Perturbing Event Logs to Identify Cost Reduction Opportunities: A Genetic Algorithm-based Approach

W.Z. Low, J. De Weerdt, M.T. Wynn, A.H.M. ter Hofstede, W.M.P. van der Aalst, and S. vanden Broucke

Abstract— **Organisations are constantly seeking new ways to improve operational efficiencies. This research study investigates a novel way to identify potential efficiency gains in business operations by observing how they are carried out in the past and then exploring better ways of executing them by taking into account trade-offs between time, cost and resource utilisation. This paper demonstrates how they can be incorporated in the assessment of alternative process execution scenarios by making use of a cost environment. A genetic algorithm-based approach is proposed to explore and assess alternative process execution scenarios, where the objective function is represented by a comprehensive cost structure that captures different process dimensions. Experiments conducted with different variants of the genetic algorithm evaluate the approach's feasibility. The findings demonstrate that a genetic algorithm-based approach is able to make use of cost reduction as a way to identify improved execution scenarios in terms of reduced case durations and increased resource utilisation. The ultimate aim is to utilise cost-related insights gained from such improved scenarios to put forward recommendations for reducing process-related cost within organisations.**

I. Introduction

Business process improvement is concerned with identifying process redesign opportunities bearing in mind the potential impact that these redesign actions may have on different dimensions such as time, cost, quality and flexibility [15][16]. By having detailed insights into how business operations were carried out in the past, it is possible to explore whether these same operations can be performed better (e.g., can cases be completed faster?, can operational cost be reduced?, can the quality of the outcomes be improved?). This paper presents a technique to intelligently search for alternative business process execution scenarios with the aim of "improving the history". The main objective is to discover execution scenarios which are cheaper (better) than the original (baseline) scenario to gain insights for future redesign activities.

The starting point of our cost-informed process improvement approach is an event log that contains a detailed record of business operations over a certain time period. A number of key characteristics of the process are kept the same (such as the activities performed and their durations, and the arrival times of cases), while other elements within the log (such as resource allocations, ordering and start times of activities) are adjusted in order to explore different execution scenarios. By making use of a generic cost structure that assigns cost to different trade-offs, the cost of various execution scenarios are computed and compared.

The contribution of this paper is the development of a genetic algorithm-based approach to facilitate the exploration of different execution scenarios. Optimisation strategies are defined to explore cost-optimal execution scenarios that take into account trade-offs from multiple process dimensions.

The remainder of this paper is organised as follows. The related work is reviewed and discussed in Section II. In Section III, a cost-informed process improvement approach is proposed with the facilitation of a motivating example. Next, the genetic algorithm-based solution approach is discussed in Section IV. Section V discusses the experimental results. Section VI concludes this paper and states potential future work.

II. Related Work

Within the field of business process management (BPM), the concepts of business process redesign (BPR) and improvement (BPI) are highly relevant to researchers and practitioners. A number of case studies have looked into creating a framework to list and classify best practices to facilitate BPR within organisations [15][17][19]. In [16], a number of BPR best practices and approaches are provided. BPR has also been applied and evaluated via case studies carried out in organisations from various fields [12][31]. An approach that uses performance measures to quantify the impact and trade-offs of business process redesign actions on all dimensions of workflow performance has also been developed [11].

BPR and BPI have an overlapping interest with operations research, which is defined as "a scientific approach to decision making that seeks to best design and operate a system, usually under conditions requiring the allocation of scarce resources" [26]. This normally involves the use of one or more mathematical models, where understanding of a situation could be further promoted by mathematical representations of an actual situation [26]. An optimisation problem describes the model that seeks to find values of the decision variables that optimise an objective function. There are a number of optimisation problem categories, such as shop scheduling problems [10] and resource-constrained (multi-) project scheduling problems (RCPSP/RCMPSP) [3][6]. These can be

W.Z. Low, M.T. Wynn, A.H.M. ter Hofstede, and W.M.P. van der Aalst are with Queensland University of Technology (QUT), Australia (emails: w4.low@qut.edu.au, m.wynn@qut.edu.au, a.terhofstede@qut.edu.au).

A.H.M. ter Hofstede and W.M.P. van der Aalst are also with Technische Universiteit Eindhoven (TU/e), The Netherlands (email: w.m.p.v.d.aalst@tue.nl).

J. De Weerdt and S. vanden Broucke are with KU Leuven, Belgium (emails: jochen.deweerdt@kuleuven.be, seppe.vandenbroucke@kuleuven.be).

This work is supported by Australian Research Council (ARC) Discovery Grant with the Grant Number 120101624

addressed by, but not limited to, techniques such as simulated annealing [1] and genetic algorithms [2].

As stated by van der Aalst [22], scheduling problems are similar to business process optimisation problems, as both disciplines aim to optimise the allocation of resources to tasks. In [25], the topics of business process modelling, analysis, and optimisation are reviewed. Optimisation has been applied to the design of business processes using genetic algorithms [24]. The effects that resource behaviours and relationships have on resource allocations and the business process have been studied in [8][14]. Various methods to schedule or assign resources based on their properties (for example, performance, compatibility, skill sets, and many more) have also been investigated [4][5][13][17][18]. In addition, [9] uses optimisation techniques as an attempt to optimise resource allocation in business processes.

Optimality of business processes is determined by multiple, often conflicting objectives [7]. In [20], reduction of cost and flow time is used as the goal of a business process optimisation approach. Xu et al. looked into a number of variants where business processes were optimised using multiple objectives [27][28][29]. In [28], the flow time and the cost of business processes were optimised separately. The structural features of business processes is then explored to enable further utilisation of resources. Different heuristic scheduling strategies that take into account resource availability constraints such as resource slots, resource capabilities, process task dependencies, and instance deadlines have also been studied [29]. Nonetheless, cost was not taken into account in this work. Two business process resource planning methods, where deadline and resource utilisation trade-offs were considered, was provided in [27]. Although cost is taken into account (but is not a priority), resource availability and eligibility constraints are not taken into account for optimisation.

In contrast to the works discussed above, this paper:

- identifies the fixed and variable parts of an event log;
- identifies the cost-optimal execution scenarios by changing process history (event log) and learning from it; and
- incorporates different cost-related dimensions into a cost structure, allowing more sensible cost-based trade-offs within the genetic algorithm.

III. Cost-Informed Log Perturbation

The motivation for this research is to identify a more efficient execution scenario, where cases finish earlier and the utilisation of the resources is more optimal. We apply the notion of cost towards these time and resource efficiency measures. Alternative execution scenarios are explored where a scenario with lower cost represents a more efficient scenario.

In order to explore different execution scenarios, the identification and separation of fixed and variable parts of a typical event log is the first step in this approach. Different execution scenarios are explored by manipulating the variable parts of the event log. A cost structure is then defined as an objective function to determine the fitness of the execution scenarios in terms of process-related cost, taking into consideration cost-informed trade-offs between multiple aspects such as case durations and resource utilisation. The cost functions within the cost structure can be configured, and further customised by organisations. An execution scenario that is infeasible is defined as *unsafe*, for example, a resource does not have the authority to carry out a task. A *violation cost* is used to penalise the execution scenarios that are unsafe. Execution scenarios that are more cost-efficient are then identified and analysed. Fig. 1 illustrates the the proposed approach for the generation of cost-informed alternative scenarios.

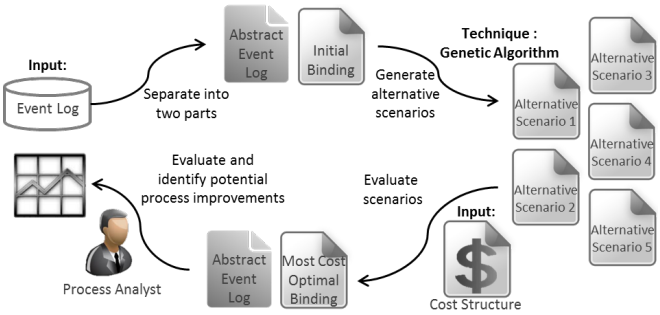

Fig. 1. Overview of the cost-informed process improvement approach.

1) Event Log: An event log is a data store that records potentially vast amounts of process events, where event-related information such as resourcing decisions, timestamp of the events, or data elements are stored [21][23]. Table I illustrates an event log fragment of the example car insurance claim process. As we are seeking improvements with regard to execution time and resource allocation, the historical event attributes that influence timing and resource allocation are considered as the variable part (i.e. binding). All other historical attributes, such as case properties are kept fixed, thus part of what we call the abstract event log.

2) Abstract Event Log: An abstract event log defines the event log attributes that will remain fixed, which include the collection of cases, activities, tasks, and case or activity properties. Activities are mapped to their corresponding case, task, and duration. Arrival times of cases and the order of activities within a case are also defined in the abstract event log.

3) Binding and Concrete Event Log: A binding defines the variable component of an event log, where different execution scenarios can be generated by modifying information such as allocation of resources to activities and timestamps of those activities. A binding consists of a set of resources and their mapping onto each activity. The start and end times of activities are part of a binding as well, where:

- an activity must start after its case arrival time;
- an activity's end time must follow after the activity's start time; and
- an activity cannot start when its preceding activity is not completed yet.

TABLE I

A POSSIBLE FRAGMENT OF THE CAR INSURANCE CLAIM EVENT LOG IN CHRONOLOGICAL ORDER.

Case ID	Activity	Timestamp	Transition Type	Resource	Property (Damage Type)	...
1	Lodge Claim	10/06/13 09:31:00	Start	IC1	Windscreen	...
1	Lodge Claim	10/06/13 09:39:00	Complete	IC1	Windscreen	...
1	Review Claim	10/06/13 09:42:00	Start	IC5	-	...
2	Lodge Claim	10/06/13 09:45:00	Start	IC1	Theft	...
2	Lodge Claim	10/06/13 09:50:00	Complete	IC1	Theft	...
2	Review Claim	10/06/13 09:55:00	Start	IC3	-	...
1	Review Claim	10/06/13 10:00:00	Complete	IC5	-	...
2	Review Claim	10/06/13 10:10:00	Complete	IC3	-	...
...	...	...	...	...	...	...

TABLE II

AN ABSTRACT EVENT LOG (LEFT) AND BINDING (RIGHT) THAT CORRESPONDS TO THE CAR INSURANCE CLAIM EVENT LOG IN TABLE I.

Abstract Event Log						
ID	Case ID	Activity	Property (Damage Type)	Duration	Preceding Activity	Succeeding Activity
341	1	Lodge Claim	Windscreen	00:08:00	{}	{Review Claim}
342	1	Review Claim	-	00:18:00	{Lodge Claim}	{Appoint Assessor}
343	2	Lodge Claim	Theft	00:05:00	{}	{Review Claim}
344	2	Review Claim	-	00:15:00	{Lodge Claim}	{Appoint Assessor}
...	...	...	...	...	...	...

Binding			
ID	Start Time	Complete Time	Resource
341	10/06/13 09:31:00	10/06/13 09:39:00	IC1
342	10/06/13 09:42:00	10/06/13 10:00:00	IC5
343	10/06/13 09:45:00	10/06/13 09:50:00	IC1
344	10/06/13 09:55:00	10/06/13 10:10:00	IC3
...	...	...	...

The combination of an abstract event log and a binding forms a *concrete event log*. Table II illustrates the abstract event log and the binding that corresponds to the event log in Table I. By changing the information in the bindings, alternative execution scenarios can be produced.

4) Safe Bindings: A binding is considered **safe** iff:

- a resource works on at most one activity at one point in time;
- a resource involved in the execution of an activity must be allowed or authorised to perform it (resource authorisations are defined in the cost structure); and
- the order of the activities within a case is preserved (activities belonging to different cases may be reordered).

A **violation** is a breach of any of the rules above. An **unsafe** binding contains one or more violations, and a violation cost function is used to penalise these unsafe bindings.

5) Resource Utilisation: Resource utilisation is defined as the time where the resource is busy/working on an activity within a specified time frame. The horizon specifies the time frame that is used to compute the resources' utilisations and is defined within the cost structure. If the binding is safe, all resource utilisations will be between zero and one. The resources' cost rates are determined based on their utilisation.

6) Cost Structure: A cost structure represents a generic data model that stores a set of cost functions for process-related cost computations. Organisations can define functions to calculate the cost of a case, the cost of an activity, and the resource utilisation cost. To compute the total cost of a concrete event log, the cost of cases, activities, and resources are added together. The properties that contributes to the three cost functions are listed below:

- Case costs are computed by taking into account case durations and other case properties;
- Activity costs are calculated by aggregating the cost of properties such as the tasks, the resource-task combination, the activity-related durations, and the activity-related properties; and
- Resource costs are computed based on the cost of resources, the cost-rate (per duration) of resources, and the cost-rate of resources for a certain utilisation rate.

TABLE III

TASKS AND THE ROLES THAT ARE ALLOWED TO PERFORM THEM.

Roles	Allowed Tasks
Insurance Clerks (IC)	Lodge Claim, Review Claim, Appoint Assessor, Request Assessment
Insurance Assessors (IAss)	Assess Car, Assess Customer
Insurance Managers (IM)	Decide Claim (Allowed to execute all tasks, although undesired)
Insurance Accountants (IAcc)	Approve, Reject

This cost structure is used as a basis of the objective function within the proposed genetic algorithm-based solution.

A simplified car insurance claim process is used as a running example. The process is simulated using CPN Tools in order to obtain an event log. Fig. 2 depicts a BPMN model illustrating the car insurance claim process.

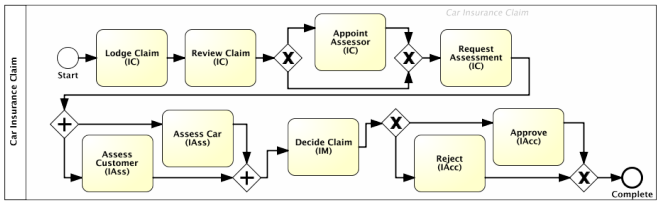

Fig. 2. A BPMN model illustrating the car insurance claim process.

The process consists of 9 tasks and 14 resources that are categorised into 4 roles. The CPN model was designed in such a way that resources do not perform more than one activity at a time, and each task can only be executed by certain role(s) (group of resources). Table III describes the roles that are allowed to perform the respective tasks.

Table IV illustrates some examples of the defined cost functions which are elaborated below:

- The cost of a case is calculated based on its case properties and its duration. An additional Service Level Agreement (SLA) has been specified, in which cases that ran overtime are penalised.
- The cost of an activity is calculated based on each activity's resource allocation. A higher cost is incurred if an inappropriate/less-desired resource executes that activity. For example, a manager performing an activity yields a higher cost than a clerk.
- Resource costs are calculated based on resource utilisation. The per minute cost rate is determined by the resource utilisation for the specified time horizon. An

assumption for this example is that the desired resource utilisation is 0.8. The cost rate is set in such a way that a resource's utilisation between 0.75 and 0.85 within the hour is cheapest. Likewise, under- or over-utilisation of resources results in a high cost rate.

IV. Genetic Algorithms-Based Optimisation

The aim of this research is to identify less expensive execution scenarios by exploring different possible bindings for a given event log. We apply a genetic algorithm to facilitate the construction and exploration of the massive search space. Genetic algorithms use the principles of evolution to guide the search. In this case, special-purpose crossover and mutation operators are applied to a population of bindings, which is subsequently evaluated according to reductions in cost. A genetic algorithm-based approach was opted for because of its flexibility and adaptability, along with robust performance and global search characteristics [25]. We deem these characteristics necessary because the NP-hard optimisation problem at hand is non-linear, high-dimensional, and prone to many local optima. The non-linear nature discourages the use of LP or ILP techniques, while the high-dimensionality makes the use of a brute force or Monte Carlo-inspired approach impracticable. Furthermore, due to the many local optima, heuristic approaches and simulated annealing are deemed less suitable as they tend to be more prone to converge to such suboptimal solutions [30]. Therefore, it is argued that a genetic algorithm-based approach is the most adequate technique for our problem.

A. Operators

A safe variant and unsafe variant have been designed for each of the operators. The safe variants ensure that only safe bindings are produced, whereas the unsafe variants allow unsafe bindings to be generated and brought forward to the next generation. The list of operators are:

1) **Crossover**. The crossover operator cross-breeds the properties (activity start time and resource allocation) of a selected number of activities between two bindings. A crossover point is picked randomly from the parent binding's list of activities, and a specified number of activity crossovers (*crossover frequency*) is applied, where the activity's start time and resource allocation are swapped between two parent bindings. Fig. 3 illustrates how crossovers are performed. For the safe crossover variant, if the crossover produces safe bindings, the safe bindings are kept and brought forward to the next generation. If not, the bindings are discarded, and the parent bindings are brought forward to the next generation instead. For the unsafe variant, the bindings produced are brought forward to the next generation regardless of whether they are safe or not.
2) **Time Mutation**. The time mutation operator changes the start times of a selected number of activities within a binding individual. For a number of cases (*case frequency*), a number of activities (*activity frequency*) are selected to have their start times altered. A new start time is chosen between the activity's possible minimum and maximum start times. The safe time mutation variant checks the chosen start time for potential violations of the activity order and the resource allocation authorisation. The activity's start time is mutated if it does not result in violations. If the mutation will result in an unsafe binding, no mutation is performed. The unsafe variant mutates the activity's start time regardless of its potential for violations.
3) **Resource Mutation**. The resource mutation operator swaps the resource that is executing an activity. For a random number of cases (*case frequency*), a random number of activities (*activity frequency*) are selected to have their resource allocation mutated. For the safe resource mutation variant, a different resource is randomly picked from the pool of *idle* resources for mutation. If there is no resource available, no mutation is performed. The unsafe mutation variant randomly picks a new resource from the pool of *all* resources.
4) **New Heuristic Binding**. This operator introduces a new safe binding into the population. Although typically applied with a low probability, this operator reduces the chance of the algorithm confining itself to a local optimum neighbourhood. For each case (randomly ordered), the earliest possible start time is proposed for an activity, and the algorithm attempts to identify a resource that is suitable (allowed to execute the activity) and available during the activity's proposed execution time frame. If there is a suitable resource, the activity is scheduled with the proposed start time and resource allocation. If no suitable resource is found, a new start time is proposed again (based on previously scheduled activities), and the resource allocation process is repeated. This process is iterated until all activities have been scheduled. As this operator ensures the generation of a safe binding, therefore there is no need for an unsafe variant.

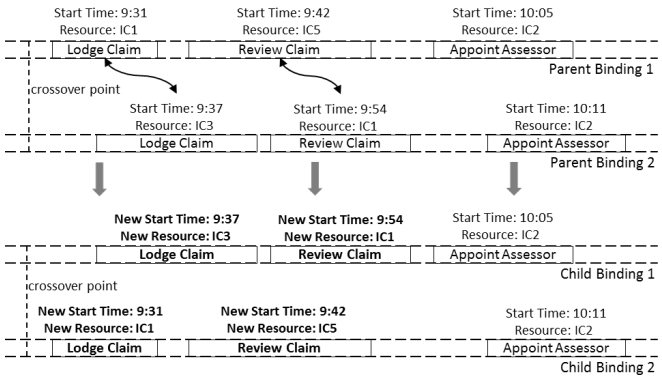

Fig. 3. How crossovers are performed (crossover frequency = 2).

There is a probability that unsafe operators will fail to produce safe bindings. Hence, a **repair function** was introduced. Unsafe bindings are repaired by re-assigning activity start times, while preserving activity order and resource allocation. For each activity in a case, an earliest possible start time

TABLE IV

EXAMPLE OF COST FUNCTIONS DEFINED IN THE COST STRUCTURE.

Cost Type	Property	Value	Cost Rate
Case	Damage Type & Case Duration	Windscreen	\$4 per minute
	Case Duration	-	\$2000 if it takes more than 3 hours, and \$200 for every subsequent hour after that
Activity	Activity & Resource	Lodge Claim + IC1	\$10 per invocation
	Activity & Resource	[Over-qualified Resources]	\$1000 per invocation
Resource	Resource Utilisation	Between 0 and 0.15 (under-utilised)	\$45 per minute
	Resource Utilisation	Between 0.75 and 9.85 (optimum utilisation)	\$1 per minute
	Resource Utilisation	Higher than 0.9 (over-utilised)	\$20 per minute

is proposed and checked. If the resource is already utilised during the proposed time frame, a new start time for the activity is proposed (based on previously scheduled activities), and the checking process is repeated. All activities in all cases iteratively undergo this process until all activities have been scheduled. Additional safe-after-repair variants have been introduced for the crossover, the time mutation, and the resource mutation operators. This allows more crossovers and mutations to happen.

The genetic algorithm is sensitive to the frequency of the operators being applied. The frequencies for the operator variants can be found in Table V. Safe crossover has a really low success rate due to the difficulty in obtaining safe bindings. Although a higher frequency will increase the number of safe bindings generated, it is kept low for computational reasons. The time mutation operators are designed in such a way that activities are rescheduled to start earlier if there are idle resources. This will have a positive impact on the cost, therefore the high time mutation frequency. However, there is a chance of activities being scheduled to start slightly later as well. This allows more activities to be scheduled earlier between those "time gaps", avoiding the possibility that cases or activities are being delayed substantially until resources free up. Due to the nature of the roles and resources in the business process used for the experiment (where Insurance Managers are underutilised and the rest of the resources are overutilised), the resource mutation tends to allocate the managers to perform activities for which they are overqualified. Even though this levels out the utilisation of resources, it results in higher activity costs. Hence, a low resource mutation frequency is enforced. The application frequency for the "New Heuristic Binding" operator is set fairly low in order to avoid a majority of the binding population being replaced.

B. Genetic Algorithm Variants

Four genetic algorithm variants were put together based on the developed operators. Variants 1 - 3 only produce safe bindings, whereas variant 4 ignores the safeness of the bindings. The new heuristic binding operator is used in every algorithm variant to reduce the probability of the algorithm confining itself to a local optimum neighbourhood.

1) **Variant 1, Safe**. Operators within this variant will always generate safe bindings. Otherwise, unsafe bindings will be discarded or mutations that will result in violations are skipped. This variant consists of the *safe crossover*, *safe time mutation*, *safe resource mutation*, and *new heuristic binding* operators.
2) **Variant 2, Safe after Repair**. Bindings will be crossovered or mutated whilst ignoring the safeness requirement. The repair function is then applied to obtain safe bindings. Operators in this variant includes *safe after repair crossover*, *safe after repair time mutation*, *safe after repair resource mutation*, and *new heuristic binding*. This variant is introduced to indirectly allow a higher success rate of the safe operators.
3) **Variant 3, Heuristic**. The heuristic approach only uses the *new heuristic binding* operator. New safe bindings are generated where activities are scheduled to start as soon as possible if the authorised resources are idle.
4) **Variant 4, Unsafe**. The operators in this variant do not enforce the safeness requirement of the bindings. Operators in this variant includes *unsafe crossover*, *unsafe time mutation*, *unsafe resource mutation*, and *new heuristic binding*. Violation costs are introduced in the fitness function to enable the genetic algorithm to penalise unsafe bindings, so that they are less likely to be selected and proceed to the next generation.

V. EXPERIMENTS

This section first describes the experimental setup used to evaluate the effectiveness of a genetic algorithm-based approach which is then followed by a detailed discussion about the experimental results. Four genetic algorithm variants described in the previous section are benchmarked against a randomised approach where new bindings are randomly generated with randomised activity start times and randomised resource allocations. A number of log variants are also used in this experiment.[1] [2]

1) Log Variants: Three (simulated) log variants of the car insurance claim process are used to evaluate the performance of the different algorithm variants (see Table VI). The first log variant contains 100 cases with an average of 7 to 8 activities per case. The second variant has double the number of cases (by doubling the case arrival rate) compared to the first variant. The third log has double the number of activities in a case (by repeating the process again) compared to the first variant. The goal of the introduction of two alternative logs is to evaluate the cost optimisation approach against logs where, 1) the number of cases is higher and the arrival rate is

[1]QUT's High Performance Computing (HPC) facility was used to run these experiments. `http://www.itservices.qut.edu.au/researchteaching/hpc/hw_catalogue.jsp`

[2]Files can be obtained via `http://yawlfoundation.org/cost/logbasedcostanalysisandimprovement.html`.

TABLE V

FREQUENCIES FOR DIFFERENT OPERATOR VARIANTS.

Operator Variants	Safe Operators	Safe after Repair Operators	Unsafe Operators
Crossover Frequency	1% of total activity count	Poisson (50% of total activity count)	Poisson (20% of total activity count)
Time Mutation (Case Frequency)	Poisson (100% of total case count)	Poisson (10% of total case count)	Poisson (100% of total case count)
Time Mutation (Activity Frequency)	Discrete (0% to 100% of total activity count)	Discrete (0% to 100% of total activity count)	Discrete (0% to 100% of total activity count)
Resource Mutation (Case Frequency)	Poisson (10% of total case count)	Poisson (5% of total case count)	Poisson (10% of total case count)
Resource Mutation (Activity Frequency)	Discrete (0% to 100% of total activity count)	Discrete (0% to 100% of total activity count)	Discrete (0% to 100% of total activity count)
New Heuristic Binding (Chance Frequency)	5% of total population count		

Poisson (*mean*) = pick a random value from a Poisson distribution with the stated mean.
Discrete (*min* to *max*) = pick a random value from a discrete uniform distribution between the stated minimum and maximum values.

increased, and 2) the number of activities in a case is higher while the number of resources remains the same. Different cost structures are also defined for each log variant.

TABLE VI

CHARACTERISTICS OF THE THREE EVENT LOG VARIANTS.

Log Variant & Name	No. of Cases	No. of Tasks	No. of Resources/Roles
1 - Normal	100	9	14 resources, 4 roles
2 - Double Cases	200	9	14 resources, 4 roles
3 - Double Tasks	100	18	14 resources, 4 roles

A. Experiment Parameters and Configurations

A number of parameters for this experiment have been fixed across different log and algorithm variants. The parameters are fixed in consideration of the search space and the experiment's feasibility.

- Population Size: 50
- Time Discretisation: Yes (True)
- Time Block: 1-minute blocks
- Elite Count: 5% of the total population count
- Resource Utilisation Horizon: 1 hour (resource utilisation rate is calculated for the past hour)
- Selection Strategy: Tournament Selection (with a 75% probability that a fitter individual will be selected to undergo crossover and mutation)
- Initial Population's Bindings: 100% safe bindings

The experiment parameters above are fixed for all experiment variants except for algorithm variant 4 (unsafe variant). Due to that algorithm variant ignoring the safeness requirement, the initial population can be generated without ensuring the safeness of the bindings, reducing computational time. Each violation occurrence within the unsafe bindings are then penalised with a violation cost to discourage unsafe execution scenarios. Example cost structures and their respective violation costs were developed for the three different event logs that were generated.

B. Result Analysis and Discussion

Statistics gathered from the experiments are summarised in Table VII. A five-fold average has been computed across the log variants. All variants discovered cheaper alternative execution scenarios, although different algorithms achieve different rates of cost reduction.

In terms of the time taken for each algorithm variant to complete 500 evolution generations (elapsed time), variant 4 (unsafe) performs the fastest across all log variants. However, the cost reduction is less evident compared to the rest of the algorithm variants, especially algorithm variant 1 (safe). Also, algorithm variant 1 (safe) is more scalable compared to other algorithm variants, which can also be observed across all log variants.

Taking a closer look at the overall cost, where a reduction in cost is indicated with a negative percentage, it can be observed that the execution scenario identified by algorithm variant 1 (safe variant) has the lowest cost, followed by algorithm variant 3 (the heuristic variant). This observation holds across the three log variants. For log variants 2 (double cases) and 3 (double tasks), the cost improvement realised by all algorithm variants plateaued between -39% and -46%. Due to the limited number of resources and the high number of activities, the process can only improve so much, while not compromising utilisation of the resources. In addition, each of the algorithm variants successfully reduces cost for the different cost types. However, the extent of reduction for each cost type varies. For instance, we can observe that algorithm variant 3 (heuristic) can reduce resource costs significantly when compared with other variants. This can be explained by how the heuristic algorithm works, as it tries to allocate resources to activities as soon as they are available, lowering the resource costs.

In addition, several non-cost indicators were used as measurements. The average waiting time (AWT) between activities within the case is calculated, where a reduction in AWT will result in an increase in resource utilisation and a reduction of SLA breaches, therefore bringing down the cost. For each log variant, the number of cases that breached the pre-defined SLA deadlines are aggregated, where a reduction in SLA breach count typically means a reduction in cost. All algorithm variants not only considerably reduce the average waiting time and SLA breach count, but also increase the resource utilisation, which in turn reduces overall cost. It can also be observed that algorithm variant 3 (heuristic), in particular, performs better in reducing AWT and in increasing the resource utilisation. This is, again, due to the way the algorithm works — by allocating resources to activities as soon as possible. For the experiments that ran using log variant 3 (double tasks), only a slight increase and even a decrease in resource utilisation can be observed. The reason for this is that only a limited number of resources are executing a high number of activities, resulting in a situation where most of the resources are overutilised originally. Cost reductions can be achieved by a slight adjustment of resource utilisations, where high cost caused by resource under- or over-utilisation are lowered by mediating the utilisation of resources. This demonstrates the complexity of cost-based

TABLE VII

EXPERIMENT OUTCOMES FOR ALL LOG VARIANTS USING DIFFERENT ALGORITHM VARIANTS

Log Variants	Log Variant 1				Log Variant 2				Log Variant 3			
Algorithm Variants	**Variant 1 (Safe)**	**Variant 2 (Repair)**	**Variant 3 (Heuristic)**	**Variant 4 (Unsafe)**	**Variant 1 (Safe)**	**Variant 2 (Repair)**	**Variant 3 (Heuristic)**	**Variant 4 (Unsafe)**	**Variant 1 (Safe)**	**Variant 2 (Repair)**	**Variant 3 (Heuristic)**	**Variant 4 (Unsafe)**
Elapsed Time	0:48:20	3:30:06	1:08:51	**0:38:37**	1:46:34	21:27:35	5:44:59	**1:37:00**	1:13:33	16:28:24	3:33:19	**0:55:16**
Execution Cost	**-31.61%**	-12.62%	-15.15%	-12.32%	**-44.97%**	-39.56%	-41.08%	-39.74%	**-45.87%**	-41.77%	-43.54%	-41.88%
Case Cost (C)	-42.97%	-41.92%	-0.29%	-7.83%	-33.61%	-32.96%	-0.13%	-7.63%	-34.14%	-35.52%	-0.58%	-4.51%
Activity Cost (A)	-51.45%	-10.50%	-48.67%	-15.98%	-24.13%	-11.15%	-48.88%	-9.18%	-26.00%	-15.82%	-51.25%	-12.70%
Resource Cost (R)	-10.94%	-11.78%	-67.80%	-20.12%	-24.73%	-14.69%	-82.29%	-11.54%	-21.47%	-12.56%	-83.07%	-12.75%
Average Waiting Times (AWT)	-78.33%	-78.19%	**-80.71%**	-79.81%	-52.30%	-54.88%	**-54.92%**	-55.49%	-52.11%	-53.63%	**-54.32%**	-52.06%
SLA Breach Count	**-85.20%**	-80.80%	-83.20%	-84.80%	**-53.02%**	-50.79%	-47.30%	-47.94%	-65.92%	-65.31%	**-69.39%**	-64.49%
Resource Utilisation	14.33%	20.91%	**23.42%**	20.95%	16.99%	24.72%	**26.67%**	21.65%	-8.18%	1.85%	**5.60%**	1.29%

trade-offs. Considering the emphasis on cost reduction, and the balance between all the non-cost indicators and elapsed time, it can be concluded that algorithm variant 1 (safe) exhibited the best results.

From an efficiency gain perspective, Fig. 4 illustrates the cost of new bindings generated for log variant 1 (normal) over 500 evolution generations using algorithm variant 1 (safe). We can observe the reduction in cost across different cost types as the number of evolutions continues. The new binding resulted in a cost reduction of $249,550.00 when compared with the original binding, which has a total cost of $717,021.00 (a reduction of 34.8%), which indicates that there is room for efficiency gains if we can learn from the past to determine how the cost can be reduced.

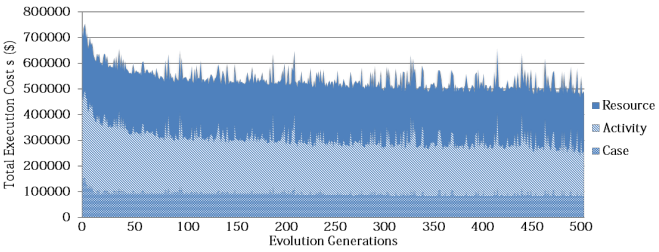

Fig. 4. Total cost reduction of log variant 1 (normal) over 500 evolution generations using algorithm variant 1 (safe).

We now turn our attention to the resource efficiency metrics. Fig. 5 compares the utilisation rate of a resource - Insurance Assessor 2 (IAss2) within log variant 1 (normal) using algorithm variant 1 (safe). In the example cost structure, under/over utilisation of a resource is discouraged and this is reflected in a higher cost rates for such cases than in the optimal case. As a result, the graph shows that under- and overutilisation of IAss2 has been discouraged, by increasing and levelling the utilisation of IAss2. The increase in resource utilisation also resulted in reduced waiting times for cases and increased process efficiency, reducing the overall log duration by roughly 30%.

Fig. 6 compares the activities' timestamps, the overall case duration, and the total waiting time for case number 10, within log variant 1 (normal) before and after the experiment, where algorithm variant 1 (safe) was used. It can be observed that not only the overall case duration has decreased, the total waiting time has been reduced too, completing the case before the SLA deadline is due. In addition, the total duration that resources that are utilised between 50% to 85% of the time during the past hour, has increased.

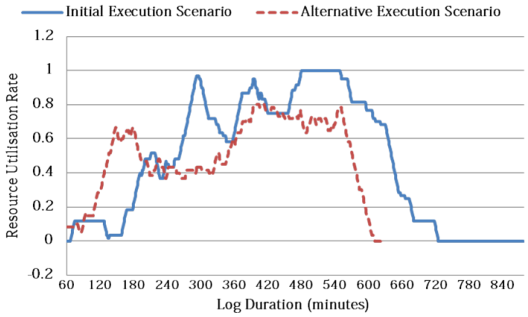

Fig. 5. Comparison of utilisation rates of resource Insurance Assessor 2.

In addition, an approach that explores possible execution scenarios randomly was run for a week as a benchmark. Due to the huge amount of unsafe candidates in the search space, the approach failed to identify a better execution scenario within the given time frame. Even worse, in Table VIII, cost increases are observed across all log variants. Hence, a randomised approach seems unable to effectively and efficiently explore execution scenarios that will incur lower cost due to the complexity and the large search space of this problem.

TABLE VIII

EXPERIMENT OUTCOMES USING THE RANDOMISED APPROACH ON ALL LOG VARIANTS FOR ONE WEEK.

Log Variant	Log Variant 1	Log Variant 2	Log Variant 3
Evolution Generations	241291	78696	88112
Execution Cost	150.27%	279.82%	94.98%

In this section, we discussed the approach undertaken to evaluate the proposed genetic-algorithm based cost optimisation environment with different log variants. These preliminary results show that it is possible to learn from the history by generating alternative scenarios to satisfy the goal of cost minimisation. Of course, it is not yet possible to generalise these results across processes with different characteristics (e.g., the number of activities, case arrival rates, the number of resources) and with different cost structures.

VI. CONCLUSION

This paper proposes a novel cost-informed process improvement approach that enables the generation and comparison of alternative process execution scenarios while taking into account trade-offs in terms of cost. This approach is based on the identification of the fixed and variable parts

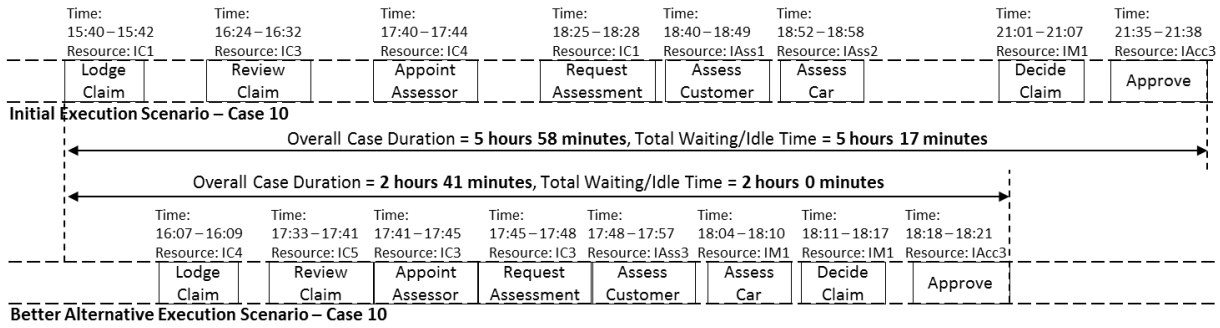

Fig. 6. Comparison between the initial execution scenario and a better alternative execution scenario for case 10.

of an event log. On top of this framework, a comprehensive and adaptive cost structure that captures different cost-related dimensions has been proposed and incorporated as the objective function. Finally, the optimisation is realised with the implementation of a set of genetic algorithm variants. Observe that in our approach, the overall cost of a business process (based on the process behaviour represented in a log) is computed, not the cost of individual cases.

In future work, cost-related insights can be derived by learning from the improved history and recommendations for cost reductions can be put forward. It is also possible to investigate techniques to better visualise the generated outcomes, alongside potential improvements to the algorithms. Moreover, an actionable methodology for identifying and advocating for significant business process improvements in organisations based on concrete cost reduction insights could be explored.

REFERENCES

[1] S.P. Brooks and B.J.T. Morgan. Optimization using simulated annealing. *The Statistician*, 44(2):241–257, 1995.

[2] D.E. Goldberg. *Genetic algorithms in search, optimization, and machine learning*. Addison-Wesley Professional, 1989.

[3] J.F. Gonçalves, J.J.M. Mendes, and M.G.C. Resende. A genetic algorithm for the resource constrained multi-project scheduling problem. *European Journal of Operational Research*, 189(3):1171–1190, 2008.

[4] A. Greasley. Using business-process simulation within a business-process reengineering approach. *BPM Journal*, 9(4):408–420, 2003.

[5] A. Greasley and S. Barlow. Using simulation modelling for BPR: resource allocation in a police custody process. *International Journal of Operations & Production Management*, 18(9/10):978–988, 1998.

[6] S. Hartmann and D. Briskorn. A survey of variants and extensions of the resource-constrained project scheduling problem. *European Journal of Operational Research*, 207(1):1–14, 2010.

[7] I. Hofacker and R. Vetschera. Algorithmical approaches to business process design. *Computers & Operations Research*, 28(13):1253–1275, 2001.

[8] Z. Huang, X. Lu, and H. Duan. Resource behavior measure and application in business process management. *Expert Systems with Applications*, 39(7):6458–6468, 2012.

[9] Z. Huang, W.M.P. van der Aalst, X. Lu, and H. Duan. Reinforcement learning based resource allocation in business process management. *Data & Knowledge Engineering*, 70(1):127–145, 2011.

[10] H.C. Hwang and B.K. Choi. Workflow-based dynamic scheduling of job shop operations. *International Journal of Computer Integrated Manufacturing*, 20(6):557–566, 2007.

[11] M.H. Jansen-Vullers, P.A.M. Kleingeld, M.W.N.C. Loosschilder, M. Netjes, and H.A. Reijers. Trade-offs in the performance of workflows–quantifying the impact of best practices. *Business Process Management Workshops*, 4928:108–119, 2008.

[12] M.H. Jansen-Vullers, M. Netjes, H.A. Reijers, and M.J. Stegeman. A redesign framework for call centers. In *Business Process Management*, pages 306–321, 2006.

[13] Akhil Kumar, Remco M. Dijkman, and Minseok Song. Optimal resource assignment in workflows for maximizing cooperation. In *Business Process Management*, pages 235–250, 2013.

[14] L.J. Li, J. Gao, K. Chen, and H. Jiang. The identification of irrationally allocated resources in business process based on network centrality analysis. *International Journal of Computer Integrated Manufacturing*, 24(8):748–755, 2011.

[15] S.L. Mansar and H.A. Reijers. Best practices in business process redesign: validation of a redesign framework. *Computers in Industry*, 56(5):457–471, 2005.

[16] S.L. Mansar and H.A. Reijers. Best practices in business process redesign: use and impact. *Business Process Management Journal*, 13(2):193–213, 2007.

[17] M. Netjes. *Process Improvement: The Creation and Evaluation of Process Alternatives*. PhD thesis, Eindhoven University of Technology, 2010.

[18] F. Niedermann, A. Pavel, and B. Mitschang. Beyond roles: Prediction model-based process resource management. In *Business Information Systems Workshops*, pages 5–17. Springer, 2011.

[19] H.A. Reijers and S.L. Mansar. Best practices in business process redesign: an overview and qualitative evaluation of successful redesign heuristics. *Omega*, 33(4):283–306, 2005.

[20] H.A. Reijers and K.M. van Hee. Product-based design of business processes applied within the financial services. *Journal of Research and Practice in Information Technology*, 34(2):110–122, 2002.

[21] A.H.M. ter Hofstede, W.M.P. van der Aalst, M.Adams, and N.Russell, editors. *Modern Business Process Automation - YAWL and its Support Environment*. Springer, 2010.

[22] W.M.P. van der Aalst. Petri net based scheduling. *Operations Research Spektrum*, 18(4):219–229, 1996.

[23] W.M.P. van der Aalst. *Process Mining: Discovery, Conformance and Enhancement of Business Processes*. Springer, 2011.

[24] K. Vergidis and A. Tiwari. Business process design and attribute optimization within an evolutionary framework. In *IEEE Congress on Evolutionary Computation*, pages 668–675. IEEE, 2008.

[25] K. Vergidis, A. Tiwari, and B. Majeed. Business process analysis and optimization: beyond reengineering. *IEEE Trans. Syst., Man, Cybern. C, Appl. Rev.*, 38(1):69–82, 2008.

[26] W.L. Winston and J.B. Goldberg. *Operations research: applications and algorithms*. Thomson/Brooks/Cole Belmont, 2004.

[27] J. Xu, C. Liu, and X. Zhao. Resource planning for massive number of process instances. In *On the Move to Meaningful Internet Systems: OTM 2009*, pages 219–236. Springer, 2009.

[28] J. Xu, C. Liu, X. Zhao, and Z. Ding. Incorporating structural improvement into resource allocation for business process execution planning. *Concurrency and Computation: Practice and Experience*, 2012.

[29] J. Xu, C. Liu, X. Zhao, and S. Yongchareon. Business process scheduling with resource availability constraints. In *On the Move to Meaningful Internet Systems*, pages 419–427. Springer, 2010.

[30] X. Yang. *Introduction to Mathematical Optimization*. Cambridge International Science Publishing, 2008.

[31] M. zur Muehlen and D.T. Ho. Service process innovation: a case study of BPMN in practice. In *Hawaii international conference on system sciences, proceedings of the 41st annual*, pages 372–372. IEEE, 2008.