Discovering Characteristics of Stochastic Collections of Process Models

Kees M. van Hee[1], Marcello La Rosa[2,3], Zheng Liu[1], and Natalia Sidorova[1]

[1] Eindhoven University of Technology, The Netherlands
{k.m.v.hee,z.liu3,n.sidorova}@tue.nl
[2] Queensland University of Technology, Australia
m.larosa@qut.edu.au
[3] NICTA Queensland Lab, Australia

Abstract. Process models in organizational collections are typically created by the same team and using the same conventions. As such, these models share many characteristic features like size range, type and frequency of errors. In most cases merely small samples of these collections are available due to e.g. the sensitive information they contain. Because of their sizes, these samples may not provide an accurate representation of the characteristics of the originating collection. This paper deals with the problem of constructing collections of process models from small samples of a collection, with the purpose to estimate the characteristics of this collection. Given a small sample of process models drawn from a real-life collection, we mine a set of *generation parameters* that we use to generate arbitrarily-large collections that feature the same characteristics of the original collection. In this way we can estimate the characteristics of the original collection on the generated collections. We extensively evaluate the quality of our technique on various sample datasets drawn from both research and industry.

1 Introduction

Today there are millions of business process models around [10]. They are used for various reasons such as documentation of business procedures, performance analysis, and process execution. Organizations have their own process model collections to describe their business procedures. For example, Suncorp, one the largest Australian insurers, maintains a repository of over 6,000 process models [9]. Also consultancy firms and software companies like SAP provide collections of "reference models" to develop customer-specific process models.

With the proliferation of process models, it comes a variety of tools to manipulate such models, e.g. process editors, simulation tools and conformance checkers. These tools, developed by both software vendors and academics, contain often sophisticated algorithms that use process models as input. In order to evaluate these algorithms, and in particular to determine their amortized performance, we use benchmarks. In this context, a benchmark is a fixed set of process models that is used to compare different algorithms. These benchmarks either come from practice or are manually constructed to have some extreme properties. For instance, benchmarks can be selected from various existing public process repositories. One of the first process repositories for research

purpose is Petriweb [4], and recently AProMoRe [10] was developed. Although benchmarking is a popular technique, there is a drawback: one can only compare algorithms with respect to the same set of process models and we have no idea how they behave on other models. This is one of the motivations for our research: we would like to be able to derive *statistically sound statements* over algorithms. In order to do so we have to define distributions over the set of all possible process models. We want such a distribution to reflect the modeling style of a company for which the performance estimation is made.

In this paper, we deal with the problem of generating stochastic process model collections out of small samples drawn from an existing (real-life) collection. The generated collections should be arbitrarily large and accurately reproducing the characteristics of the existing collection. We choose workflow nets as the language to model processes, but our method is not restricted to this particular model. For model generation, we use construction (model refinement) rules, applied iteratively, starting with the most primitive workflow net possible, consisting of one place only. The probability distribution for the selection of the next construction rule is one of our generation parameters. Another generation parameter is the number of refinement steps, being a random variable, independent of the rules themselves. We use a Poisson distribution there, but any other discrete probability on the natural numbers will do as well. In this way we can make an arbitrarily large collection from a small dataset.

To estimate the generation parameters from a small collection of models, we iteratively apply nine construction rules used in the generation process in the reverse direction—as reduction rules. Assuming that the sample was generated via these rules, we estimate the probability with which each rule was used. These parameters are then used to guide the generation of new process models from the sample. The generated collection can be used for benchmarking, and moreover, it can be used for determining characteristics of the original collection (or the collection the company will obtain if they continue to use their current modeling practices) with probabilistic accuracy. For example, we can determine the distribution of the longest or shortest path, or the deadlock probability. As a result, even a very small dataset, on which a direct accurate estimation of characteristic values is impossible, provides enough information for generating enough process models, allowing one to give precise characteristic values for the underlying stochastic collection. This sounds like magic but is very close to bootstrap estimation in classical statistics [13]. While a collection of, for example, five process models would give us just five numbers showing the length of the shortest path, making it difficult to estimate the distribution of the length of the shortest path over the whole collection, at the same time, these five models would provide enough information to estimate the generation parameters with a high precision (assuming that these five models are of a sufficiently large size). Then we can generate an arbitrarily large collection from the same distribution as the sample's one.

To make sure the generation parameters and the characteristics of the generated collections do correspond to those of the originating process model collection, we tested our approach on very large samples generated both by ourselves and extracted from real-life datasets. In the first set of experiments we generated a large collection with some parameters and we took small samples from this collection in order to estimate

the generation parameters. Since we knew the real parameters, we could determine the quality of the estimations, and they turned out to be very good. We also did this for the collections coming from practice. Here we also took small samples from reasonably large collections and we compared the characteristics estimated using our approach with the characteristics of the original collection as a whole. Here our approach also performed quite well.

The rest of the paper is organized as follows. Section 2 introduces some basic concepts. Section 3 describes how workflow nets can be generated using our rules. Section 4 shows how the generation parameters can be estimated. Sections 5 and 6 present our empirical results. Section 7 concludes the paper.

2 Preliminaries

Workflow nets are a subclass of Petri nets extensively applied to the formal specification and verification of business processes [1]. In addition, mappings exist between process modeling languages used in industry (e.g. UML ADs, EPC, BPMN, BPEL) and workflow nets. These mappings provide a basis for transferring the results outlined in this paper to concrete process modeling notations. In the following, we formally define Petri nets and workflow nets.

For a set S, $\mathbb{B}(S)$ denotes the set of all bags over S, i.e. $\mathbb{B}(S) : S \rightarrow \mathbb{N}$ where $\mathbb{N}$ is the set of natural numbers. With $[a^3, b^2, c]$ we denote a bag with 3 occurrences of a, 2 occurrences of b, and 1 occurrence of c. A sequence σ of length $l \in \mathbb{N}$ over S is a function $\sigma : \{1, \ldots, l\} \rightarrow S$, which we denote as $\sigma = \langle \sigma(1), \sigma(2), \ldots, \sigma(l) \rangle$. The set of all finite sequences over S is denoted as S^*.

Definition 1 (Petri net). *A Petri net is a 3-tuple $N = (P, T, F)$, where P and T are two disjoint sets of* places *and* transitions *respectively, $F \subseteq (P \times T) \cup (T \times P)$ is a* flow relation*. Elements of $P \cup T$ are the* nodes *of N, elements of F are the* arcs*.*

Given a node $n \in (P \cup T)$, we define its *preset* ${}^\bullet n = \{n' \mid (n', n) \in F\}$, and its *postset* $n^\bullet = \{n' \mid (n, n') \in F\}$.

Definition 2 (Workflow net). *A workflow net is a 5-tuple $N = (P, T, F, i, f)$ where (P, T, F) is a Petri net, $i \in P$ is the initial place, such that ${}^\bullet i = \emptyset$, $f \in P$ is the final place, such that $f^\bullet = \emptyset$, and each node $n \in P \cup T$ is on a directed path from i to f.*

Soundness is an important property of workflow nets [1]. Intuitively, this property guarantees that a process always has an option to terminate and that there are no dead transitions, i.e. transitions that can never be executed.

3 Generation of Petri Nets

In order to generate workflow nets, we employ a stepwise refinement approach with a number of construction rules. We first define the construction rules, and then introduce the approach of generation of workflow nets. Note that it has been proved in [7] that the construction rules enable us to generate any arbitrary Petri net.

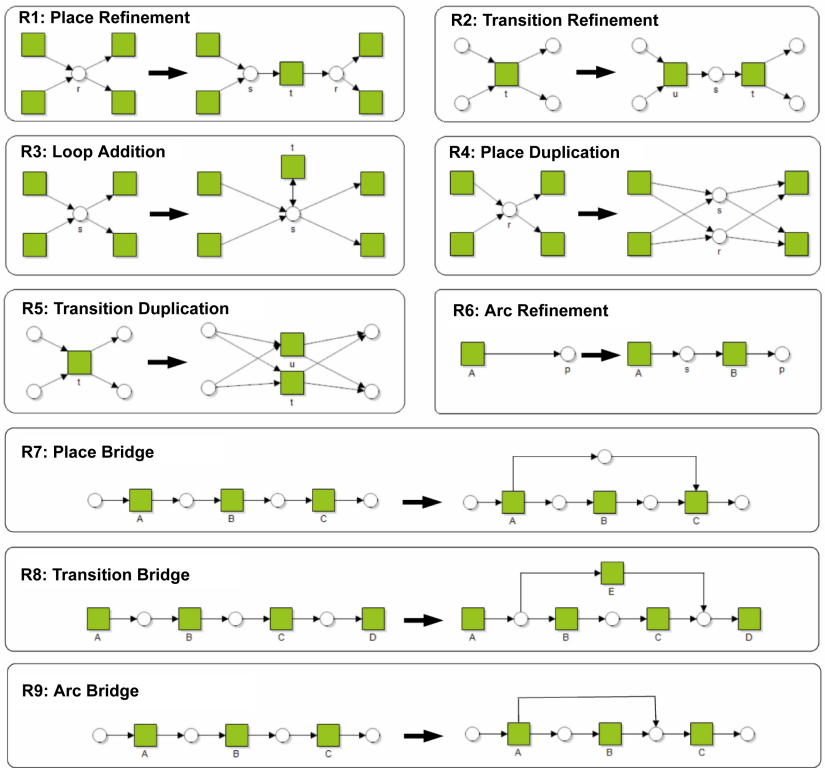


Fig. 1: Construction rules

3.1 Construction Rules

We consider a set of nine construction rules provided in Fig. 1. Rules $R_1, \ldots, R_5$ were studied by Berthelot in [2] and Murata in [12] as reduction rules that preserve liveness and boundedness properties of Petri nets. The rules are often called *Murata rules* (In fact Murata considered one more rule, a loop addition with a (marked) place, similar to R_3. We do not use this rule since it would destroy the soundness property). These rules are used in [5] to generate the so-called *Jackson nets*, which is a class of well-formed workflow nets. Besides Murata rules, we also propose rules $R_6, \ldots, R_9$ to generate Petri nets. Let N be the original net and R be one of the construction rules. If we apply R to N then we get a generated net N'. If we use R in the inverse direction, then we can get N again by reducing N'. Consequently, each rule R defines a binary relation φ_R on the set of nets. Let $\mathcal{N}$ be the set of all Petri nets, $\varphi_{R_i} \subseteq \mathcal{N} \times \mathcal{N}$. In our case we have $(N, N') \in \varphi_{R_i}$. In a similar way we define the construction rules as follows.

Definition 3 (Place refinement rule R_1). *Let $N, N' \in \mathcal{N}$. We say $(N, N') \in \varphi_{R_1}$ if and only if there exist places $s, r \in P'$, $s \neq r$ and a transition $t \in T'$ such that: $\bullet t = \{s\}, t\bullet = \{r\}$, $s\bullet = \{t\}, \bullet s \neq \emptyset$, $\bullet s \not\subseteq \bullet r$. The net N satisfies: $P = P' \setminus \{s\}$, $T = T' \setminus \{t\}$, $F = (F' \cap ((P \times T) \cup (T \times P))) \cup (\bullet s \times t\bullet)$.*

Definition 4 (Transition refinement rule R_2). *Let $N, N' \in \mathcal{N}$. We say $(N, N') \in \varphi_{R_2}$ if and only if there exist a place $s \in P'$ and transitions $t, u \in T'$, $t \neq u$ such that:*

$\bullet s = \{u\}, s\bullet = \{t\}$, $\bullet t = \{s\}, t\bullet \neq \emptyset$, $u\bullet \not\subseteq t\bullet$. *The net* N *satisfies:* $P = P^{'} \setminus \{s\}$, $T = T^{'} \setminus \{u\}$, $F = (F^{'} \cap ((P \times T) \cup (T \times P))) \cup (\bullet u \times s\bullet)$.

Definition 5 (Arc refinement rule R_3). *Let* $N, N' \in \mathcal{N}$. *We say* $(N, N^{'}) \in \varphi_{R_3}$ *if and only if there exist two nodes* $m, n \in P^{'} \cup T^{'}$, *such that:* $|\bullet m| = 1$, $m\bullet = \{n\}$, $|n\bullet| = 1$, $\bullet n = \{m\}$, $(\bullet m \times n\bullet) \cap F^{'} = \emptyset$. *The net* N *satisfies:* $P \cup T = (P^{'} \cup T^{'}) \setminus \{m, n\}$, $F = (F^{'} \cap ((P \times T) \cup (T \times P))) \cup (\bullet m \times n\bullet)$.

Definition 6 (Place duplication rule R_4). *Let* $N, N' \in \mathcal{N}$. *We say* $(N, N^{'}) \in \varphi_{R_4}$ *if and only if there exist two places* $s, r \in P^{'}, s \neq r$ *such that:* $\bullet s = \bullet r$, $s\bullet = r\bullet$. *The net* N *satisfies:* $P = P^{'} \setminus \{s\}$, $T = T^{'}$, $F = F^{'} \cap ((P \times T) \cup (T \times P))$.

Definition 7 (Transition duplication rule R_5). *Let* $N, N' \in \mathcal{N}$. *We say* $(N, N^{'}) \in \varphi_{R_5}$ *if and only if there exist two transitions* $t, u \in T^{'}, t \neq u$ *such that:* $\bullet t = \bullet u$, $t\bullet = u\bullet$. *The net* $P = P^{'}$, $T = T^{'} \setminus \{u\}$, $F = F^{'} \cap ((P \times T) \cup (T \times P))$.

Definition 8 (Arc refinement rule R_6). *Let* $N, N' \in \mathcal{N}$. *We say* $(N, N') \in \varphi_{R_6}$ *if and only if there exist two nodes* $m, n \in P' \cup T'$, *such that:* $|\bullet m| = 1$, $m\bullet = \{n\}$, $|n\bullet| = 1$, $\bullet n = \{m\}$, $(\bullet m \times n\bullet) \cap F' = \emptyset$. *The net* N *satisfies:* $P \cup T = (P' \cup T') \setminus \{m, n\}$, $F = (F' \cap ((P \times T) \cup (T \times P))) \cup (\bullet m \times n\bullet)$.

Definition 9 (Place bridge rule R_7). *Let* $N, N' \in \mathcal{N}$. *We say* $(N, N') \in \varphi_{R_7}$ *if and only if there exist one place* $s \in P'$ *and two transitions* $u, t \in T'$ *such that:* $\bullet s = \{u\}$, $s\bullet = \{t\}$. *The net* N *satisfies:* $P = P' \setminus \{s\}$, $T = T'$, $F = F' \cap ((P \times T) \cup (T \times P))$.

Definition 10 (Transition bridge rule R_8). *Let* $N, N' \in \mathcal{N}$. *We say* $(N, N') \in \varphi_{R_8}$ *if and only if there exist one transition* $t \in T'$ *and two places* $s, r \in P'$ *such that:* $\bullet t = \{s\}$, $t\bullet = \{r\}$. *The net* N *satisfies:* $P = P'$, $T = T' \setminus \{t\}$, $F = F' \cap ((P \times T) \cup (T \times P))$.

Definition 11 (Arc bridge rule R_9). *Let* $N, N' \in \mathcal{N}$. *We say* $(N, N') \in \varphi_{R_9}$ *if and only if there exist two nodes* $s, r \in P' \cup T'$, *such that* $(s, r) \in F'$. *The net* N *satisfies:* $P = P'$, $T = T'$, $F = F' \setminus \{(s, r)\}$.

In Figure 1 we only listed examples of the rules. For instance, in Rule R_7, we can also use a place to bridge transitions A and C from C to A, instead of from A to C as shown in the Fig. 1. We observe that different rules can generate the same structure. It is clear that the place refinement rule and the transition refinement rule can both generate sequential structures. Another example is shown in Fig. 2. In order to generate N', from N we can apply either the loop addition rule on the place s, or the transition duplication rule on the transition C.

It has been proved in [5] that the rules $R_1 \ldots R_5$ preserve the soundness property of workflow nets (with respect to a given marking). Therefore, all Jackson nets are sound. Moreover, in [8] the authors proved that Jackson nets are generalized sound, which is a more powerful property and important for refinements. It is easy to show by an example that the rules R_7, R_8, R_9 destroy the soundness property. For instance, in Fig. 3 after applying a place bridge rule from transition B to transition A to a sound net N, a deadlock is introduced in the refined net N' and N' is not sound anymore.

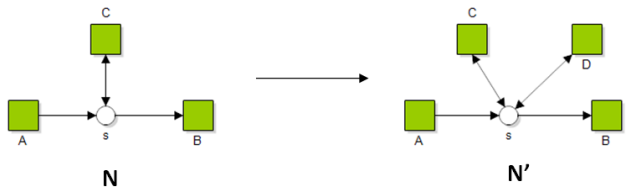


Fig. 2: Transition duplication rule and loop addition rule result in the same structure

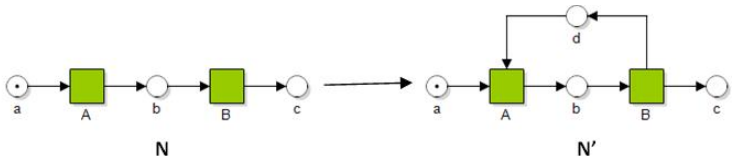


Fig. 3: Place bridge rule breaks soundness

3.2 Generation of Workflow Nets

In order to generate a workflow net, we adopt a stepwise refinement approach. Given two workflow nets N and N', N generates N' if and only if N' can be obtained from N by applying zero or more times of a construction rule, without applying any rules to the initial and the final places of N. Let $\mathcal{N}$ be the set of all workflow nets, $\varphi^* \subseteq \mathcal{N} \times \mathcal{N}$. Let $\mathcal{R}$ be the set of all construction rules. We define $(N, N') \in \varphi^* \Leftrightarrow \exists R \in \mathcal{R} : (N, N') \in \varphi_R \vee \exists N'' \in \mathcal{N}, \exists R \in \mathcal{R} : (N, N'') \in \varphi_R \wedge (N'', N') \in \varphi^*$.

Our approach for generating workflow nets has three determinants, or *generation parameters*: construction rules, probabilities of applying the rules, and the total number of times the construction rules are used per net, which is the stopping criterion and is called *size* (of the net) in this paper. We assume a Poisson distribution for the size, and take a random number as the size of a net accordingly.

Generation starts with one single place. Initially, the only applicable rule is the place refinement rule that results in the generation of the workflow net consisting of the initial place and the final place connected by one transition. In every subsequent step, we select a rule from the construction rules whose conditions hold, i.e. *enabled* rules. In order to select a rule from all the enabled rules to extend the net, we take a uniform distribution over all the enabled construction rules, e.g. all the enabled rules have the same probability to be chosen. The generation stops when the size has been reached. Thus, such a mechanism makes net generation a random process, and every particular net has certain probability to be produced. A formal definition of the probability space for a stochastic collection of process models can be found in the technical report [6].

4 Discovery of Generation Parameters

Workflow nets are generated by using a set of generation parameters. Once the generation parameters have been specified, we are able to generate an arbitrarily large set of

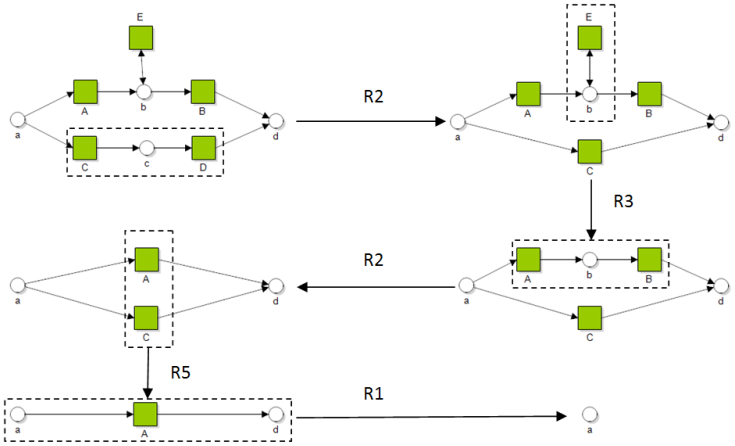


Fig. 4: An example of workflow net reduction

workflow nets (i.e. a "collection") where each net has certain probability to occur. We consider a set of workflow nets that we encounter as a sample of its collection. Given a sample of workflow nets, we would like to determine the collection to which the sample belongs by estimating the generation parameters of the collection from the sample nets. We call such an estimation procedure *generation parameter mining*.

To derive the generation parameters from a sample of nets, in our estimation procedure we employ a reduction process according to the construction rules $R1, \ldots, R9$. During reduction, in each step we apply a construction rule (in the inverse direction) as a reduction rule to reduce a net until we reach the initial net, which is a single place (see Section 3). We require that after each reduction the reduced net remains a workflow net. During reduction we apply the rules in the following order:

Step 1 Keep applying the refinement rules ($R1, R2$) to reduce a net until the net cannot be reduced any further by the refinement rules, then go to Step 2.

Step 2 Apply the duplication rules ($R4, R5$) to reduce the net. After one successful reduction either by $R4$ or by $R5$ go back to Step 1. If both rules cannot reduce the net, go to Step 3.

Step 3 Apply the loop addition rule ($R3$) to reduce the net. If the loop addition rule successfully reduces the net, then go back to Step 1. Otherwise go to Step 4.

Step 4 Apply the bridge rules ($R7, R8, R9$) to reduce the net. After one successful reduction by any bridge rule go back to Step 1. If none of the bridge rules can reduce the net, terminate.

Figure 4 displays an example of such a reduction process. During reduction we record the number of times we use each rule. We can see which rules were used at least once and for each applied rule we compute the overall probability of its application. In this way we estimate the generation parameters.

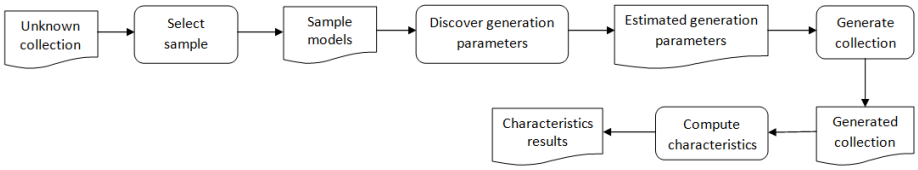


Fig. 5: Procedure of estimating characteristics of an unknown collection

Estimation of characteristics One of the reasons we consider collections of process models is because we want to determine characteristic properties of a collection based on its sample. Of course, it can be done on the sample directly, using available statistical methods. Typically the larger the sample size, the more accurate the collection characteristics can be estimated. If the sample size is not big enough, the characteristic estimations may not accurately reflect those of the whole collection. One way to boost accuracy is to increase the sample size. In our case, for any unknown collection, we estimate the generation parameters from a sample using the approach introduced in the early part of this section. Once we obtain these parameters, we can generate arbitrarily large samples. Consequently, based on the law of large numbers and the central limit theorem, we can estimate the characteristics of the original collection with any precision. This approach is illustrated in Fig. 5.

We tested our approach focusing on the following characteristics of a workflow net: 1) number of nodes, 2) average fanin and fanout, 3) length of the longest path, 4) length of the shortest path, 5) soundness probability, 6) deadlock probability, and 7) average number of strongly connected components. We compare the estimations with respect to their 1) mean, 2) standard deviation, and 3) confidence interval of the mean over the collection. Confidence interval of the mean is calculated as $[\overline{Y} - t_{(\frac{\alpha}{2},N-1)}\frac{s}{\sqrt{N}}, \overline{Y} + t_{(\frac{\alpha}{2},N-1)}\frac{s}{\sqrt{N}}]$, where $\overline{Y}$ is the sample mean, s is the sample standard deviation, N is the sample size, α is the desired significance level, $t_{(\frac{\alpha}{2},N-1)}$ is the upper critical value of the t-distribution with $N-1$ degrees of freedom.

5 Evaluation of the Estimation Quality

We tested our approach performing a series of experiments on collections of workflow nets generated as described in Section 3 using our plug-in to the ProM toolset [3]. To measure the quality of the generated collections, we considered various metrics. For the sake of space, we report the results on two sample metrics: *the length of the longest path* and *the soundness probability*.

Quality of the estimations of generation parameters We generated a collection of 100 workflow nets with a set of chosen generation parameters. We treat this collection as our original collection. Note that this original collection itself is a sample of the entire collection determined by the original generation parameters. We divided the original collection into 20 samples, 5 nets per sample. On each sample, we estimated

the generation parameters and thus obtained 20 sets of the generation parameters. Note that in practice we only have one sample, from which we mine the generation parameters. In this test we used 20 samples only for the purposes of testing. Fig. 6 lists the results of the original generation parameters (used to generate the original collection) and the estimated generation parameters (we consider the transition refinement rule and the place refinement rule together as they both generate sequential structures).

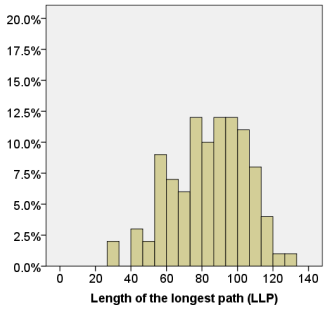
	Probability of R1 and R2	**Probability of R5**	**Probability of R4**	**Size mean**
original parameters	0.60	0.20	0.20	200
parameters' estimations 1	0.58	0.21	0.22	201
parameters' estimations 2	0.63	0.18	0.19	194
parameters' estimations 3	0.61	0.21	0.19	209
parameters' estimations 4	0.61	0.20	0.20	203
parameters' estimations 5	0.59	0.20	0.21	206
⋮	⋮	⋮	⋮	⋮
parameters' estimations 20	0.60	0.20	0.20	207
Avg.	0.60	0.20	0.20	201
Std.Dev.	0.0162	0.0168	0.0105	8.6601

Fig. 6: Original generation parameters and their estimations on samples

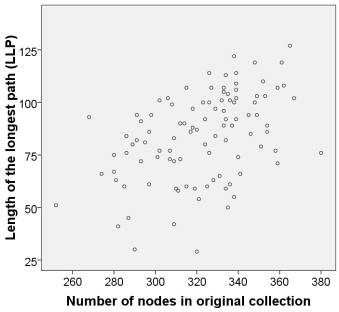
According to Fig. 6, the original construction rules and their probabilities are 60% for both the transition refinement rule and the place refinement rule, and 20% for the transition duplication rule, the place duplication rule and the mean value of size. The stopping criterion of net generation is 200. From each sample, we can always mine exactly the same construction rules as the original rules. On average, the probabilities of the construction rules estimated on all 20 samples are: 60% for both the transition refinement rule and the place refinement rule, 20% for the transition duplication rule, 20% for the place duplication rule. Compared against the probabilities of the original construction rules, we have got exactly the same probabilities as estimations. The average of the estimated mean value of size is 201, and this result is very close to the original value 200. Consequently, we succeeded in mining the generation parameters by using our generation parameter mining algorithm.

Distribution of the length of the longest path In this test we used the same collection of 100 workflow nets generated in Section 5, and treated it as the original collection. Using each set of the generation parameters' estimations (see Section 5), we generated a new collection of 100 workflow nets using our generation approach. Thus we generated 20 new collections. We used the generated collections to represent the original collection. In order to measure the quality of the generated collections, we considered the length of the longest path (LLP), which is a structural property of workflow nets, as our metric in this experiment. In a workflow net the longest path is a path between the initial place and the final place such that the total number of transitions on this path is maximized discarding loops. The LLP is the number of transitions on the path.

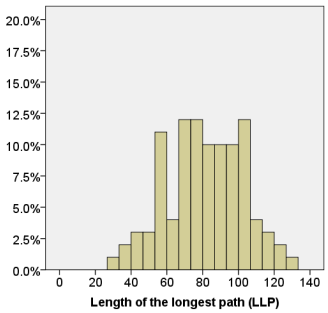
Figure 7 displays the histograms of the LLP in the original collection, one of the samples and a collection generated with the generation parameters computed on this sample. The figure also gives the scatter plot that shows that the correlation between the number of nodes and the LLP in the original collection is very weak. According to our experiment, it is difficult to find a strong estimator for the LLP, so we cannot



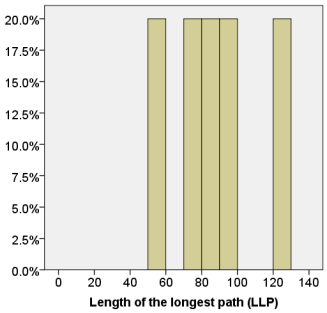
(a) Histogram of the LLP distribution in the *original* collection



(b) Scatter plot relating the number of nodes and LLP in the nets of the *original* collection



(c) Histogram of the LLP distribution in the *generated* collection



(d) Histogram of the LLP distribution in the *sample*

Fig. 7: Evaluation results for the LLP metric

compute the LLP based on another characteristic. We take the number of nodes as an example here. From the correlation plot we can observe that there is no strong correlation between the number of nodes and the LLP. Hence we cannot build a good model to compute the LLP from the number of nodes. In Fig. 7 we can observe that the histogram of the generated collection produces a distribution of LLP similar to the distribution that can be derived from the histogram of the original collection, while the histogram of the sample indicates a completely different distribution.

We further computed the 80th and the 90th percentiles (which are the values below which 80%, resp., 90% of the observations are found) for the LLP in the original collection and in all the collections generated. The results are listed in Fig. 8. The 80th and the 90th percentiles of the original collection are 102.00, resp., 107.90. On average, the 80th and the 90th percentiles of all the generated collections are 101.69, resp., 108.28. The Shapiro-Wilk test (see e.g. [11]) confirmed that we can assume that both the sample of the twenty 80th percentiles and the sample of the twenty 90th percentiles of the generated collections come from a normally distributed population, and taking into account the estimations of the standard deviations of these two distributions (6.88 and 6.51 respectively), we conclude that in 95% of cases the approximation error on the generated collection will not exceed 13.72% for the 80th percentile and 13.02% for the 90th percentile.

	80% **percentile**	90% **percentile**
original collection	102.00	107.90
generated collection 1	95.00	101.00
generated collection 2	103.80	112.00
generated collection 3	108.80	112.00
generated collection 4	104.80	111.90
generated collection 5	103.40	107.00
⋮	⋮	⋮
generated collection 20	100.80	108.00
Avg.	101.69	108.28
Std.Dev.	6.88	6.51

Fig. 8: 80th and 90th percentiles of LLP in the original collection and in the generated collections

Soundness probability In this experiment we considered soundness. Using a set of original generation parameters (different from the ones used in Section 5), we generated a collection of 50 workflow nets. We included the place bridge rule in the original generation parameters to introduce unsoundness (recall that bridge rules breaks soundness as illustrated in Section 3.1). Otherwise the nets would all be sound by construction. We regarded this collection as the original collection, and divided it into 10 samples, 5 nets per sample. From each sample we estimated the generation parameters from which we generated a new collection of 50 workflow nets. In order to measure the quality of the collections generated, we tested the probability of soundness, which is a behavioral property of workflow nets.

	Soundness Probability
original collection	46%
sample 1	20%
generated collection 1	50%
sample 2	60%
generated collection 2	44%
sample 3	20%
generated collection 3	42%
⋮	⋮
sample 10	60%
generated collection 10	46%
Avg. of samples	46%
Std.Dev. of samples	18.97
Avg. of generated collections	45.8%
Std.Dev. of generated collections	4.57

Fig. 9: Soundness probability

Figure 9 presents the results of soundness probability of the original collections, all samples and all generated collections. Accordingly, in the original collection 46% of nets are sound. On average, soundness probability of samples is 46%, and soundness probability of the generated collections is 45.8%. Again, the Shapiro-Wilk test confirmed that we can assume that the sample of the soundness probability estimated on the generated collections comes from a normally distributed population with the mean value of 45.8% and the standard deviation of 4.57%, implying that in 95% of cases we can expect that the estimation of soundness probability on a generated collection will

Table 1: Estimations of generation parameters from 10 samples of process models from practice

	Probability of R1 and R2	Probability of R5	Probability of R4	Probability of R3	Probability of R8	Size mean
1	0.69	0.22	0.01	0.03	0.06	32
2	0.73	0.20	0.01	0.06	0	14
3	0.84	0.10	0.05	0.02	0	13
4	0.85	0.08	0.06	0.02	0	13
5	0.85	0.12	0	0.03	0	15
6	0.72	0.04	0.06	0.09	0.09	14
7	0.72	0.11	0	0.07	0.1	12
8	0.74	0.18	0	0.02	0.06	17
9	0.76	0.02	0	0.08	0.15	12
10	0.73	0.12	0.07	0.03	0.05	23
Avg.	0.76	0.12	0.03	0.05	0.05	17
Std.Dev.	0.06	0.07	0.03	0.03	0.05	6.35

be at most 9.14%. Note that the standard deviation of the samples is 18.97%, implying a 4 times greater error factor than in the generated collections.

We repeated the same experiments along other metrics (i.e. average fanin and fanout, length of the shortest path, deadlock probability, and average number of strongly connected components) obtaining similar results. Hence we conclude that the generated model collections closely resemble the characteristics of the original collection, of which only a small, casually extracted sample was available.

6 Evaluation with Industry Models

In this series of tests, we tested our approach with process models from practice. We randomly extracted 50 process models from a collection of 200 models of a manufacturing company, which we acquired from the AProMoRe process model repository [10]. We assumed that these models could be generated by the generation mechanism in Section 3. We treated this collection as our original collection, and divided it into ten samples, five nets per sample. For each sample, we derived an estimation of the generation parameters and generated a collection of 200 nets based on these parameters. Thus we had ten generated collections, each having 200 nets. In order to test the quality of the generated collections, we considered various characteristics of workflow nets as metrics. Below, we report the results on the estimations of the generation parameters and two metrics: *length of the shortest path* (structural metric) and *soundness probability* (behavioral metric).

Estimation of generation parameters Table 1 lists the generation parameters estimated on each sample. Here we consider the transition refinement rule and the place refinement rule together as they both generate sequential structures.

In this experiment, models were obtained from practice, so we did not know the original generation parameters (the generation parameters used to generate the original

collection). In Table 1, the standard deviation of each generation parameter (bottom row in Table 1) is actually large. This is because models in different samples vary with respect to size (e.g. number of nodes) and structure. For instance, the means of size of sample 1 and sample 2 are 32 and 14, respectively, which means models in sample 1 are generally larger (e.g. in terms of number of nodes) than models in sample 2. There is no parallel structure in models in (e.g.) sample 5, as the place duplication rule was not used in generation (the probability of the place duplication rule is estimated as 0).

Distribution of the length of the shortest path In this experiment we considered the length of the shortest path (LSP). In a workflow net the shortest path is a path between the initial place and the final place such that the total number of transitions on this path is minimized. LSP is the number of transitions on the path. Fig. 10 lists the results of the LSP in the original collection and in all the generated collections. According to Fig. 10, the average LSP in the original collection is 7.14. On average, the LSP in all the generated collections is 7.29. Shapiro-Wilk test was performed to show that we can assume a normal distribution with the mean value of 7.29 and the standard deviation of 1.46 for the average LSP for the generated collections, implying that with a high probability the generated collections will approximate the original collection well with respect to the average LSP value.

	LSP
original collection	7.14
generated collection 1	7.34
generated collection 2	6.60
generated collection 3	6.31
⋮	⋮
generated collection 10	7.98
Avg.	7.29
Std.Dev.	1.46

Fig. 10: Average LSP

Let us zoom in one sample to measure the quality of our approach. Figure 11 displays the mean and the 95% confidence interval for the mean of LSP for the original collection, sample 1 and the generated collection 1.

The mean of sample 1 is 6. It falls outside the 95% confidence interval for the mean of the original collection, which ranges from 6.42 to 7.86. This confidence interval means that we are 95% sure that the real mean of the original collection falls between 6.42 and 7.86. Thus it is clear that we can poorly estimate this characteristic of the original collection from this sample. On the other hand, the mean of the generated collection 1 is 7.34. This value falls inside the 95% confidence interval for the mean of the original collection, and is also very close to the mean of the original collection, which is 7.14. The 95% confidence intervals for the mean of the original collection and that of the generated collection 1 almost fully overlap. Consequently, the generated collection 1 very well represents the original collection along LSP.

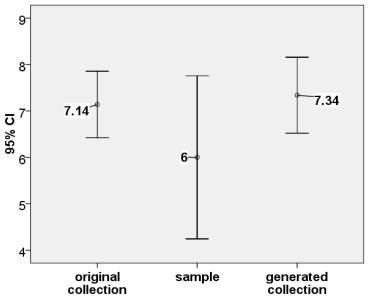


Fig. 11: Mean and 95% confidence interval of the LSP for original collection, sample 1 and generated collection 1

Soundness probability In this experiment we computed the probability of soundness for models in the 10 collections that we generated. Figure 12 lists the results of the soundness probability in the original collection and in all the generated collections. The

results show that all nets in the original collection are sound as the soundness probability of the original collection was 100% (this implies that all samples from this collection are 100% sound too). The soundness probabilities of the generated collections 1, 6, and 10 are 96%, 89%, and 85%, respectively (note that our net generation technique do preserve soundness in general), and the rest of the generated collections are all 100% sound. On average the soundness probability of all generated collections is 97%. The Shapiro-Wilk test showed that we cannot assume a normal distribution for the data in this case, however due to the low value of standard deviation in the distribution of soundness probability estimated on the generated collections, we conclude that the generated collections will allow to estimate the soundness probability well enough.

As shown in Table 1, we estimated the generation parameters (probability distribution for Murata's rules) on samples $2, 3, 4, 5$. The generated collections $2, 3, 4, 5$ consist of Jackson nets only, and these nets are (generalized) sound (see Section 3). To generate samples $7, 8, 9$, both the transition bridge rule and Murata's rules *without* place duplication rule are needed. These rules allow to generate state machine workflow nets only (i.e. workflow nets that do not allow for parallelism), as none of the rules can introduce a parallel structure. Because state machine workflow nets are always sound, the generated collections $7, 8, 9$ are 100% sound.

To reduce nets from samples $1, 6, 10$, both transition bridge rules and Murata's rules are necessary. Since the soundness property can be destroyed by the bridge rules (see Section 3), soundness is not guaranteed for the generated collections $1, 6, 10$. Remarkably, in spite of the source of unsoundness in the form of the bridge rules, the soundness probability on the generated collections $1, 6$ and 10 is still very high.

The results conducted with real-life process models confirm the results we obtained with the dataset that we generated artificially. From these results we can already state with some confidence that our generated model collections can closely reproduce the characteristics of a given collection, when only a small sample is available. Thus, the generated models can be used as benchmarks to test algorithms that operate on process models, and produce statistically valid results.

collection	**probability of soundness**
original collection	1
generated collection 1	0.96
generated collection 2	1
generated collection 3	1
generated collection 4	1
generated collection 5	1
generated collection 6	0.89
generated collection 7	1
generated collection 8	1
generated collection 9	1
generated collection 10	0.85
Avg.	0.97
Std.Dev.	0.05

Fig. 12: Soundness probability of original collection (from practice) and generated collections

7 Conclusion

We proposed an approach called *generation parameter mining* to discover a collection of process models based on a sample from that collection. We assume that nets can be generated by certain generation parameters by stepwise refinement. Such generation parameters consist of a set of defined construction rules, probabilities of applying the rules, and net size. Thus, we mine these generation parameters by reducing the sample

nets using the construction rules in the inverse direction. Once we obtain the generation parameters, we can use them to determine the whole collection of the given sample. To the best of our knowledge, there is no similar research to our work in this field yet.

There are a number of good reasons for discovering whole collections from samples. First, it is very difficult to estimate the characteristics of a collection accurately with a small sample size. Hence if we can identify the entire population, we are able to estimate the characteristics of that population at any level of accuracy by generating as many models as necessary from the population. Second, given that we can build arbitrarily large collections which faithfully reproduce the features of the original collection, our generated models can be used for benchmarking purposes.

We extensively tested our approach using both process models generated by our software tool and process models from practice. The results indicate that the generation parameters of an original collection can be successfully estimated on a small sample of the collection. In order to test the quality of the generated collections, we considered various process model characteristics. The results of these latter tests show that the generated collections accurately represent the original collection, whereas the original collection is poorly represented by the samples.

There are many interesting questions to explore in the future. We will further test our approach on more model collections from practice, and we will consider other characteristics e.g. deadlock probability, number of strongly connected components, maximal number of concurrent transitions. As we can get the same net by reducing a net using different rules, this may cause incorrect generation parameters from estimation. This issue becomes more problematic if bridge rules are involved. For instance, bridge rules break soundness: if bridge rules and their probabilities are estimated wrongly, the soundness probability of the population will be incorrectly estimated. Therefore, we intend to improve our approach by refining the rule mining phase.

Acknowledgments This research is partly funded by the NICTA Queensland Lab.

References

1. van der Aalst, W.M.P.: The Application of Petri Nets to Workflow Management. The Journal of Circuits, Systems and Computers 8(1), 21–66 (2001)
2. Berthelot, G.: Transformations and Decompositions of Nets. In: Advances in Petri Nets. vol. 254, pp. 360–376. Springer (1987)
3. van Dongen, B.F., Alves der Medeiros, A.K., Verbeek, H.M.W., Weijters, A.J.M.M., van der Aalst, W.M.P.: The ProM Framework: A New Era in Process Mining Tool Support. In: Applications and Theory of Petri Nets. vol. 3536, pp. 444–454. Springer (2005)
4. Goud, R., van Hee, K.M., Post, R.D.J., van der Werf, J.M.E.M.: Petriweb: A Repository for Petri Nets. In: Proc. of ICATPN. Lecture Notes in Computer Science, vol. 4024, pp. 411–420. Springer (2006)
5. van Hee, K.M., Hidders, J., Houben, G., Paredaens, J., Thiran, P.: On the Relationship between Workflow Models and Document Types. Information Systems 34(1), 178–208 (2008)
6. van Hee, K.M., La Rosa, M., Liu, Z., Sidorova, N.: Discovering Characteristics of Stochastic Collections of Process Models. BPM Center Report BPM-11-06, BPMcenter.org (2011)
7. van Hee, K.M., Liu, Z.: Generating Benchmarks by Random Stepwise Refinement of Petri Nets. In: Proc. of APNOC. pp. 30–44 (2010)

8. van Hee, K.M., Sidorova, N., Voorhoeve, M.: Generalised Soundness of Workflow Nets Is Decidable. In: Proc. of ICATPN. Bologna, Italy (2004)
9. La Rosa, M., Dumas, M., Uba, R., Dijkman, R.: Merging business process models. In: Proc. of CoopIS. pp. 96–113. Springer (2010)
10. La Rosa, M., Reijers, H.A., van der Aalst, W.M.P., Dijkman, R.M., Mendling, J., Dumas, M., Garcia-Banuelos, L.: AProMoRe: An Advanced Process Model Repository. Expert Systems with Applications 38(6) (2011)
11. Montgomery, D.C., Runger, G.C.: Applied Statistics and Probability for Engineers. Wiley & Sons, 5 edn. (2011)
12. Suzuki, I., Murata, T.: A Method for Stepwise Refinement and Abstraction of Petri Nets. Journal of Computer and System Sciences 27(1), 51–76 (1983)
13. Wu, C.F.J.: Jackknife, Bootstrap and Other Resampling Methods in Regression Analysis. Annals of Statistics 14(4), 1261–1295 (1986)