Computing manuscript No.
(will be inserted by the editor)

Revising History for Cost-Informed Process Improvement

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

Received: date / Accepted: date

Abstract Organisations are constantly seeking new ways to improve operational efficiencies. This study investigates a novel way to identify potential efficiency gains in business operations by observing how they were 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 these trade-offs can be incorporated in the assessment of alternative process execution scenarios by making use of a cost environment. A number of optimisation techniques are proposed to explore and assess alternative execution scenarios. The objective function is represented by a cost structure that captures different process dimensions. An experimental evaluation is conducted to analyse the performance and scalability of the optimisation techniques: integer linear programming (ILP), hill climbing, tabu search, and our earlier proposed hybrid genetic algorithm approach. The findings demonstrate that the hybrid genetic algorithm is scalable and performs better compared to other techniques. Moreover, we argue that the use of ILP is unrealistic in this setup and cannot handle complex cost functions such as the ones we propose. Finally, we show how cost-related insights can be gained from improved execution scenarios and how these can be utilised to put forward recommendations for reducing process-related cost and overhead within organisations.

Keywords Business Process Analysis · Business Process Improvement · Process Mining · Optimisation · Cost-Informed · Genetic Algorithm

Mathematics Subject Classification (2000) 68U01 · 68U35

W. Z. Low · M. T. Wynn · A. H. M. ter Hofstede
Queensland University of Technology (QUT), Brisbane, Australia
E-mail: w4.low@qut.edu.au

S. K. L. M. vanden Broucke · J. De Weerdt
KU Leuven, Belgium

W. M. P. van der Aalst
Technische Universiteit Eindhoven (TU/e), The Netherlands

1 Introduction

Within the field of business process management, the concept of business process redesign and improvement is highly relevant to researchers and practitioners. 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 [14]. Approaches ranging from Six Sigma [19], Lean Thinking [34], and Kaizen [16], to Value-Based Management (VBM) [15], Economic Value Added (EVA) [21], and value driver trees [20], are constantly being employed by organisations to conduct business process improvement and redesign activities. Process mining facilitates the identification of improvement opportunities by providing techniques to discover, monitor, and improve processes by extracting knowledge from business process histories [27]. By obtaining 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 (for example, can process instances be completed faster? Can operational cost be reduced? Can the quality of the outcomes be improved?).

Business processes are often evaluated using the devil's quadrangle, where time, cost, quality, and flexibility are considered as improvement dimensions [3]. Hence, business process redesign and improvement activities are often thought to have an overlapping interest with operations research, as operations research is interested in maximising (profit, productivity, or utilisation) or minimising (cost or risk) objectives of real-world problems [33]. Within an organisation, there may be trade-offs between these dimensions, making it impossible to maximise all four dimensions [3]. It is then up to organisations to determine which dimension(s) should be the focus of business process redesign and improvement, with (either positive or negative) consequences on other dimension(s) of the quadrangle. Various means to gain insights from the time and resource perspectives have been looked into [11]. Cost reduction has also been the objective of several investigations [22,37]. However, there is a lack of multi-objective business process improvement initiatives that take into account different execution costs and their respective trade-offs.

This paper proposes a cost-informed technique to intelligently search for alternative business process execution scenarios with the aim of "Improving the history", taking into account trade-offs between time, cost, and resource utilisation. The main objectives of this work are 1) to discover execution scenarios that are cheaper (better) than the original (baseline) scenario to gain insights for future redesign activities, 2) to propose a generic, adaptable cost structure, and 3) to evaluate the performance and scalability of the proposed optimisation techniques. The approach of this research is inspired by the Design Science Research Methodology, where the emphasis is placed on design and development of artefacts to evaluate and improve their functionality [10]. 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 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 process instances) while other elements within the log (resource allocations and start times of activities) are adjusted to explore different execution scenarios. By making use of a generic and adaptable cost structure, the cost of various execution scenarios can be computed and compared.

The contribution of this paper is fourfold. One, we advocate a cost-informed process improvement approach that splits the event log into two parts (fixed and variable parts) and search for alternative execution scenarios by perturbing the variable part. Two, a cost structure is introduced to cater for different cost trade-offs. Three, several optimisation techniques have been introduced and formalised as an attempt to solve the optimisation problem. Optimisation strategies were defined to explore cost-optimal execution scenarios that take into account trade-offs from multiple process dimensions. Finally, an experimental evaluation is carried out to assess the performance and scalability of the proposed techniques in facilitating the exploration of different execution scenarios. This paper extends our previous work [13] in some areas:

– In addition to the proposed hybrid genetic algorithm, three additional optimisation techniques are investigated, which are integer linear programming (ILP), hill climbing, and tabu search.
– The fundamentals (event log, resource utilisations, and many more) and the cost structure used in our previous work [13] are formalised.
– The concept of working hours is introduced and formalised, leading to a more realistic involvement of resources.
– A more complex motivating example, inspired by a real-world business process in the insurance domain, is presented.
– A new set of experiments is performed, comparing the performance and scalability of the proposed optimisation techniques against our hybrid genetic algorithm [13].

The remainder of this paper is organised as follows. Related work is reviewed and discussed in Section 2. In Section 3, a cost-informed process improvement approach is proposed and illustrated through a motivating example. Next, the proposed optimisation techniques are discussed in Section 4. Section 5 discusses the experimental results and the limitations. Lastly, section 6 concludes this paper and states potential future work.

2 Related Work

The research agenda on cost-aware business process management [36] advocated for the full integration of a cost perspective in all phases of the business process management life cycle, enabling the management of business process operation costs in a structured manner. Vom Brocke et al. [31] proposed an information model to link the ARIS accounting structure with Event-Driven Process Chains (EPCs) during the process design stage. A process accounting model (PAM) was proposed to integrate and structure both accounting and process data to support the design, execution, and control of business processes [23], in line with the push for cost perspective integration into business process management lifecycle phases. In our earlier work [35], a conceptual data model for cost model was designed, informed by requirements gathered from domain experts and a literature study of costing techniques such as activity-based costing (ABC), time-driven activity-based costing (TDABC), and resource consumption accounting (RCA). These works aim to provide organisations with better cost insights and cost control over their business processes, potentially reducing the cost of their operations.

Operations research encompasses a wide range of problem-solving techniques to enhance both decision-making ability and efficiency. These techniques include, but are not limited to, integer linear programming (ILP) [18], hill climbing [1], tabu search [7], simulated annealing [4] and genetic algorithms [8]. Each technique has a different approach towards a given optimisation problem, where it depends on aspects such as how the search space is explored and how the accumulated search experience is exploited [2]. Table 1 compares a number of optimisation techniques based on features that characterise the two aspects mentioned above. In this work, a detailed investigation of four optimisation techniques, namely, ILP, hill climbing, tabu search, and (hybrid) genetic algorithms is conducted. ILP is chosen due to its ability to provide a good and quick solution for complex optimisation problems [18]. A heuristic technique (hill climbing) is also considered for its speed and scalability when a problem has many solutions [1]. Tabu search is selected due to its relatively fast speed and its avoidance of retracing its previous steps [7]. Finally, a genetic algorithm guided by a set of heuristics is being considered for its robust performance and global search properties [8]. In comparison, slow computation of simulated annealing [4], ant colony's uncertainty in convergence time [5], as well as particle swarm's inability to work out the problems of scattering and optimisation [6], are among the reasons why these techniques did not become our first choice of investigation.

Scheduling problems are among the key problems in operations research. These problems emphasise on the optimisation of resource allocation and utilisation. Some well-known scheduling problems include shop scheduling problems [12] and resource-constrained (multi-) project scheduling problems (RC(M)PSP) [9]. Scheduling problems are similar to business process optimisation problems, as both aims to optimise the allocation of resources to tasks [26]. However, scheduling optimisation solutions are developed to solve targeted problems, and hence cannot easily be applied to other fields. As pointed out by Verigidis et al. [28], business processes involve additional components such as business rules and decisions that are hard to express mathematically, and may not be covered by scheduling problems.

Despite this, a number of studies attempted to apply scheduling techniques to optimise business processes in terms of time, cost, and resource utilisation. Within the area of optimising business process modelling and design, Verigidis et al. [30] reviewed the topics of business process modelling, analysis, and optimisation. Wang et al. [32] developed an evolutionary, multi-objective

Table 1: Features and the characteristics of common optimisation techniques (adapted from [2]).

Techniques / Features	Hill Climbing [1]	Tabu Search [7]	Genetic Algorithm [8]	Simulated Annealing [4]	Ant Colony [5]	Particle Swarm [6]
Trajectory Search	$\neg$	$\surd$	$\neg$	$\surd$	$\neg$	$\neg$
Population-based	$\neg$	$\neg$	$\surd$	$\neg$	$\surd$	$\surd$
Memory Usage	$\exists$	$\surd$	$\exists$	$\neg$	$\surd$	$\surd$
Multiple Neighbourhood	$\surd$	$\neg$	$\exists$	$\neg$	$\neg$	$\exists$

$\surd$ = feature is present; $\exists$ = feature is partially present; $\neg$ = feature is not present.
*This classification only serves as an indication of the techniques' characteristics and does not apply to all implementations.

framework as a first attempt to optimise business process designs. Their work tackles constrained process optimisation problems, emphasising on reducing the number of infeasible solutions generated. An evolutionary multi-objective optimisation algorithm to generate a series of diverse optimised business process designs was also proposed [29]. Tiwari et al. [25] proposed an optimisation framework that generates a number of different optimised business process designs, and they evaluated the optimisation results through an experiment. There are other works that looked into optimisation from a business process execution perspective. Yu and Buyya [40] investigated the optimisation problem of minimising the cost and execution time of workflow scheduling. This work only minimises either one of these optimisation objectives, but not both. Xu et al. [37] investigated the optimisation of flow time and cost of business processes separately and improved the process structure according to resource allocation requirements. 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 [38]. Nevertheless, the execution cost of the business process is not considered in this study. Table 2 provides a summary of authors, techniques, foci, and objectives of the related works. While most of the business process execution-related studies discussed above deal with the minimisation of cost, the minimisation of execution time, and complex resource behaviour, none of them propose an approach that takes all of these into account. In contrast to the works discussed above, this paper:

- Investigates how the execution of a business process can be improved;
- Identifies cost-optimal execution scenarios by perturbing the process history (event log) and learn from it;
- Introduces working hours for resources to enable a more accurate cost measurement; and
- Incorporates different cost-related dimensions into a cost structure (for example, activity-based cost, time-based cost, and resource utilisation), which allows the use of more sensible cost-based trade-offs as the objective function for the optimisation techniques.

Table 2: A summary of the related works.

Authors	Optimisation Technique(s)	Foci	Objective(s)
Wang et al. [32]	Genetic Algorithm	Process Design	Reduce the number of infeasible solutions generated by tailoring operators used by a genetic algorithm.
Verigidis and Tiwari [29]	Genetic Algorithm	Process Design	Generate diverse, optimised business process designs for the same process requirements.
Tiwari et al. [25]	Genetic Algorithm & Particle Swarm	Process Design	Generate alternative optimised process designs and performed an experimental evaluation.
Yu and Buyya [40]	Heuristics & Genetic Algorithm	Process Execution	Scheduling workflow applications by either minimising cost or execution time while satisfying constraints.
Xu et al. [37]	Heuristics	Process Design & Execution	Optimise resource allocation based on cost and execution time, and allow process structure to adapt the optimised resource allocation.
Xu et al. [38]	Heuristics	Process Execution	Allocate resources based on a set of heuristic rules during build time to maximise the success rate of scheduling.

3 Cost-Informed Log Perturbation

The motivation for this research is to identify a more cost-efficient execution scenario. The notion of cost is applied towards a number of efficiency measures. For example, efficiency measures such as service level agreement and resource allocation rules (for example, roles, skills, etc.) are often introduced by organisations to ensure adequate provision of services and appropriate allocation of resources. Hence, desirable desirable utilisation rates are promoted (see Definition 6 in Subsection 3.1), and unsuitable resources and service level agreement violations are discouraged (see Definition 7 in Subsection 3.1). Alternative execution scenarios are explored where a scenario with a lower cost represents a more efficient scenario.

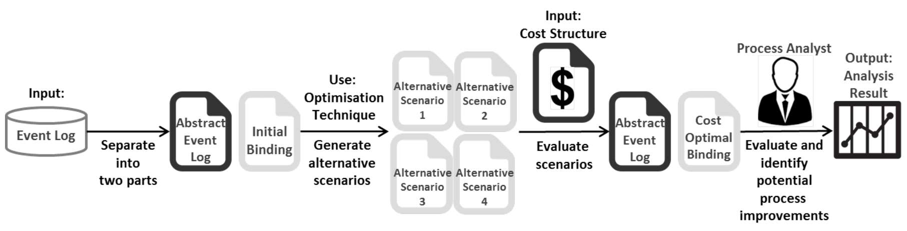


Fig. 1: Overview of the cost-informed process improvement approach.

Fig. 1 illustrates the proposed approach for the generation of cost-informed alternative scenarios. To explore different execution scenarios, the identification and separation of the fixed part and the variable part of a typical event log is the first step in this approach. An abstract event log (a detailed explanation follows below, see Definition 2) contains the fixed part of an event log, whereas a binding (detailed explanation in Definition 3) contains the variable part of an event log. Then, by using a number of optimisation techniques, different execution scenarios are explored by manipulating the variable part of the event log, in this case, the binding. The defined cost structure is used as an objective function to determine the fitness of the execution scenarios in terms of process-related cost. Note that the cost structure can be further configured and customised by organisations, taking into consideration cost-informed trade-offs between multiple aspects such as process instance durations, activity invocation, and resource allocation and utilisation. The abstract event log and binding combination result in an alternative execution scenario. The most cost-efficient alternative execution scenarios are then identified and analysed.

In this section, the fundamentals and the corresponding formalisation are discussed, followed by a motivating example.

3.1 Fundamentals

Definition 1 (Event Log) An event log is a data store that records potentially vast amounts of event-related information [24,27]. An event log consists of a collection of process instances (*cases*). For each case, there is a sequence of *activities* (commonly referred to as "work items"), where an activity is an instance of a *task*. Each activity has a *start time* and an *end time*, a *resource* that is executing the activity, as well as any additional *properties* that are related to the activity.

Table 3: A fragment of a car insurance claim event log in chronological order.

Case ID	Activity	Start Time	End Time	Resource	Property (Damage Type)	...
1	CAR	10/06/13 09:31:00	10/06/13 09:39:00	A5	Windscreen	...
1	RAR	10/06/13 09:42:00	10/06/13 10:00:00	IS2	-	...
2	CAR	10/06/13 09:45:00	10/06/13 09:50:00	A1	Theft	...
2	RAR	10/06/13 09:55:00	10/06/13 10:10:00	IS1	-	...
...	...	...	...	...	...	...

Table 3 illustrates an event log fragment of the example car insurance claim process that is used as the motivating example. As we are seeking improvements concerning execution time and resource allocation, the historical event attributes that influence timing and resource allocation are considered as the variable part (i.e. the *binding*). All other historical attributes, such as cases and activities, are kept fixed (i.e. the *abstract event log*). This includes the properties of cases and activities, as they do not affect the optimisation of process executions, hence are being kept the same (invariable).

Definition 2 (Abstract Event Log) Let $TS = \mathbb{R}$ be the set of possible timestamps, $Dur = \mathbb{R}_1^+$ the set of durations, and Val the set of all possible property values (records, tables, and many more). $L = (\mathcal{C}, \mathcal{A}, \mathcal{T}, case, task, art, dur, prop, \prec)$ is an abstract event log where:

- $\mathcal{C}$ is a set of cases,
- $\mathcal{A}$ is a set of activities (commonly referred to as "work items"),
- $\mathcal{T}$ is a set of tasks,

- the sets $\mathcal{C}$, $\mathcal{A}$, and $\mathcal{T}$ are pairwise disjoint and finite,
- $\mathit{case} \in \mathcal{A} \to \mathcal{C}$ is a surjective function mapping activities to cases,
- $\mathit{task} \in \mathcal{A} \to \mathcal{T}$ is a surjective function mapping activities to tasks,
- $\mathit{art} \in \mathcal{C} \to \mathit{TS}$ is a function specifying the arrival time of cases,
- $\mathit{dur} \in \mathcal{A} \to \mathit{Dur}$ is a function mapping activities to durations,
- $\mathit{prop} \in (\mathcal{C} \cup \mathcal{A}) \to \mathit{Val}$ is a function mapping cases and activities to their invariable properties,
- $\prec \subseteq \mathcal{A} \times \mathcal{A}$ defines a partial order on activities within cases:
  - $\forall_{a_1,a_2 \in \mathcal{A}}\ a_1 \prec a_2 \ \Rightarrow\ \mathit{case}(a_1) = \mathit{case}(a_2)$ (activities of different cases are unordered),
  - $\forall_{a_1,a_2 \in \mathcal{A}}\ a_1 \prec a_2 \ \Rightarrow\ a_2 \not\prec a_1$ (asymmetric), and
  - $\forall_{a_1,a_2,a_3 \in \mathcal{A}}\ (a_1 \prec a_2 \ \wedge\ a_2 \prec a_3) \ \Rightarrow\ a_1 \prec a_3$ (transitive).
- $\mathit{case}^{-1} \in \mathcal{C} \to \mathcal{P}(\mathcal{A})$ is defined such that for $c \in \mathcal{C}$:
  - $\mathit{case}^{-1}(c) = \{a \in \mathcal{A} \mid \mathit{case}(a) = c\}$.
- $\mathit{task}^{-1} \in \mathcal{T} \to \mathcal{P}(\mathcal{A})$ is defined such that for $t \in \mathcal{T}$:
  - $\mathit{task}^{-1}(t) = \{a \in \mathcal{A} \mid \mathit{task}(a) = t\}$.

Definition 3 (Binding, Concrete Event Log) Let $L = (\mathcal{C}, \mathcal{A}, \mathcal{T}, \mathit{case}, \mathit{task}, \mathit{art}, \mathit{dur}, \mathit{prop}, \prec)$ be an abstract event log. $B = (\mathcal{R}, \mathit{res}, \mathit{st}, \mathit{et}, \mathit{WH})$ is a binding for L where:

- $\mathcal{R}$ is a set of resources,
- $\mathit{res} \in \mathcal{A} \to \mathcal{R}$ is a surjective function mapping each activity onto a resource,
- $\mathit{st} \in \mathcal{A} \to \mathit{TS}$ assigns a start time to each activity and $\mathit{et} \in \mathcal{A} \to \mathit{TS}$ assigns an end time to each activity such that:
  - $\forall_{a \in \mathcal{A}}\ \mathit{st}(a) \geq \mathit{art}(\mathit{case}(a))$ (activities must start on or after the arrival time of the case they belong to),
  - $\forall_{a \in \mathcal{A}}\ \mathit{st}(a) < \mathit{et}(a)$ (the start time of an activity must be earlier than its end time), and
  - $\forall_{a_1,a_2 \in \mathcal{A}}\ a_1 \prec a_2 \ \Rightarrow\ \mathit{et}(a_1) \leq \mathit{st}(a_2)$ (the end of an activity must not come after the start of an activity that it precedes).
- $\mathit{WH} : \mathcal{R} \to \mathcal{P}(\mathit{TS} \times \mathit{TS})$ is a set of timestamp pairs denoting the valid working hours of resources (where resources are allowed to execute activities), such that:
  - for $(t, t') \in \mathit{WH}(r)$, $\mathit{stw} \in \mathit{WH} \to \mathit{TS}$ denotes the starting time of the working hour, i.e. $\mathit{stw}(t, t') = t$, and $\mathit{etw} \in \mathit{WH} \to \mathit{TS}$ denotes the end time of the working hour, i.e. $\mathit{etw}(t, t') = t'$,
  - for all $r \in \mathcal{R}$, and all $p_1, p_2 \in \mathit{WH}(r)$, $p_1 = p_2$, or $\mathit{stw}(p_2) > \mathit{etw}(p_1)$, or $\mathit{stw}(p_1) > \mathit{etw}(p_2)$,
  - furthermore, for all $p \in \mathit{WH}(r) : \mathit{stw}(p) < \mathit{etw}(p)$.

We adapt our activity duration mapping function, Dur, to take working hours into account:

$$\mathit{dur}^{\mathit{WH}}_{(L,B)}(a) = \sum_{p \in \mathit{WH}(\mathit{res}(a))} \mathit{dur}(\mathit{st}(a), \mathit{et}(a), \mathit{stw}(p), \mathit{etw}(p)),$$

where

$$\mathit{dur}(\mathit{st}(a), \mathit{et}(a), \mathit{stw}(p), \mathit{etw}(p)) =$$

$$\begin{cases} 0 & \text{if } et(a) \leq stw(p) \vee etw(p) \leq st(a) \\ et(a) - st(a) & \text{if } st(a) \geq stw(p) \wedge et(a) < etw(p) \\ etw(p) - st(a) & \text{if } etw(p) \leq et(a) \wedge stw(p) < st(a) \wedge etw(p) > st(a) \\ et(a) - stw(p) & \text{if } stw(p) \geq st(a) \wedge etw(p) > et(a) \wedge stw(p) < et(a) \\ etw(p) - stw(p) & \text{if } stw(p) > st(a) \wedge etw(p) < et(a) \end{cases}$$

The working hours of a resource are taken into account when the duration of an activity is calculated. Thus, the activity duration remains the same across different bindings, and the time gaps where resources are not working do not influence the duration of an activity.

The combination of L and B, (L, B) is a concrete event log, where only the res, st, and et are changed. $\mathcal{R}$ and WH will remain the same. A concrete event log represents an (alternative) execution scenario. Table 4 illustrates the abstract event log and the binding that corresponds to the event log in Table 3. By changing the information in the bindings, alternative execution scenarios (new concrete event logs) can be produced.

Table 4: An abstract event log (top) and binding (bottom) that correspond to the car insurance claim event log in Table 3.

Abstract Event Log

ID	Case ID	Activity	Property (Damage Type)	Duration	Preceding Activity	Succeeding Activity
341	1	CAR	Windscreen	00:08:00	{}	{RAR}
342	1	RAR	-	00:18:00	{CAR}	{NCU}
343	2	CAR	Theft	00:05:00	{}	{RAR}
344	2	RAR	-	00:15:00	{CAR}	{NCU}
...	...	...	...	...	...	...

Binding

ID	Start Time	Complete Time	Resource
341	10/06/13 09:31:00	10/06/13 09:39:00	A5
342	10/06/13 09:42:00	10/06/13 10:00:00	IS2
343	10/06/13 09:45:00	10/06/13 09:50:00	A1
344	10/06/13 09:55:00	10/06/13 10:10:00	IS1
...	...	...	...

Definition 4 (Resource Eligibility) Let (L, B) be a concrete event log with $L = (\mathcal{C}, \mathcal{A}, \mathcal{T}, \mathit{case}, \mathit{task}, \mathit{art}, \mathit{dur}, \mathit{prop}, \prec)$ and $B = (\mathcal{R}, \mathit{res}, st, et, WH)$. $\mathit{can} \in \mathcal{T} \rightarrow \mathcal{P}(\mathcal{R})$ returns the set of resources that are eligible to perform a task (can perform), such that if $\mathit{can}(t) = R$, any resource in R can perform task t.

Organisations can define eligibility of resources for task execution through the use of roles and rules (e.g. required skills). Furthermore, the availability of resources can also be explicitly captured (e.g. by specifying working hours).

Definition 5 (Safe Binding) Let $L = (\mathcal{C}, \mathcal{A}, \mathcal{T}, \mathit{case}, \mathit{task}, \mathit{art}, \mathit{dur}, \mathit{prop}, \prec)$ be an abstract event log, and can be the set of resource eligibility. $B = (\mathcal{R}, \mathit{res}, st, et, WH)$ is a safe binding for L iff:

- $\forall r \in \mathcal{R}\ \forall t \in \mathit{TS} : |act^{res}_{(L,B)}(r,t)| \leq 1$ (a resource works on at most one activity at any point in time),
- $\forall_{a\in\mathcal{A}}\ res(a) \in can(task(a))$ (a resource involved in the execution of an activity must be eligible to perform it), and
- $\forall_{a\in\mathcal{A}}\ \exists_{p_1,p_2\in \mathit{TS}\times \mathit{TS}}\ [(p_1, p_2 \in \mathit{WH}(res(a)))\wedge(stw(p_1) \leq st(a) \leq etw(p_1))\wedge (stw(p_2) \leq et(a) \leq etw(p_2))]$ (the execution of activities must be within the defined time blocks where resources are allowed to execute activities).

A binding is deemed unsafe if 1) a resource works on multiple activities at any point in time, 2) an activity is not performed by an eligible resource (see Definition 4), or 3) an activity is executed outside the defined working hours of its allocated resource. 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.

Definition 6 (Utilisation) Let (L, B) be a concrete event log with $L = (\mathcal{C}, \mathcal{A}, \mathcal{T}, case, task, art, dur, prop, \prec)$ and $B = (\mathcal{R}, res, st, et, \mathit{WH})$. Let $r \in \mathcal{R}$, $t \in \mathit{TS}$ and $h \in \mathit{Dur} \setminus \{0\}$, then the utilisation of resource r at time t using a horizon h is given by:

$$util^{h}_{(L,B)}(r,t) = \frac{\displaystyle\sum_{\substack{a\in\mathcal{A}, res(a)=r,\\ st(a)<t\ \wedge\ et(a)>(t-h)}} dur^{WH}_{(L,B^h)}(a)}{\displaystyle\sum_{p\in \mathit{WH}(r)} dur(t-h, t, stw(p), etw(p))},$$

where $B^h = (\mathcal{R}, res, st', et', \mathit{WH})$, and

$$st'(a) = \begin{cases} st(a) & \text{if } st(a) \geq t-h \wedge st(a) < t \\ t-h & \text{if } st(a) < t-h \wedge et(a) > t-h \end{cases},$$

and

$$et'(a) = \begin{cases} et(a) & \text{if } et(a) \geq t-h \wedge et(a) \leq t \\ t-h & \text{if } et(a) > t \wedge st(a) > t \end{cases}.$$

Resource utilisation is the fraction of time a resource is busy (executing activities) in a specified time frame. The horizon specifies the time frame that is used to compute the resources' utilisation rates. If B is a safe binding for L, then $0 \leq util^{h}_{(L,B)}(r,t) \leq 1$ for any resource r, time t, and for any positive time horizon $h > 0$. $\mathit{Util} = [0, 1]$ is the set of possible utilisations. If a safe binding is produced, the utilisation of any resource will be between zero and one within a positive time horizon.

Definition 7 (Cost Structure) Let $\mathcal{R}$ be a set of resources, $\mathcal{T}$ be a set of tasks, and $\mathit{Costs} = \mathbb{R}^+_0$ a set of costs. $\mathit{CS} = (costs^{case}, costs^{act}, costs^{res})$ is a cost structure over $\mathcal{R}$ and $\mathcal{T}$ where:

- $costs^{case} \in (\mathit{Dur} \times \mathit{Val}) \to \mathit{Costs}$ such that $costs^{case}(d, v)$ are the costs of a case having duration $d \in \mathit{Dur}$ and invariable properties $v \in \mathit{Val}$,

– $costs^{act} \in (\mathcal{T} \times \mathcal{R} \times Dur \times Val) \to Costs$ such that $costs^{act}(t, rs, d, v)$ are the costs of executing task $t \in T$ by resources $rs \subseteq \mathcal{R}$, having duration $d \in Dur$, and invariable properties $v \in Val$, and
– $costs^{res} \in (\mathcal{R} \times Dur \times Util) \to Costs$ such that $costs^{res}(r, d, u)$ are the costs of using resource $r \in \mathcal{R}$ for $d \in Dur$ time units and having a utilisation of $u \in Util$.

A cost structure represents a generic data model that stores a set of cost functions for process-related cost computations (see [13] for details). Organisations can not only define functions to calculate the cost of a case, an activity, and a resource, but also functions to enforce business rules such as service level agreements and resource eligibility. To compute the total cost of a concrete event log, the cost of cases, activities, and resources are aggregated. Shorter case durations that do not breach the service level agreement, optimal resource utilisation, and appropriate resource allocation are desired as the goal is to achieve a reduction in overall execution cost. This cost structure is used as a basis of the objective function within the proposed optimisation techniques.

3.2 Motivating Example

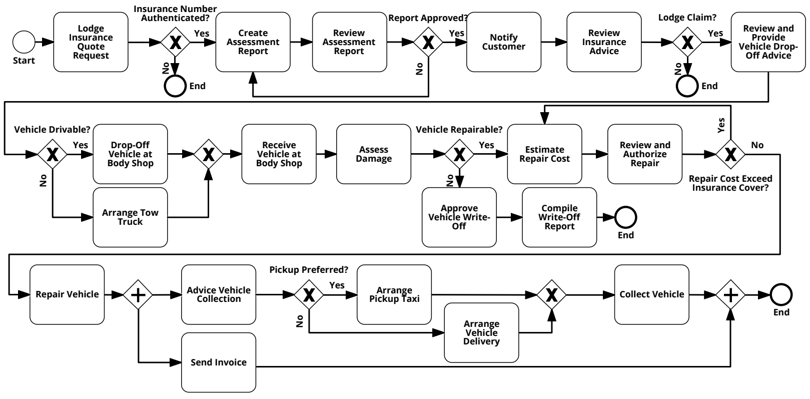


Fig. 2: A BPMN model illustrating the car insurance claim process.

Fig. 2 illustrates a car insurance claim process that was used as a motivating example in this work. The process was inspired by AAMI's car insurance claim process[1], where AAMI is an insurance provider in Australia. The process starts off within the Insurance Company when a customer lodges an insurance quote request for his damaged car. If the customer's insurance number is positively authenticated, an Insurance Adjuster will create an assessment report

[1]`http://www.aami.com.au/car-insurance/claimsprocess`

based on the quote description. An Insurance Supervisor will then review the assessment report. If the report is approved, the Insurance Adjuster will notify the customer. If the report is rejected, the Insurance Adjuster will need to recreate the report. The customer will then decide whether to lodge a claim by reviewing the insurance advice. Should the customer lodge a claim, a Service Coordinator will review and provide advice on dropping-off the vehicle at the body shop. If the vehicle is drivable, the customer will drop-off the vehicle at the body shop. If not, the Service Coordinator will arrange a tow truck to collect the vehicle on behalf of the customer.

Within the Body Shop, a Foreman receives the damaged vehicle and assesses the damage. If the damage is not repairable, a Body Shop Supervisor will approve the vehicle for write-off, and a Write-Off Specialist will compile a write-off report afterwards. If the damage is repairable, the Foreman will estimate the repair cost and provide feedback to the Insurance Supervisor. If the estimation exceeds the insurance cover, the estimation will be rejected, and the Foreman will need to readjust the estimated repair cost. Otherwise, the Foreman will repair the vehicle accordingly. After the repair is completed, the Body Shop Supervisor will send an invoice to the Insurance Company while the Service Coordinator advises the customer for vehicle collection. The vehicle can either be delivered to the customer, or a pickup taxi can be arranged to transport the customer to the Body Shop for vehicle pick-up. The process is complete when the customer collects the vehicle.

The process was translated into a Petri net [17] and was simulated using CPN Tools[2] to obtain an event log with 500 cases. The process spans across two organisational groups, which consists of 16 tasks, 6 roles, and 29 resources. Each task can only be executed by a certain role(s) (group of resources). Each resource group (role) is responsible for a fixed set of tasks. Realistic working hours were also introduced in the process simulation, where all resources are only allowed to work between 9 am and 5 pm.

Table 5 illustrates an excerpt of the defined cost functions elaborated below:

- The cost of a case is calculated based on its case properties and its duration. An additional service level agreement has been specified, in which cases that ran overtime are penalised.
- The cost of an activity is calculated based on that activity's resource allocation. A higher cost is incurred if an inappropriate/less-desired resource executes that activity.
- Resource costs are calculated based on resource utilisation. The per minute cost rate is determined by the resource attributes (for example, skills and roles) and utilisation for the specified time horizon. An assumption for this example is that the desired resource utilisation is 0.8. A stepped cost rate, where a resource's utilisation between 0.75 and 0.85 within the time horizon is cheapest is used. On the other hand, resource under- or over-utilisation results in a higher cost rate.

[2] `http://cpntools.org/`

Table 5: Example of cost functions defined in the cost structure.

Cost Type	**Property**	**Value**	**Cost Rate**
Case	Damage Type & Case Duration	Windscreen	\$40 per hour
	Case Duration	[Service Level Agreement Breach]	\$1500 if it takes more than 5 days, and \$500 for every subsequent day after that
Activity	Activity & Resource	CAR & A1	\$100 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 0.85 (optimum utilisation)	\$1 per minute
	Resource Utilisation	Higher than 0.9 (over-utilised)	\$20 per minute

4 Optimisation Techniques

This section discusses the optimisation techniques proposed to identify less costly execution scenarios by exploring different possible bindings for a given event log. The event log and the cost structure defined in Section 3 are used as the problem definition and the objective function of our proposed optimisation techniques respectively. In our previous work, we looked into a hybrid genetic algorithm to find a cost-optimised event log [13]. Extending from that, we aim to compare our approach against other potential optimisation techniques, as well as make optimisation more realistic by taking working hours into account. Hence, we look into ILP as our first approach towards cost optimisation. However, a number of drawbacks were encountered, leading us towards the use of heuristic methods as a next logical step. We re-run the experiments and compare methods such as tabu search and hill climbing against the hybrid genetic algorithm-based approach that we have proposed in our previous paper [13]. The details on the optimisation techniques, the difficulties faced, and further analyses are discussed in the remainder of this section.

4.1 ILP

A subclass of the integer programming problems, integer linear programming (ILP) problems are a mathematical description of an optimisation problem in which some or all the constraints and the objective functions have to be integers [18] and expressed in linear form. As ILP can take into account different real-life aspects, it was chosen as the first approach because it is often a good solution for complex problems and allows the definition of problem structure formally.

As only the resource allocations and activity start times can be changed, the key aspect of the ILP formalisation is that the resource assignment and start time assignment are represented as a binary "hypercube", containing a binary value for each (CaseNo, ActNo, Resource, Time) combination, denoting a particular resource and starting time for each activity in every case.

We propose an ILP formalisation of our problem as follows. Given:

- An ordered list of case identifiers: $C = \{C_1, C_2, \ldots, C_{|C|}\}$.
- An ordered list of activities per case: $A = \{A_i | i = 1, \ldots, |C|\}$ with: $A_i = \{A_{i,1}, \ldots, A_{i,|A_i|}\}$.

- An ordered list of resources: $R = \{R_1, \ldots, R_{|R|}\}$.
- An ordered list of time slots: $T = \{T_1, \ldots, T_{|T|}\}$.
- $arr \in C \rightarrow T$: the arrival time of a case.
- $dur \in A \rightarrow \mathbb{N}$: the duration of an activity.
- $forb_{actres} \in A \rightarrow \mathcal{P}(\mathcal{R})$: the set of resources which may not be assigned to an activity.
- $forb_{acttime} \in A \rightarrow \mathcal{P}(\mathcal{T})$: the set of time slots where an activity may not be scheduled.
- $forb_{actres} \in R \rightarrow \mathcal{P}(\mathcal{T})$: the set of time slots where a resource may not be scheduled.
- $cost_{dur} \in \mathbb{N} \rightarrow \mathbb{R}$, $cost_{start} \in \mathbb{N} \rightarrow \mathbb{R}$, $cost_{res} \in A \times R \rightarrow \mathbb{R}$, $cost_{dur} \in R \rightarrow \mathbb{R}$: functions denoting duration, starting (delay), resource usage and resource duration costs respectively.

With variables:

- $x_{i,j,r,t} \in \{0, 1\}$ with: solution variables representing a binary hypercube denoting whether activity $A_{i,j}$ has been scheduled to start at time T_t using resource R_r.
- $s_{i,j} \in T$: integer helper variables denoting the start time of activity $A_{i,j}$.
- $e_{i,j} \in T$: integer helper variables denoting the end time of activity $A_{i,j}$.
- $u_r \in \mathbb{N}$: integer helper variable denoting utilisation value of resource R_r.

We can formulate a candidate objective function as follows:

$$min \sum_{C_i \in C} cost_{dur}(e_{i,|A_i|} - s_{i,1}) + \tag{1}$$

$$\sum_{C_i \in C} cost_{start}(s_{i,1}) + \tag{2}$$

$$\sum_{A_{i,j} \in A,\ R_r \in R,\ T_t \in T} cost_{res}(A_{i,j})\ x_{i,j,r,t} + \tag{3}$$

$$\sum_{R_r \in R} cost_{util}(R_r)\ u_r \tag{4}$$

Where (1) is used to calculate a cost associated to the duration of each case, depending on the scheduled starting times of the first and last activity in each case. (2) indicating the starting cost (postponing the start of a case leads to higher costs). (3) indicating the cost of assigning a particular resource to a particular activity, and (4) indicating the cost of utilisation, which is fixed per resource but multiplied by the u_r helper variable to estimate the utilisation rate of each resource. The constraints, then, were formalised as follows:

$$\forall_{A_{i,j} \in A} : \sum_{R_r \in R,\ T_t \in T} x_{i,j,r,t} = 1 \tag{1}$$

$$\forall_{A_{i,j} \in A} : \sum_{R_r \in R,\ T_t \in T : t < arr(C_i)} x_{i,j,r,t} = 0 \tag{2}$$

$$\forall_{A_{i,j} \in A : j > 1} : e_{i,j-1} - s_{i,j} \geq 0 \tag{3}$$

$$\forall_{R_r \in R, T_t \in T} : \sum_{A_{i,j} \in A} \sum_{\substack{m \in \{0, \dots, dur(A_{i,j})-1\}: \\ t-m>0 \ \wedge \ t-m \leq \max(T)}} x_{i,j,r,t-m} \leq 1 \tag{4}$$

$$\forall_{A_{i,j} \in A} : \sum_{R_r \in forb_{actres}(A_{i,j}),\ T_t \in T} x_{i,j,r,t} = 0 \tag{5}$$

$$\forall_{A_{i,j} \in A} : \sum_{\substack{R_r \in R, \\ T_t \in forb_{acttime}(A_{i,j})}} \sum_{\substack{m \in \{0, \dots, dur(A_{i,j})-1\}: \\ t-m>0 \ \wedge \\ t-m \leq \max(T)}} x_{i,j,r,t-m} = 0 \tag{6}$$

$$\forall_{R_r \in R} : \sum_{\substack{A_{i,j} \in A, \\ T_t \in forb_{restime}(R_r)}} \sum_{\substack{m \in \{0, \dots, dur(A_{i,j})-1\}: \\ t-m>0 \ \wedge \\ t-m \leq \max(T)}} x_{i,j,r,t-m} = 0 \tag{7}$$

$$\forall_{R_r \in R} : u_r = \sum_{A_{i,j} \in A,\ T_t \in T} dur(A_{i,j})\ x_{i,j,r,t} \tag{8}$$

$$\forall_{A_{i,j} \in A} : s_{i,j} = \sum_{R_r \in R,\ T_t \in T} t\ x_{i,j,r,t} \tag{9}$$

$$\forall_{A_{i,j} \in A} : e_{i,j} = \sum_{R_r \in R,\ T_t \in T} (t\ x_{i,j,r,t} + dur(A_{i,j})) \tag{10}$$

Constraint (1) ensures that exactly one resource and start time is assigned to each activity. (2) ensures that the start time of activities adheres to the arrival time of cases. (3) ensures that the ordering of activities is adhered to. (4) ensures that resources do not overlap in time. Note that this constraint needs to be checked over multiple time periods, as the $x_{i,j,r,t}$ variable only represents the scheduled starting time for each activity. (5) allows for prohibiting resources to be assigned to particular activities. (6) and (7) allow to constrain working times for activities and resources respectively. (8), (9) and (10) are constraints ensuring the assignment of the helper variables.

A number of remarks are relevant regarding the formalisation above. First, note that the construction of the helper variables is optional and added for clarity. That is, they can be immediately inserted into each expression whilst retaining linearity. Second, it is important to note that the utilisation expression as introduced before cannot be expressed in a linear form, and is thus estimated here using an absolute integer variable. Hence, the cost structure is adapted to a heuristic utilisation cost computation (a fixed cost per resource invocation), where the number of time slots a resource is used (in total) was counted and a cost is derived based on that. Third, although we express our decision variable with a four-dimensional subscript, it is also possible to express this as a three-dimensional variable, i.e. an activity identifier, a resource identifier, and a time identifier. Fourth, due to the nature of integer linear programming, time is discretised into a list of “time slots”. Resources are also expressed as a list, but can nevertheless still allow to assign groups of resources to activities (with each possible subgroup then being captured in a separate R_r item). Fifth, it is not possible to construct the problem by means of two separate decision variables (for example, t_{act} and r_{act}) with one representing the assigned starting time and one representing the assigned resource, as this would render it impossible to formulate the problem in linear form. The same remark holds when expressing the assignment of time as a non-binary integer (representing the index of the assigned time slot) instead of a binary vector.

Although we have successfully applied the ILP formalisation introduced above on small problems, this method becomes unusable on larger problems, due to the exponentially growing variable size and complexity concerning input size. Additionally, as mentioned above, the resource utilisation calculation, where the utilisation of a resource within a certain horizon is computed, cannot be done linearly. Due to these drawbacks, we explore other heuristic techniques as an alternative to ILP below.

4.2 Hill Climbing

Hill climbing is a local search mathematical optimisation technique that employs an iterative search that attempts to find a better solution by incrementally changing a single element of an arbitrary solution. Hill climbing is the first choice after ILP because it is relatively easy to implement and is relatively fast for problems that have many solutions. An initial random safe solution, in this case, a random generation of a safe binding is chosen as a starting point. By going through the neighbours, an objective function is used to evaluate the neighbours, with cost minimisation as its goal. If the cost of a particular neighbour is lower than the initial solution, the solution is replaced with the corresponding neighbour. The search is repeated until the stop criteria have been met, or no neighbours with lower cost, i.e. a local optimum has been found.

We have customised the neighbourhood search function to increase its speed and scalability. We first applied an incremental search approach, where a neighbour solution was produced for each activity, for each allowed time period, and for each available resource. However, due to the high number of candidate neighbours generated in each iteration when applying this procedure, we improved the scalability by sampling candidate neighbours for each activity. To further improve the speed and scalability of this method, we adapted the approach to generate a fixed number of nearby neighbours, that is, a global sample over all possible activity, time, and resource possibilities. To generate a neighbour (safe binding), an activity is first picked randomly, for which the time periods where the activity can start and the eligible resources are obtained. A start time for that activity is chosen, as well as an eligible resource. The chosen start time and resource are applied to generate a neighbour and subsequently passed on to the hill climbing method for further analysis.

Either with or without sampling, the main disadvantage of the hill climbing method is that it will not necessarily converge to a global minimum. Given the complexity of the proposed problem for this study, it is very likely that the hill climbing method will result in a local minimum. Hence, we investigated another optimisation technique as a remedy to this drawback.

4.3 Tabu Search

Tabu search is a metaheuristic search method that employs local search methods used for mathematical optimisation [7]. A distinguishing feature of tabu search is the use of a "memory" construct, where recent searches are recorded in a so-called tabu (forbidden) list for a set amount of iterations. Tabu search was investigated because it has a relatively fast execution time. Also, the use of a tabu list prevents the search from retracing its steps and reinvestigating previous moves to try and avoid local minima.

Tabu search also starts off from a single, random safe solution, in this case, a random generation of a safe binding. Based on the binding, a set of nearby neighbours that are not in the tabu list is explored. The neighbourhood is searched in a similar manner to the hill climbing's neighbourhood search. The objective function is then used to evaluate the cost of the neighbours, where cost minimisation is the ultimate goal. The current solution is replaced with the neighbour solution with the lowest cost, and the tabu list is updated with the recently explored solutions. The search is repeated and only terminated when the stop criterion is met. To increase the speed and scalability of tabu search, we perform a similar sampling procedure as described in the previous subsection that only generates a fixed amount of neighbours in each iteration. Due to the potential memory constraint that is dependent on the tabu list size, we looked into a genetic algorithm-based optimisation as an alternative approach to our problem.

4.4 Hybrid Genetic Algorithm

We also developed a hybrid genetic algorithm to facilitate the construction and exploration of the massive search space at hand in this context [13]. 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 are 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 [30]. An additional heuristic approach, where there is a low chance of generating a new random solution, was incorporated to minimise the chances of the genetic algorithm converging into a local minimum. Hence, a hybrid genetic algorithm resulted from the addition of a heuristic operator.

Reinforcing the previous arguments, 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 impractical. Furthermore, due to the many local optima, traditional heuristic approaches (for example, hill climbing) and simulated annealing are deemed less suitable as they tend to be more prone to converge to such suboptimal solutions [39]. Therefore, it is argued that a hybrid genetic algorithm-based approach is the adequate technique for our problem.

Genetic operators are responsible for evolving a next generation of a population (multiple solutions). To produce a new solution (individual), the operators are applied to a parent or a pair of parent solutions. A number of operators have been developed for the hybrid genetic algorithm. They are designed in such a way that unsafe bindings produced will be penalised heavily by the objective function. This will encourage the generation of safe bindings, taking into account resource availability, authorisation, and working hour restrictions. The operators used in this work and their frequencies are as follows.

1. **Crossover**. The crossover operator cross-breeds the properties (activity start time and resource allocation) of a number of activities between two bindings. A crossover point is picked randomly from the parent bindings' list of activities, and a specified number of activity crossovers (*crossover frequency: 1% of total activity count*) are applied, where the activities' start times and resource allocations are swapped between two parent bindings. 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.
2. **Time Mutation**. The time mutation operator changes the start times of a number of activities within a binding individual. For a number of cases (*case frequency: Poisson (100% of total case count)*[3]), a number of activities (*activity frequency: Discrete (0% to 100% of total activity count)*[4]) 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 chosen start time will be checked for potential violations of the activity order and the resource allocation authorisation. If the mutation will result in an unsafe binding, no mutation is performed. The activity's start time is mutated if it does not result in violations.
3. **Resource Mutation**. The resource mutation operator swaps the resource that is executing an activity. For a random number of cases (*case frequency: Poisson (10% of total case count)*[3]), a random number of activities (*activity frequency: Discrete (0% to 100% of total activity count)*[4]) are selected to have their resource allocation mutated. A different resource is randomly picked from the pool of *eligible* (allowed to execute the activity) and *idle* (available) resources (refer to Definition 4 in Subsection 3.1) for mutation. If there is no resource available, no mutation is performed.

[3]Poisson (*mean*) = pick a random value from a Poisson distribution with the stated mean.

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

4. **New Heuristic Binding**. This operator introduces a new safe binding into the population. Although typically applied with a low probability (*heuristic frequency: 5% of total population count*), 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 (based on previously scheduled activities), and the resource allocation process is repeated. This process is iterated until all activities have been scheduled. This operator ensures the generation of a safe binding.

5 Experimental Setup and Discussion

This section first describes the experimental setup used to evaluate the effectiveness and efficiency of different approaches, followed by a detailed discussion about the experimental results. Two experiments are conducted using two event logs, and the performance and scalability of the four optimisation techniques are discussed and compared.

5.1 Experiment Setup

To evaluate the performance and scalability of the proposed optimisation techniques, two event logs of the car insurance claim process were used. As the goal of this study is to improve the business process within the organisation, activities that are performed by customers were filtered out. After filtering, the simulated event log contained 453 cases (this log is referred to as *Log A* from now on). The second event log (this log is referred to as *Log B* from now on) is a sliced version of Log A, where only the initial 10% in terms of time ordering of the log is taken into consideration. Log B contained only 45 cases.

The experiments were executed over the period of one week[5,6]. To increase the speed of the optimisation techniques, and to enable sensible objective function calculation, the timestamps within the event log were discretised into one-minute blocks. Results from the previous experiment performed [13] indicates that safe variants of the technique (operators that only explore safe bindings) provided better results. Hence, in this experiment, only safe bindings are explored. Apart from ILP, where its cost structure (objective function) is linearised and estimated as described above, the cost structure used across

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

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

different algorithms and different logs is the same. The objective functions and parameters for the different algorithms are fixed in consideration of the search space and the experiment's feasibility. Table 6 details the settings and specifications for each optimisation algorithms.

Table 6: Settings and specifications for the optimisation algorithms.

Optimisation Technique	**Objective Function Description**	**Additional Specifications**
ILP	Minimisation of duration, cost of delay, resource assignment and resource utilisations, linearised and estimated.	N/A
Hill Climbing	Full cost structure applied to each population member.	(i) 100000 neighbour generation per iteration.
Tabu Search	Full cost structure applied to each population member.	(i) 100000 neighbour generation per iteration. (ii) Tabu size of 5000.
Hybrid Genetic Algorithm	Full cost structure applied to each population member.	(i) Population size of 50. (ii) Top 5% of total population count as elites. (iii) Tournament Selection strategy, 75% probability that a fitter individual will be selected.

5.2 Result Analysis and Discussion

The goal of the experiments is to measure the performance of the proposed optimisation techniques in terms of cost reduction over time. The performance of ILP was first experimented using Log B. For this relatively small-sized log, the experiment was unable to complete as it required more than 80GB of stack memory to contain the problem structure. Consequently, we did not proceed to experiment with Log A, and we conclude that utilising ILP for optimisation is infeasible for real world usage. Following that, two experiments using both Log A and Log B were conducted over a one week period to compare the performance and scalability of the remaining three techniques.

Fig. 3 illustrates the cost reduction of Log B achieved by the three optimisation techniques over a week. In this experiment, all three techniques managed to discover cheaper alternative execution scenarios. The hill climbing and tabu search techniques performed slightly better with a cost reduction of over 13%. The hybrid genetic algorithm's performance fell slightly short of that, with a 12.75% reduction in cost over time. Although hill climbing and tabu search have a higher cost reduction rate over time, the hybrid genetic algorithm managed to gain a high reduction in a small time, reducing up to 12% within the first 6 hours from when the experiment was initialised. However, the performance of the hybrid genetic algorithm plateaued after that. Hill climbing and tabu search reached 12% roughly 40 hours into the experiment, and although slow, continually reduced the cost and surpassed the hybrid genetic algorithm after two days of running time. It should be noted that the hill climbing technique restarted around the 100-hour mark, as no cheaper alternative execution scenario was found during the search iteration. For this

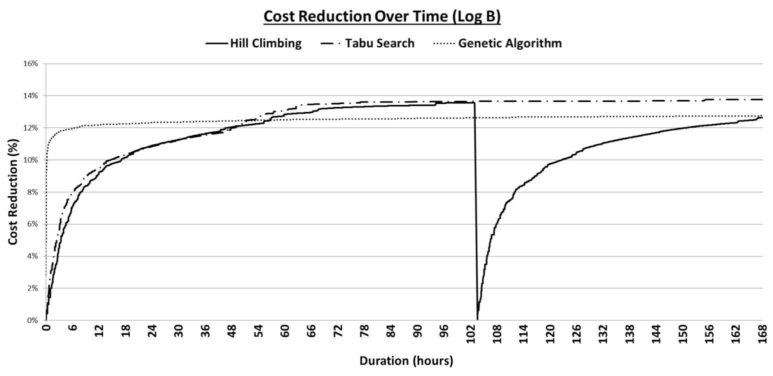


Fig. 3: Log B's cost reduction over a period of one week.

experiment, we thus observe that the hill climbing, tabu search, and hybrid genetic algorithm perform equally well.

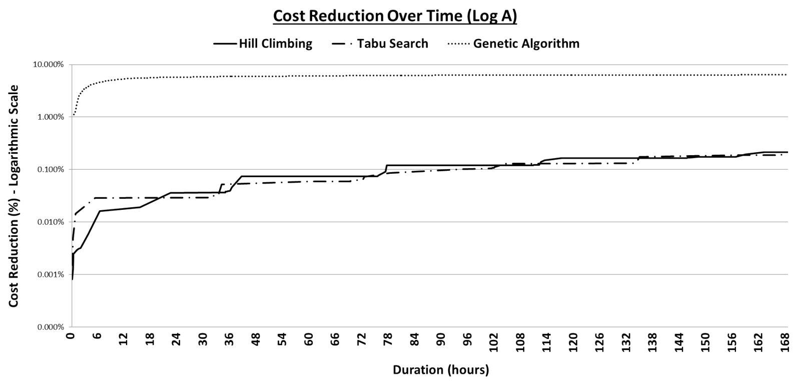
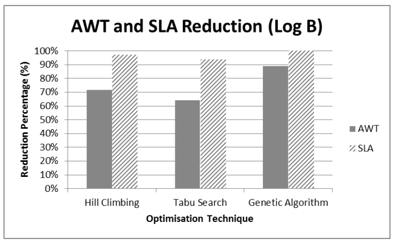
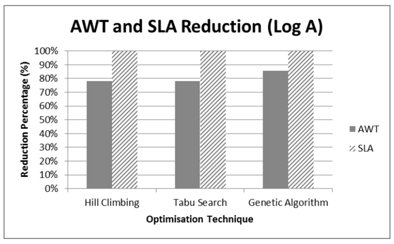


Fig. 4: Log A's cost reduction (logarithmic scale) over a period of one week.

The experiment conducted with Log B is for scalability test purposes only, and the size of this log is a far cry from a real-life event log. Hence, we turn our attention to the second experiment to gain a more concrete conclusion. We now analyse the experiment that made use of Log A, with results illustrated in Fig. 4. The hybrid genetic algorithm performed distinctively better in this scenario, with a cost reduction of 6.36% after one week of computation, whereas hill climbing and tabu search only managed to reduce the cost by approximately 0.2%. Again, a similar pattern to the previous experiment regarding the performance of the hybrid genetic algorithm was observed. The hybrid genetic algorithm rapidly reduced the cost, reducing 4.5% of the cost



(a) Experiment results in terms of average waiting time between activities and service level agreement breach count using Log B.



(b) Experiment results in terms of average waiting time between activities and service level agreement breach count using Log A.

Fig. 5: Experiment result in terms of non-cost indicators.

in the first 6 hours, and up to 5.5% in the following 12 hours, which strongly highlights the validity of this approach for real-life sized event logs.

Also, several non-cost indicators were used as measurement factors. The average waiting time between activities within the case was calculated, where a reduction in average waiting time results in a reduction of service level agreement breaches, therefore bringing down the cost. Fig. 5 shows the changes in relation to these two non-cost indicators. All techniques managed to reduce the average waiting time while also reducing the service level agreement breach count. It was additionally be observed that the hybrid genetic algorithm, in particular, performs slightly better in reducing the average waiting time, reducing at least 85% of the average waiting time compared with less than 78% for the other two techniques. All techniques also managed to considerably reduce, or even eliminate the number of cases that breach the service level agreement. From a resource perspective, the resources are utilised in a much more effective way, resulting in changes to their overall resource utilisation percentage. Our cost optimisation technique not only achieved a reduction in cost, but also increased the efficiency in a number of process aspects.

In this section, we discussed the approach undertaken to evaluate the proposed cost optimisation techniques with different logs. These preliminary results show that it is possible to learn from the history by generating alternative scenarios to satisfy the goal of cost minimisation. In a nutshell, the ILP technique is straightforward, but its application to real-world business processes is unrealistic. The hill climbing and tabu search techniques perform well with small sized logs. Although most of the options to improve the efficiency of the techniques have been considered, the hybrid genetic algorithm approach still outperforms them on realistic logs. Considering the emphasis on cost reduction, the importance of scalability, and the balance between the non-cost indicators, it can be concluded that the hybrid genetic algorithm exhibited the best results.

Now we focus on some of the potential limitations of the presented work. We assume that both start and complete times of activities are recorded in an event log. This assumption enables us to compute accurate activity durations and allows us to reduce activity wait times by shifting forward the start time of the activity if a suitable resource is available. However, we acknowledge that it not all event logs generated by IT systems contain both start and complete times for activities. In such cases, organisations need alternative methods to calculate or approximate activity durations, or design a cost function that caters for this deficiency. We would also like to note that the car insurance claims example and its corresponding event log are not generated by a domain expert. The example is inspired by a widely available insurance claim process[7], as well as our research group's expertise with process models and logs from the insurance sector[8]. The example cost structure used in the paper is intended to demonstrate the extent of the proposed generic cost structure which is backed by insights gained from literature in the areas of management accounting and business process improvement. It is possible that some of the cost functions and cost values presented in this cost structure may not be realistic for a particular organisation. To conduct a more thorough evaluation of our approach, we are dependent on an organisation providing us with a detailed event log and a tailored cost structure, which is stated as part of our future work.

6 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 fixed and variable parts of an event log. On top of this framework, an adaptive cost structure that captures different cost-related dimensions has been proposed and incorporated as the objective function. A number of optimisation techniques were implemented and subsequently compared. The experiments performed concluded that, although the time required for optimisation is quite high, the hybrid genetic algorithm is still the best approach to achieve a cost-informed process improvement in terms of performance and scalability. Observe that in our approach, the overall cost of a business process (based on the process behaviour represented in an event log) is computed, not the cost of individual cases. Organisations can also utilise our proposed approach to perform scenario analysis or what-if analysis by using different cost structures, enabling process analyst to analyse various to-be scenarios using different cost functions and efficiency measures defined within the cost structure.

As future work, our proposed approach could be evaluated not only by using a real-world scenario, but also by using a real-world cost structure. The viability and performance of other optimisation techniques could be investigated as well. Another possible direction of future work is to explore the relaxation of optimisation constraints, where our recommended technique 1)

[7]`http://www.aami.com.au/car-insurance/claimsprocess`

[8]`http://bpm-research-group.org/research/rss-publications`

could be extended to allow activities to be executed in parallel, and 2) could incorporate changes in activity durations to cater for productivity differences among resources. It is also possible to explore the differences between the baseline and improved event logs and investigate visualisation techniques to better portray the degree to which these differences contribute to cost minimisation. This may lead to the development of a methodology for deriving cost-related insights and using these insights as a basis for improvement recommendations.

Acknowledgements This work is supported by Australian Research Council (ARC) Discovery Grant 120101624. We would also like to thank Professor Erhan Kozan for his input to the ILP formalisation.

References

1. E.H.L. Aarts and J.K. Lenstra. *Local search in combinatorial optimization.* Princeton University Press, 2003.
2. M. Birattari, L. Paquete, T. Stiitzle, and K. Varrentrapp. Classification of Metaheuristics and Design of Experiments for the Analysis of Components. 2001.
3. N. Brand and H. van der Kolk. Workflow Analysis and Design. *Deventer: Kluwer Bedrijfswetenschappen*, 1995.
4. S.P. Brooks and B.J.T. Morgan. Optimization using simulated annealing. *The Statistician*, 44(2):241–257, 1995.
5. A. Colorni, M. Dorigo, V. Maniezzo, et al. Distributed Optimization by Ant Colonies. In *Proceedings of the first European conference on artificial life*, volume 142, pages 134–142. Paris, France, 1991.
6. R.C. Eberhart and J. Kennedy. A New Optimizer Using Particle Swarm Theory. In *Proceedings of the sixth international symposium on micro machine and human science*, volume 1, pages 39–43. New York, NY, 1995.
7. F. Glover and M. Laguna. *Tabu search.* Springer, 1999.
8. D.E. Goldberg. *Genetic algorithms in search, optimization, and machine learning.* Addison-Wesley Professional, 1989.
9. S. Hartmann and D. Briskorn. A survey of variants and extensions of the resource-constrained project scheduling problem. *Eur. J. Oper. Res.*, 207(1):1–14, 2010.
10. A.R. Hevner, S.T. March, J. Park, and S. Ram. Design science in information systems research. *MIS Quarterly*, 28(1):75–105, 2004.
11. 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.
12. 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.
13. W.Z. Low, J. De Weerdt, M.T. Wynn, A.H.M. ter Hofstede, W.M.P. van der Aalst, and S.K.L.M. vanden Broucke. Perturbing event logs to identify cost reduction opportunities: A genetic algorithm-based approach. In *Proceedings of the IEEE CEC 2014, Beijing, China, July 6-11, 2014*, pages 2428–2435, 2014.
14. 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.
15. J.D. Martin and J.W. Petty. *Value based management: the corporate response to the shareholder revolution.* Oxford University Press, 2001.
16. I. Masaaki. Kaizen: The Key to Japan's Competitive Success. *New York, McGraw-Hill*, 1986.
17. T. Murata. Petri Nets: Properties, analysis and applications. *Proceedings of the IEEE*, 77(4):541–580, 1989.
18. G.L. Nemhauser and L.A. Wolsey. *Integer and combinatorial optimization*, volume 18. Wiley New York, 1988.

19. R.P. Neuman and R. Cavanagh. *The Six Sigma Way: How GE, Motorola, and Other Top Companies are Honing Their Performance.* McGraw Hill Professional, 2000.
20. A. Rappaport. *Creating shareholder value: a guide for managers and investors.* Simon and Schuster, 1999.
21. R. Ray. Economic value added: Theory, evidence, a missing link. *Review of Business*, 22(1/2):66, 2001.
22. H.A. Reijers and K.M. van Hee. Product-based design of business processes applied within the financial services. *J. Res. Pract. Inf. Tech.*, 34(2):110–122, 2002.
23. C. Sonnenberg and J. vom Brocke. The Missing Link between BPM and Accounting: Using Event Data for Accounting in Process-oriented Organizations. *Business Process Management Journal*, 20(2):213–246, 2014.
24. 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.
25. A. Tiwari, K. Vergidis, and C. Turner. Evolutionary multi-objective optimisation of business processes. In *Soft Computing in Industrial Applications*, pages 293–301. Springer, 2010.
26. W.M.P. van der Aalst. Petri net based scheduling. *Operations Research Spektrum*, 18(4):219–229, 1996.
27. W.M.P. van der Aalst. *Process Mining: Discovery, Conformance and Enhancement of Business Processes.* Springer, 2011.
28. K. Vergidis, D. Saxena, and A. Tiwari. An evolutionary multi-objective framework for business process optimisation. *Applied Soft Computing*, 12(8):2638–2653, 2012.
29. 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.
30. 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.
31. J. vom Brocke, C. Sonnenberg, and U. Baumoel. Linking accounting and process-aware information systems - towards a generalized information model for process-oriented accounting. *European Conference on Information Systems*, pages 1–13, 2011.
32. K. Wang, A. Salhi, and E.S. Fraga. Process design optimisation using embedded hybrid visualisation and data analysis techniques within a genetic algorithm optimisation framework. *Chem. Eng. Process. Process Intensif.*, 43(5):657–669, 2004.
33. W.L. Winston and J.B. Goldberg. *Operations research: applications and algorithms.* Thomson/Brooks/Cole Belmont, 2004.
34. J.P. Womack and D.T. Jones. *Lean Thinking: Banish Waste and Create Wealth in Your Corporation.* Simon and Schuster, 2010.
35. M.T. Wynn, W.Z. Low, A.H.M. ter Hofstede, and W. Nauta. A Framework for Cost-aware Process Management: Cost Reporting and Cost Prediction. *Journal of Universal Computer Science*, 20(3):406–430, 2014.
36. M.T. Wynn, J. De Weerdt, A.H.M. ter Hofstede, W.M.P. van der Aalst, H.A. Reijers, M.J. Adams, C. Ouyang, M. Rosemann, and W.Z. Low. Cost-aware Business Process Management: A Research Agenda. *24th Australasian Conference on Information Systems*, 2013.
37. J. Xu, C. Liu, X. Zhao, and Z. Ding. Incorporating structural improvement into resource allocation for business process execution planning. *Concurr. Comp - Pract. E.*, 25:427–442, 2012.
38. J. Xu, C. Liu, X. Zhao, and S. Yongchareon. Business process scheduling with resource availability constraints. In *OTM to Meaningful Internet Systems*, pages 419–427. Springer, 2010.
39. X. Yang. *Introduction to Mathematical Optimization.* Cambridge International Science Publishing, 2008.
40. J. Yu and R. Buyya. Scheduling scientific workflow applications with deadline and budget constraints using genetic algorithms. *Scientific Programming*, 14(3-4):217–230, 2006.