notions of multi-dimensional costs, and introduce the generation of cost-optimal traces as an optimization problem based on Pareto optimization.

Definition 3. *An Extended Dynamic Condition and Response Graph (EDCR graph) is a tuple $EG = \langle G, \Phi, \$, \oplus \rangle$, where:*

1. *G is a DCR graph,*
2. *$\Phi = \{\phi_1, \ldots, \phi_n\}$ is a finite set of features,*
3. *$\$ \colon \Phi \times Act \to \mathbb{R}$. is the cost of the action associated with a feature, and*
4. *$\oplus \colon \Phi \times (\mathbb{R} \times \mathbb{R}) \to \mathbb{R}$ is the aggregation function according to a feature.*

We assume that $\oplus$ is commutative, associative, and has an identity element, thus $\oplus(\phi, (r_1, r_2)) = \oplus(\phi, (r_2, r_1))$ and $\oplus(\phi, (r_3, \oplus(\phi, (r_2, r_1)))) = \oplus(\phi, (\oplus(\phi, (r_3, r_2), r_1)))$. Then we write $\oplus(\phi, (r_1, r_2, \ldots, r_j))$ to denote the aggregation of j results for feature ϕ. A parametric definition of aggregation allows us to have multiple ways of treating resources. For instance, some features will be aggregated with a simple sum, while others can be aggregated with a minimum/maximum among the resources.

Let EG be a $EDCR$ graph and $a \in Act$. Then we define $\alpha(\cdot) \colon Act \to \mathbb{R}^{|\Phi|}$ to denote the multi-dimensional cost of executing an activity. For representing the cost of an action a in EG, we define $\alpha(a)$ as: $\alpha(a) = (\$(\phi_1, a), \ldots, \$(\phi_n, a))$.

Let $t = \langle (e_0, a_1), \ldots, (e_m, a_m) \rangle$ be a trace in EG. The cost of feature ϕ on t is denoted $\$(\phi, t)$ and it is defined as: $\$(\phi, t) = \oplus(\phi, \$(\phi, a_1), \ldots, \$(\phi, a_m))$ and the total cost of t is defined as the tuple: $\alpha(t) = (\$(\phi_1, t), \ldots, \$(\phi_n, t))$.

We can now define our optimization problem related to EDCR graphs.

Definition 4. *Given a $EDCR$ graph $EG = (G, \Phi, \$, \oplus)$, and a maximum trace length k, the problem of finding optimal traces of a $EDCR$ graph (*EDCR-OPT *for short), is defined as follows:*

Input: $EG, k \in \mathbb{N}, (opt_\phi)_{\phi \in \Phi}$

Output: $T = \{\langle (e_0, a_0), \ldots, (e_j, a_j) \rangle \mid j \leq k\}$ *such that $\alpha(t)$ is Pareto optimal for each $t \in T$ according to the optimization criteria $(opt_\phi)_{\phi \in \Phi}$ ($opt_\phi \in \{min, max\}$).*

4 Implementing EDCR-OPT as a Constraint Optimization Problem

In this section, we present our solution to the problem specified in Definition 4. We first introduce a constraint-based model to generate a valid trace given a DCR graph. We then focus on the optimization by first presenting a Branch and Bound approach to compute an approximation of the pareto optimal frontier and and then describing a filtering process to filter out dominated solutions in the approximation.

The solution was implemented using: (i) Python version 3.9.5, (ii) MiniZinc version 2.6.2, (iii) Gecode Solver version 6.3.0, and (iv) Python-minzinc library version 0.9.0. Our implementation is available online[3].

4.1 Generating a valid trace

In this section we present a constraint-based model for generating a valid trace of an EDCR graph $EG = \langle G, \Phi, \$, \oplus \rangle$, where $G = \langle E, M, Act, \rightarrow\bullet, \bullet\rightarrow, \pm, l \rangle$, with a maximum trace length k and $T(G)$ is the transition system for G.

We formally introduce the model in terms of its inputs, outputs, and constraint as follows:
Inputs: EDCR graph $EG = \langle G, \Phi, \$, \oplus \rangle$, and a maximum length $k \in \mathbb{N}$.
Outputs: $t = \langle (e_0, a_0), \ldots, (e_j, a_j) \rangle$ such that $j \leq k$.
Constraints: $0 \leq i < j - 1$:

$$M_i \xrightarrow{(e,a)} M_{i+1} \implies Ex_{i+1} = Ex_i \cup \{e\} \tag{1}$$

$$M_i \xrightarrow{(e,a)} M_{i+1} \implies Re_{i+1} = (Re_i \backslash \{e\}) \cup \{f \mid (e,f) \in \bullet\rightarrow\} \tag{2}$$

$$M_i \xrightarrow{(e,a)} M_{i+1} \wedge (e,f) \notin \bullet\rightarrow \implies Re_{i+1} = Re_i \tag{3}$$

$$\forall (e,f) \in \rightarrow\% : M_i \xrightarrow{(e,a)} M_{i+1} \wedge (e,f) \notin \rightarrow+ \implies (In_{i+1} = In_i \setminus \{f\}) \tag{4}$$

$$\forall (e,f) \in \rightarrow+ : M_i \xrightarrow{(e,a)} M_{i+1} \implies In_{i+1} = In_i \cup \{f\} \tag{5}$$

$$M_i \xrightarrow{(e,a)} M_{i+1} \wedge (e,f) \notin \rightarrow+ \wedge (e,f) \notin \rightarrow\% \implies In_{i+1} = In_i \tag{6}$$

$$M_i \xrightarrow{(e,a)} M_{i+1} \wedge \exists (f,e) \in \rightarrow\bullet \implies (e \in In_i \wedge f \in Ex_i \vee f \notin In_i) \tag{7}$$

$$In_j \cap Re_j \cap Ex_j = \emptyset \tag{8}$$

Our constraint system is modelled based on the criteria for the LTS of DCR graphs (c.f. Def. 2). The intuition behind the equations of our constraint system is the following: Equation 1 describes the effects imposed in the set of executed markings after a transition. Equations 2 and 3 define the pending markings after an event transition. The effects of transitions in the set of included events are described by Equations 4, 5, and 6. Equation 4 excludes the target event of an exclusion relation if the event is not included at the same time. Equation 5 handles the effects of executing an event source of an inclusion relation. Equation 6 preserve the included sets if no exclusions or inclusions have been affected. Equation 7 ensures that an executed event must be allowed, meaning it must be included at that moment, and all events that are conditions for its execution must have been executed previously or be excluded. Finally, Equation 8 models the accepting state of the graph, indicating that for a trace to be accepted, there should not be any pending and included event that has not been executed.

[3] `https://github.com/JuanK120/dcrGraph`

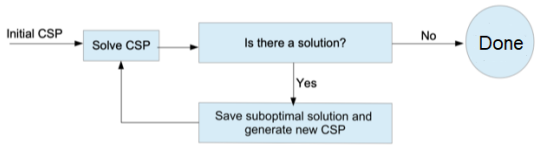

Fig. 4. An adaptation of Branch and Bound to compute a set containing the Pareto optimal frontier

4.2 Computing the Pareto optimal frontier

There are two phases in the computation of the frontier. We first compute a superset of traces that contains the frontier. In the second phase, we filter out the dominant solutions in the superset.

Computation of the superset. Features, costs, and aggregations for an Extended DCR graph implementation are added to the model of Section 4.1 following the definitions in Section 3.

The process followed for the computation of the superset is presented in Figure 4. We extend a basic constraint-based Branch and Bound approach [17] to save the solutions that are found during the execution of the process. During the iterative process, we add a constraint at each iteration to ensure that each new solution found is different, and better in at least one feature compared to the previously found solutions.

In the context of DCR graphs, we apply the Branch and Bound algorithm as follows: initially, we execute the DCR model to find a single solution. Once a possible trace, denoted as t, is found, we store this solution. Subsequently, a new CSP is created. In this CSP, we introduce an additional constraint to ensure that any subsequent solution, denoted as t', must be superior in at least one objective. This constraint is formulated as follows: $\exists(\phi \in \Phi)(\alpha(t')\phi < \alpha(t)\phi)$. We repeat this process until it becomes infeasible to find better solutions. This approach effectively controls the size of the generated solution set by discarding solutions that are known to be sub-optimal and guarantees that Pareto optimal solutions are not overlooked.

Filtering the superset. After applying the Branch and Bound algorithm and obtaining a set of potential solutions, the next step is to filter this set using the Pareto optimality concept. The Pareto optimal solutions represent the optimal trade-off solutions in a multi-objective optimization problem, where improving one objective comes at the expense of worsening another. By using a quadratic algorithm, we can efficiently evaluate and filter the potential solutions to identify the Pareto optimal solutions, which provide the best overall solutions considering all objectives simultaneously.

5 Scalability of the Solution

To evaluate the performance of our solutions, we developed a DCR graph generator that allowed us to vary multiple features of the graph. These features included trace length (k), the number of events, features, conditions, responses, inclusions, and exclusions. To assess the impact of each dimension on the tool's performance, we conducted separate tests for each dimension, fixing the other dimensions at a reasonable magnitude, and then gradually increasing the size of the dimension under examination. for example, for analyzing number of events, we fixed all other dimension at 10, as it would provide a variety of options for the model to chose from, and then we started at 15 events, then 17, then 19 and so on until reaching 45. Additionally for each step of the analysis, we generated multiple graphs with the same dimensions, the total number of graphs

generated was 3082. For each DCR graph, we collected three performance metrics: nodes, which is the number of explored nodes of the search tree of the constraint model done by the solver during the search for solutions, solveTime, which is the time spent by the MiniZinc motor running the constraint model provided by the python-minizinc tool, and totalTime,the overall solving time of the tool calculated by the python algorithm, in which we start the count at the begining of the implemented tool until the moment just before the end of the process. We created box plots for each dimension to analyze the data, representing the measures obtained from all graphs with that specific dimension. These box plots visually represent the performance measures across different dimensions. For instance, in Figure 5, the plot compares one of the features of the graph (X-axis) to the time it took the algorithm to run (Y-axis). For example, in the first plot, with conditions, each boxplot displays the interquartile range (IQR) with a median line within the box. Whiskers extend from the box, indicating the range within a certain distance from the quartiles. Any data points beyond the whiskers are considered potential outliers, represented as diamonds in this case.

All DCR graphs, data files, and plots generated can be accessed in the code repository[4]. Figure 5 illustrates the impact of different variables on the performance measures. Conditions and responses exhibit the greatest influence, as evidenced by their larger values. This is expected since conditions and responses control the graph flow and complexity of solutions. The number of events has a significant effect on the number of explored nodes, as a larger number of possible events expands the search space. However, this impact diminishes in solve time and total time measures.

Regarding the variables k (trace length) and feats (features), their impact is less pronounced compared to conditions, responses, and events. The growth in k expands the search space, albeit not as significantly as the increase in the number of events, this is because even though k allows for longer traces it does no necessarily affects, the general behaviour of a DCR graph, nor does it guarantees that there are traces of a length k, as depending of the specific DCR graph there could be the case where there are no traces beyond a lesser length than k. Feats contribute to the problem's complexity, but their effect is not as prominent as the other variables. This is because trace generation plays a larger role in the complexity of the problem, and it is logical, given that this is the combinatorial part of our Optimization problem, as trace generation involves choosing which events form a trace. Inclusions and exclusions have a minimal impact on performance measures. Although they may increase graph complexity, overall the performance depends more on specific constraints, restrictions, and requirements within the graph. Thus, compared to conditions and responses, inclusions and exclusions have a lesser effect on performance. this is because inclusions and exclusions do not significantly influence the flow of the DCR graph, inclusions depending on if there are any excluded events at any point might even not have any impact at all in the graph, as if an event being included is already included in the graph, then it is irrelevant. a similar case happens with exclusions, in general work to avoid certain combinations of events from happening, generally speaking, it doesn't go beyond that, as opposed to conditions and responses, which have a bigger impact, as they control, the requisites for events to be executed in the case of conditions, and also the requirements to reach an accepting state in case of responses.

Also, potential outliers can be noted due to the conformation of specific graphs, especially in variables that greatly impact the flow of a DCR graph, like conditions and responses, this is due to the fact that depending on how a DCR graph is composed, a large number of events in a trace might be necessary to reach an accepting state, for example, if there are chains of relations in the generated graphs, for example, if a pending event has 2 other events as conditions, and those events have other events as conditions, these would make it so that any possible trace would take

[4] `https://github.com/JuanK120/dcrGraph/tree/master/Tests/Detailed`

longer to execute, as the execution of multiple previous events would be required to reach an accepting state.

6 Conclusions

In this work, we explored process optimization based on the model based on the proposed extension for DCR graphs. We introduced DCR graphs as a modeling technique and extended them to incorporate multi-objective optimization in business processes. To evaluate our implementation we conducted a performance analysis to assess the efficiency and effectiveness of our approach. By considering various factors such as trace length, events, conditions, responses, inclusions, and exclusions, we gained insights into the impact of these variables on time and complexity. This research contributes to the field of business process optimization by offering a novel application of Extended DCR graphs. Our findings highlight the importance of considering multi-objective optimization and provide a foundation for future research. Understanding how variables affect optimization solutions enables organizations to make informed decisions and enhance their competitiveness in dynamic markets.

Future work involves considering how to integrate this approach with partially executed traces, paving the way to the integration with streaming and predictive process mining approaches. Moreover, we would like to conduct a more complex empirical study to evaluate the simultaneous variation of multiple variables and explore real-world applications with diverse events, resources, and features. Additionally, our solution should be refined and integrated into a user-friendly tool capable of handling partial traces, providing businesses with tailored recommendations for process improvement.

References

1. M. Pesic, "Constraint-based workflow management systems : shifting control to users," Ph.D. dissertation, Industrial Engineering and Innovation Sciences, 2008.
2. A. Burattin, A. Gianola, H. A. López, and M. Montali, "Exploring the conformance space (extended abstract)," in *ITBPM@BPM*, ser. CEUR Workshop Proceedings, vol. 2952. CEUR-WS.org, 2021, pp. 62–67.
3. T. T. Hildebrandt and R. R. Mukkamala, "Declarative event-based workflow as distributed dynamic condition response graphs," *arXiv preprint arXiv:1110.4161*, 2011.
4. G. De Giacomo, R. De Masellis, and M. Montali, "Reasoning on ltl on finite traces: Insensitivity to infiniteness," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 28, no. 1, 2014.
5. N. Nethercote, P. J. Stuckey, R. Becket, S. Brand, G. J. Duck, and G. Tack, "Minizinc: Towards a standard CP modelling language," in *Principles and Practice of Constraint Programming - CP*, vol. 4741. Springer, 2007, pp. 529–543.
6. N. Mahammed, S. M. Benslimane, A. Ouldkradda, and M. Fahsi, "Evolutionary Business Process Optimization using a Multiple-Criteria Decision Analysis method," in *Intl. Conf. on Computer, Information and Telecommunication Systems (CITS)*. Alsace, Colmar, France: IEEE, Jul. 2018, pp. 1–5.
7. K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: Nsga-ii," *IEEE transactions on evolutionary computation*, vol. 6, no. 2, pp. 182–197, 2002.

8. K. Vergidis, D. Saxena, and A. Tiwari, "An evolutionary multi-objective framework for business process optimisation," *Applied Soft Computing*, vol. 12, no. 8, pp. 2638–2653, Aug. 2012.
9. K. Georgoulakos, K. Vergidis, G. Tsakalidis, and N. Samaras, "Evolutionary Multi-Objective Optimization of business process designs with pre-processing," in *IEEE Congress on Evolutionary Computation (CEC)*. Spain: IEEE, Jun. 2017, pp. 897–904.
10. A. Djedovic, E. Zunic, Z. Avdagic, and A. Karabegovic, "Optimization of business processes by automatic reallocation of resources using the genetic algorithm," in *XI International Symposium on Telecommunications (BIHTEL)*. Sarajevo, Bosnia and Herzegovina: IEEE, Oct. 2016, pp. 1–7.
11. Y.-W. Si, V.-I. Chan, M. Dumas, and D. Zhang, "A Petri Nets based Generic Genetic Algorithm framework for resource optimization in business processes," *Simulation Modelling Practice and Theory*, vol. 86, pp. 72–101, Aug. 2018.
12. A. Jiménez-Ramírez, B. Weber, I. Barba, and C. Del Valle, "Generating optimized configurable business process models in scenarios subject to uncertainty," *Information and Software Technology*, vol. 57, pp. 571–594, Jan. 2015.
13. A. Burattin, G. Guizzardi, F. M. Maggi, and M. Montali, "Fifty shades of green: How informative is a compliant process trace?" in *Advanced Information Systems Engineering*, P. Giorgini and B. Weber, Eds. Cham: Springer International Publishing, 2019, pp. 611–626.
14. O. López-Pintado, M. Dumas, M. Yerokhin, and F. M. Maggi, "Silhouetting the cost-time front: Multi-objective resource optimization in business processes," in *Business Process Management Forum*, A. Polyvyanyy, M. T. Wynn, A. Van Looy, and M. Reichert, Eds. Cham: Springer International Publishing, 2021, pp. 92–108.
15. H. A. López, S. Debois, T. Slaats, and T. T. Hildebrandt, "Business process compliance using reference models of law," in *International Conference on Fundamental Approaches to Software Engineering*, 2020, pp. 378–399.
16. W. Jakob and C. Blume, "Pareto optimization or cascaded weighted sum: A comparison of concepts," *Algorithms*, vol. 7, no. 1, pp. 166–185, 2014.
17. P. van Beek, "Backtracking search algorithms," in *Handbook of Constraint Programming*, ser. Foundations of Artificial Intelligence, F. Rossi, P. van Beek, and T. Walsh, Eds. Elsevier, 2006, vol. 2, pp. 85–134. [Online]. Available: https://doi.org/10.1016/S1574-6526(06)80008-8

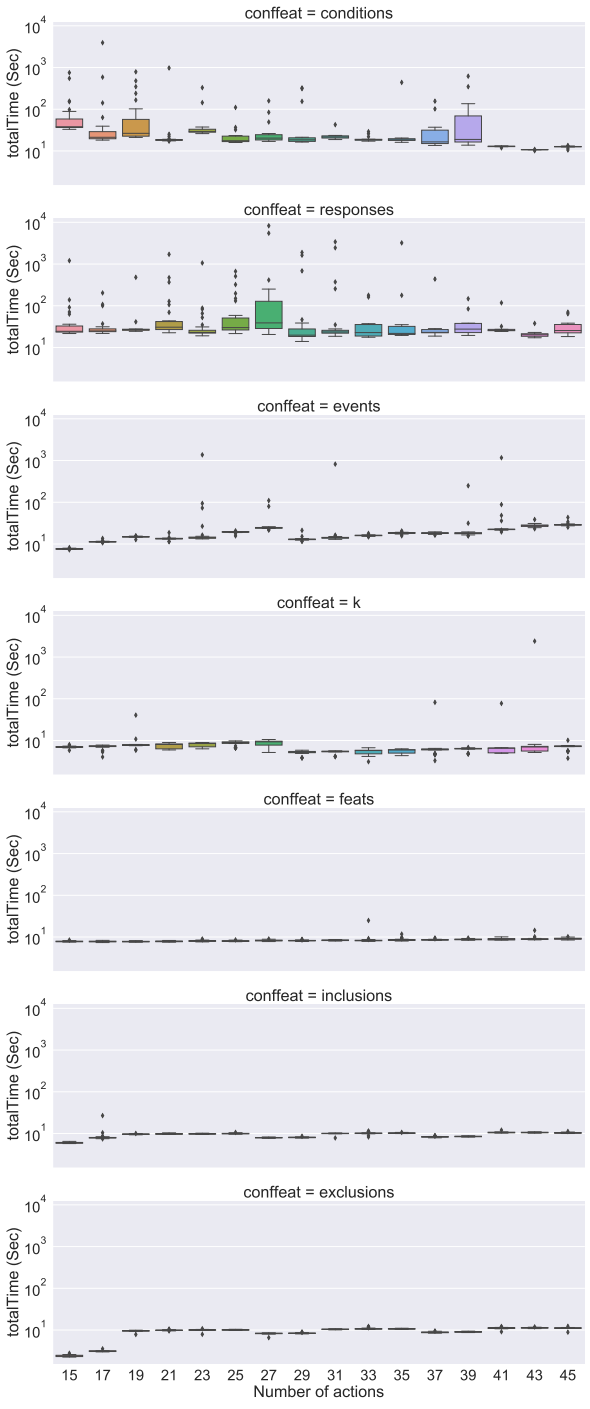

Fig. 5. Performance of the solution measuring the total time measure