IRIS AperTO

UNIVERSITÀ DEGLI STUDI DI TORINO

AperTO - Archivio Istituzionale Open Access dell'Università di Torino

Variable order metrics for decision diagrams in system verification

This is the author's manuscript

Original Citation:

Availability:

This version is available http://hdl.handle.net/2318/1764225 since 2022-03-15T08:34:05Z

Published version:

DOI:10.1007/s10009-019-00522-6

Terms of use:

Open Access

Anyone can freely access the full text of works made available as "Open Access". Works made available under a Creative Commons license can be used according to the terms and conditions of said license. Use of all other works requires consent of the right holder (author or publisher) if not exempted from copyright protection by the applicable law.

(Article begins on next page)

29 March 2024

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

Variable order metrics for decision diagrams in system verification

Elvio G. Amparore · Susanna Donatelli · Gianfranco Ciardo

Received: date / Accepted: date

Abstract Decision diagrams (DDs) are widely used in system verification to compute and store the state space of finite discrete events dynamic systems (DEDSs). DDs are organized into levels and it is well known that the size of a DD encoding a given set may be very sensitive to the order in which the variables capturing the state of the system are mapped to levels, but computing optimal orders is NP-hard. Several *heuristics* for variable order computation have been proposed and *metrics* have been introduced to evaluate these orders. However, there appears to be no published evaluation of actual predictive power for these metrics.

We propose and apply a methodology to carry out such an evaluation, based on the correlation between the metric value of a variable order and the size of the DD generated with that order. We show correlation results for several metrics taken from the literature, applied to a large set of variable orders built using different approaches, for 40 fairly diverse DEDSs taken from the literature. Our experiments show that these metrics have correlations ranging from "very weak to non-existing" to 'strong".

We also provide evidence of the positive impact of highly correlating metrics on variable order heuristics by defining and evaluating two new heuristics (an improvement of the well-known FORCE heuristic and a metric-based simulated annealing), as well as a meta-heuristic (where a metric is used to drive the selection of the "best" variable order among a set of different variable orders).

This work was supported in part by the National Science Foundation under grant ACI-1642397

Elvio G. Amparore, Susanna Donatelli
Dipartimento di Informatica, Università di Torino
E-mail: amparore,susi@di.unito.it

Gianfranco Ciardo
Computer Science Department, Iowa State University, IA, USA
E-mail: ciardo@iastate.edu

Keywords Decision diagrams · variable order metric · variable order computation

1 Introduction

Decision diagrams (DDs), which include binary and multi-valued decision diagrams (BDDs [12] and MDDs [27]), are a powerful data structure to store large structured data and are used in many system verification tools [3, 13, 15, 34, 35] to store the state space and the transition system of the discrete-state system under study. It is well-known [12] that the size of the DD encoding the state space of a system can be exponentially affected by the order in which state variables are mapped to DD levels, and this exponential factor is often encountered in practical applications: indeed the User Manual of the widely used NuSMV tool [15, page 103] states "In particular, the order of variables is critical to control the memory and the time required by operations over BDDs". Unfortunately, finding an optimal order (yielding a minimal BDD for a given function) is an NP-complete problem [10]. Two standard approaches to mitigate the DD growth are to perform a *dynamic reordering* of the DD variables during its construction, typically when the memory requirements reach a given threshold, or to use a *static ordering* aimed at determining a "good" variable order before starting the DD construction, by either relying on the user, or applying heuristics. Dynamic and static approaches can be combined: if the chosen static order leads to a large DD, dynamic reordering can be applied during its construction. In this paper, we concentrate on static ordering, as it a required starting point and in general it is difficult to have an effective dynamic reordering if the initial order is poor (an "efficiency concern" of the developers of the DD library BuDDY [26]).

But how can we identify a good heuristic for variable ordering? Are there heuristics that work (almost) always well?

In previous work [5] we benchmarked 18 different algorithms to compute variable orders on a set of 386 DEDS, Petri net models from the Model Checking Contest (MCC) held at the annual Petri Net conference [31]. The results indicate that, while some algorithms tend to perform significantly better than others, we can always find DEDSs for which a "bad" algorithm finds the best order and performs much better than the "good" algorithms. This suggests that relying on a single algorithm is unwise; however, if we consider multiple algorithms, how do we choose a heuristic when tackling a new system? Given enough resources, we could concurrently build the DD for each candidate variable order, and retain only the best one (producing the smallest DD), but this is rarely feasible for DEDS with many variables, as the number of exhaustive orders for which to build the DD equals $|V|!$ if we have $|V|$ variables.

Variable order metrics can be a viable alternative, as we can simply select, among the candidate variable orders, the one giving the smallest value for the metric. Moreover, the value of such a metric can also be used as an objective function to devise new heuristics. Of course, this approach is effective only if the metric positively correlates with the size of the DD (the lower the value of the metric for a variable order, the lower the size of the DD generated with that order). This calls for the computation of a correlation coefficient (CC) between metrics and DD sizes. Thus, we investigate three research questions:

Q1 Which correlation coefficients are adequate to evaluate variable orders metrics, and on which set of orders should they be computed?
Q2 Do metrics have significantly different predictive power?
Q3 Can knowledge of each metric correlation help improve the state space construction of DEDS?

A first contribution of this paper is to define a methodology to define and evaluate the correlation between variable order metrics and the resulting DD size, and to evaluate the metrics on a statistically significant set of models. This requires the identification of a CC, a set of metrics, a set of models, and, for each of them, a set of variable orders, so that we can build the corresponding DDs and evaluate the correlation. The methodology takes into account the goal of computing a correlation: as we shall see, different objectives require us to consider different sets of variable orders on which the correlation is computed. We use metrics and models from the literature, and a set of orders that are either the full set (in the few cases where this is possible), the orders computed by a set of heuristics for variable ordering computation [5], a set of randomly generated orders, or a set of "modified" random orders. As far as we know this is the first work devoted to an extensive experimental evaluation of the predictive power of a significant set of metrics, answering questions Q1 and Q2 above.

A second contribution is the application of metrics correlation to define new heuristics and meta-heuristics. In particular, we show how to: 1) use one of the evaluated metrics to improve FORCE [1], an iterative procedure for variable order computation used in many verification tools; 2) define a simulated-annealing procedure that uses the metrics as the objective of the optimization; 3) use the CC of the metrics to drive a meta-heuristic that selects, among a set of available orders, one to use when building the DD. We show that even a moderately correlated metric can improve FORCE performance, and that the higher the index of correlation of the metric used in simulated annealing and the meta-heuristics, the better the computed variable orders. This suggests that developing new metrics with better correlation and adapting the computation of the CCs to the specific use of a metric can lead to better heuristics and meta-heuristics. This second contribution provides an answer to question Q3. To compare heuristics and meta-heuristics, we employ a notion of *score* that allows us to compare DDs even if they arise from DEDS with state spaces differing by orders of magnitude.

As with any experimental evaluation, we had to chose a setting that is significant and general enough, yet manageable. We employ the GreatSPN tool [3,2], therefore DEDS are specified as Petri nets and the decision diagrams considered are multivalued decision diagrams generated using the Meddly library [6]. Since our decision diagrams are in canonical form and we consider only the number of nodes in the DD once it is completely built (i.e., its *final* size), our results do not strictly depend on the specific library used. The models considered are taken from the annual model-checking contest [31]. Although some of the heuristics and metrics considered are tied to the system being specified as a Petri net, others are not. Moreover, our methodology is fully general and can be easily used in other contexts with different heuristics, metrics, or formalisms for system specification.

The paper is organized as follows: Section 2 introduces necessary background, Section 3 reviews the state of the art on variable order for decision diagrams, Section 4 presents the considered metrics, Section 5 defines the methodology proposed to evaluate the correlation coefficient of variable orders metrics, Section 6 applies the methodology to study the considered metrics on a large set of models, Section 7 presents three different applications of the evaluated metrics (improvement of FORCE heuristic, definition of a new heuristic based on simulated annealing, and definition of a meta-heuristic for variable order selection), and Section 8 concludes the paper and identifies future work.

2 Background

Discrete-state models. We assume a discrete-state model defined by a set V of discrete *variables* (w.l.o.g. taking val-

ues over the natural numbers $\mathbb{N}$) and a set E of events whose occurrence can modify these variables. Given event $e \in E$, let $V(e)$ be the set of variables that affect or are affected by e and, given variable $v \in V$, let $E(v)$ be the set of events that affect of are affected by v. In our experiments, we use Petri nets [37], where V is the set of places, E is the set of transitions, $V(e)$ is the set of places with input or output arcs connected to transition e, and $E(v)$ is the set of transitions with input or output arcs connected to place v. A state of the system is the collection of values assumed by the state variables and, given an initial state, the (reachable) state space of the model is defined as the set of states that can be reached from this initial state through a sequence of events.

Invariants. A model may be subject to *invariants* constraining the combination of values the state variables may assume in any reachable state. For Petri nets, P-semiflows have been extensively studied [37]: they capture an invariant constraint of the form $\sum_{v\in\pi} w_v \cdot v_x = constant$, where π is the set of places forming the *support* of the invariant, v_x the value of the variable in a state x, w_v a positive integer weight, and the equation is true for any reachable state x of the net. While computing P-semiflows can be expensive in pathological cases, it is usually quite fast for most Petri nets, so we assume that this information (in particular, the set Π of supports of the P-semiflows, i.e., the set of places involved in each P-semiflow) is available if needed.

Decision diagrams. We employ DDs (specifically, MDDs natively supported in the Meddly library [6], not encoded as BDDs) to store sets of states and transition relations between states, and generate the state space using the Saturation [14] algorithm. This requires us to specify a *variable order*, i.e., a bijection $l : V \to \{1,...,|V|\}$ between the state variables and the DD levels. We define $final(l)$ as the number of nodes in the DD encoding the state space of the model, when the levels are ordered according to l. The maximum number of nodes stored at any one time during state-space generation, $peak(l)$, is obviously at least as large as $final(l)$, and can sometimes be much larger, but here we concentrate on the final DD size, as the peak DD size is strongly influenced not only by the variable order but also by the technique employed for DD construction (e.g., breadth-first iterations vs. Saturation) and even on finer details such as the order in which the children of a DD node are explored.

Model set for the tests.

The test models are taken from the 817 Petri nets of the MCC model set [31]. These Petri nets are generated by setting different initial markings for a set of 77 different model structures. Since the correlation evaluation we present in this paper requires to generate thousands of DDs for each considered model, we only consider the subset of the 77 models for which we can build the DD of the state space for 10.000 different orders in less than 10×10.000 cpu seconds. This leads to the 43 *"test" models* listed in Section 6 (Table 3). Of these, we discard three models whose DD size is insensitive to the variable order, for a total of 40 considered models. Of these, we have identifies three *"easy" models* with $|V| < 10$ variables, for which we can build the DD for all the $|V|!$ variable orders.

Metrics. Let $\mathfrak{S}(V)$ be the set of permutations of the variables V. A *variable order metric* is a function $m : \mathfrak{S}(V) \to \mathbb{R}$ assigning a "goodness" to each variable order, where lower is better. Section 4 describes 12 such metrics, nine taken from [5], one from [42] and two new ones we define to capture invariant information, when available.

3 State of the art

Literature on metrics is limited and we do not know of any extensive evaluation of their predictive power. The next section presents a review of available metrics and defines some new ones. Here, we discuss instead if and how some well-known DD libraries and verification tools exploit metrics for variable ordering. The choice of the libraries and of the tools to survey is driven by their popularity, but we also include less popular Petri net tools that have been entered in the MCC, since they target the same set of models we consider.

We consider two dimensions in the state of the art: static vs. dynamic ordering and verification tool vs. DD library. As stated in the introduction, *static ordering* is a procedure to define a "good" variable order prior to actual DD construction, while *dynamic ordering* (or, better, reordering) is a procedure to improve the variable order during DD construction, typically to cope with an excessively large DD given the available memory. Procedures for dynamic ordering are typically provided at the *DD library* level, i.e., the computation of an improved order is provided by the same library that provides DD creation and manipulation, although it may be triggered by the *verification tool*, to set various parameters or to disable it altogether. Procedures for static ordering are instead typically provided by the verification tool, which allows the user to choose among a set of predefined heuristics or to load a pre-computed order, usually based on structural characteristics of the model.

DD libraries and dynamic reordering heuristics. Well-known DD libraries include CUDD [43] and BuDDy [26], while Meddly [6] and LibDDD [44] are more recent libraries that also cover new forms of DDs. A survey and a comparison of currently maintained DD libraries can be found in [17]. Of the DD libraries listed above, CUDD, BuDDY, and Meddly offer dynamic reordering, while LibDDD do not support this feature at the moment. CUDD is a widely

used library offering a rich set of dynamic reordering techniques: various forms of sifting (based on the seminal work of Rudell [38]), window permutation [20], a simulated annealing approach, a genetic algorithm one, and an exact approach based on dynamic programming. Sifting and window permutation are also included in BuDDy. Meddly provides both a variable reordering capability that changes the current DD variable order into one specified by the user, as well as the traditional dynamic variable ordering based on sifting. A characteristic of all dynamic reordering algorithms that we have studied is that the comparison among potential new orders is based on the actual size of the DD built using those orders, and no metric is used. Thus, the computational cost of dynamic reordering may be significant, even if the DD with the new order is generated efficiently as a modification of the DD with the current order, since the reordering is within the library itself. On the other hand, if the initial order is not good enough, reordering is the only choice.

Static ordering heuristics. Static variable order generation is typically performed using heuristics, specialized algorithms that, given a model, compute a "good" (although likely sub-optimal) variable order l using some criteria.

One of the most used heuristic is the iterative algorithm FORCE [1], but heuristics can be rather trivial, such as a simple *Depth-first* (DFS) or *Breadth-first* (BFS) visit of the DEDS structure, or can be based on some local proximity of the variables in the DEDS structure, like *Noack* (NOACK) and *Tovchigrecko* (TOV), defined in [23], or can exploit some structuring of the model, like *Gradient-P* (GP), defined in [4], *P-chaining* (PC) and *Markov Clustering* (MCL), both defined in [5]. A number of heuristics are built on the idea, first suggested in [36], that variable order optimization can be reduced to bandwidth optimization of a matrix accounting for the dependencies among DEDS variables.; these include *Sloan* (SLO), *Sloan-16* (SLO-16), *Advanced CM* (ACM), *King* (KING), *Gibbs-Poole-Stockmeier* (GPS), and *Cuthill-Mckee* (CM, when taken from the Boost-C++ library; CM2, when taken from ViennaCL library). Finally, any of these 14 heuristics can be used to compute an initial order for the FORCE iterations. We do not instead consider heuristic defined for circuits, which assume a clear input/output flow.

An in-depth analysis of the above heuristics is outside the scope of the paper, and more info can be found in [4,5, 23,36]. Here, we consider heuristics in relation to metrics, as often heuristic have the goal of minimizing a target metric, and to show the contribution that correlation knowledge can give to the definition of new or modified heuristics.

Variable order in verification tools. Many verification tools use DDs to build the state space and verify properties on it. Tools often include other verification approaches, but we focus on DD-based features and their variable order.

One of the best known DD-based tools is NuSMV [15], which uses the library CUDD. NuSMV allows the user to specify a static variable order in input. Alternatively, it can compute an order using a built-in heuristic that creates groups of variables with mutual dependencies, according to the definition of the transition relation for the DEDS, but the manual does not specify whether this grouping is based on some form of metric optimization. If no heuristic is used, the user may choose among six different ways of computing an order. There are three basic orderings: input variables are ordered *before* or *after* state and frozen variables, or all variables are ordered as they appear in the input file. The total is six since each ordering can be implemented in two ways: since NuSMV uses BDDs, variables are translated into bit sequences and all bits of a variable may appear in adjacent levels or interleaved. Again, it appears that no variable order metric is used. Dynamic reordering is also possible, based on the dynamic reordering provided by CUDD.

CADP [21] is a tool for the design and analysis of asynchronous concurrent systems. It includes symbolic verification using BDDs through the package caesar.bdd, which takes a Lotos specification, translates it into a "structured Petri net", and performs various structural analysis steps before checking properties on a symbolic state space built using the CUDD library. Variable ordering is only dynamic, again as provided by CUDD, and no metrics is considered.

ITS-Tools [44] features a symbolic model checker for DEDS models specified in a variety of formalisms, using GAL (Guarded Action Language) as common language. The tool uses the LibDDD library, which supports hierarchical DDs. The user can choose whether to specify a variable order, use a lexicographic order, or to run FORCE. Since FORCE has an associated metrics, we can say that there is some consideration for metrics.

LTSMin [36] also features a language-independent symbolic model checker built on the Sylvane DD library [18], designed to support multi-core execution at the DD level. LTSMin includes various heuristics for static variable order (SLO, KING, CM, CM2, ACM, GPS, and FORCE).

Tina [8] is a toolbox for (timed) Petri net verification. It includes a symbolic verifier, Tina.tedd, based on a built-in library for hierarchical DDs that includes several variable ordering heuristics based on "net traversal, semiflows, flows, or names" [32]. FORCE can also be run to (possibly) improve the obtained orders. Unfortunately, no further information is available about the Tina.tedd heuristics, and the tool is not open-source, thus it is not known whether the heuristics are based on a metric optimization.

Marcie [39,23] is a symbolic model checker for Petri nets based on a built-in implementation of Interval DDs. It includes two heuristics based on the Petri net structure (NOACK and TOV), as well as FORCE. There is no explicit metric associated to the two heuristics: the algorithms are

greedy and add a variable at each step according to its "distance" (defined differently by NOACK and TOV) from the variables already added.

GreatSPN [3] is another Petri net tool featuring a symbolic verifier, based on the library Meddly. It includes a large set of variable orderings. In particular, since it is the tool we use for our experiments, it implements all heuristics and metrics considered in this paper.

SMART [13] is a tool for logic, timing, and stochastic analysis, which includes a symbolic model checker built on top of the Meddly [6] library, and uses Petri nets as its main input type. Several static variable ordering approaches are provided by SMART: simulated annealing to find orders with small values for several of the metrics used in this paper (in particular SOUPS [42]), as well as FORCE.

None of the dynamic reordering techniques provided by Meddly is exploited in SMART, nor in GreatSPN, as dynamic variable reordering is not easily integrated with Saturation [14], the default algorithm for state-space generation and model checking in both SMART and GreatSPN.

ITS-Tools, LTSMin, Tina.ted, Marcie, GreatSPN, and Smart are model checkers that competed in the MCC 2017 contest (whose models we use for our experiments).

4 The metrics considered

In this section we briefly describe the 12 metrics considered. Two of them are newly defined (PSF and PTS^P), while the other 10 are taken from the literature.

The NES, WES, and SOT metrics. The *Normalized sum of Events Spans* (NES) is a simple metric based on event "spans" [40]. Given an event e and a variable order l, let $top_l(e) = \max_{v \in V(e)} l(v)$ and $bot_l(e) = \min_{v \in V(e)} l(v)$ be the maximum and minimum levels of variables on which e depends. The event span $span_l(e)$ of e for variable order l is then $top_l(e) - bot_l(e) + 1$. The NES score is obtained as the sum of the span of each event, normalized by the number of variables and events:

$$\mathrm{NES}(l) = \frac{1}{|E| \cdot |V|} \sum_{e \in E} span(e).$$

The *Weighted Normalized sum of Event Spans* (WES) is a variation [40] of NES to account for the specificity of a state space generation based on Saturation [14]: events closer to the top of the variable order have more weight, since Saturation performs better when most events have spans positioned close to the bottom of the order:

$$\mathrm{WES}^{\alpha}(l) = \frac{1}{|E| \cdot |V|} \sum_{e \in E} span(e) \cdot \left(\frac{2 \cdot top(e)}{|V|} \right)^{\alpha},$$

where α is suggested to be 1. A third metric, again based on the events, is the *Sum of Tops* (SOT). The SOT metric [13] is defined as the sum of the $top_l(e)$ values of each event. The intuition is that the top level at which an event is activated is a key factor.

The PSF metric. The work in [4] defines a heuristic called *Gradient-P* (GP), which generates variable orders by concatenating P-semiflows. The paper shows that GP produces good variable orders. We have therefore extracted from GP a metric, named *P-semiflows spans* metric (PSF), implicitly minimized by GP. The PSF value for variable order l is

$$\mathrm{PSF}(l) = \sum_{\pi \in \Pi} \Big(\max\{ l(v) \mid v \in \pi \} - \min\{ l(v) \mid v \in \pi \} + 1 \Big).$$

For models with no P-semiflows, or with just one P-semiflow covering the entire set of variables ($\pi = V$), the PSF metric has the same value for any order l, thus it is not meaningful.

The PTS metric. Another metric playing an important role in variable ordering algorithms is the *point-transition spans* (PTS), which the FORCE algorithm [1] uses as convergence criterion. FORCE is an iterative algorithm for multi-dimensional clustering of graphs that has been adapted to variable order generation. The *center of gravity* of event e is $cog_l(e) = \frac{1}{|V(e)|} \sum_{v \in V(e)} l(v)$, while the *hyper-position* of variable v is $p_l(v) = \frac{1}{|E(v)|} \sum_{e \in E(v)} cog_l(e)$. The PTS value of the variable order l is then

$$\mathrm{PTS}_l = \sum_{e \in E} \sum_{v \in V(e)} |cog_l(e) - p_l(v)|.$$

FORCE iteratively generates a new variable order $l^{(i+1)}$ from a variable order $l^{(i)}$ based on the hyper-position $p_{l^{(i)}}(v)$ of each variable, continuing while $\mathrm{PTS}_{l^{(i+1)}} < \mathrm{PTS}_{l^{(i)}}$. As Section 7.2 shows, and as confirmed in the literature, the initial variable order $l^{(0)}$ plays a significant role.

The PTS^P metric. We introduce a variation of PTS called PTS^P, which considers both the events and the P-semiflows. For a P-semiflow $\pi \in \Pi$, let $cog_l(\pi) = \frac{1}{|\pi|} \sum_{v \in \pi} l(v)$. Then

$$\mathrm{PTS}^P{}_l = \mathrm{PTS}_l + \sum_{\pi \in \Pi} \sum_{v \in \pi} |cog_l(\pi) - p'_l(v)|$$

where $p'_l(v) = \frac{1}{\Pi(v)} \sum_{\pi \in \Pi(v)} cog_l(\pi)$, and $\Pi(v)$ is the subset of P-semiflows with variable v in their support. The PTS^P metric can be used inside the FORCE heuristic instead of the PTS metric. We study this variation in Section 7.2.

Bandwidth reduction (BR) metrics: Bandwidth, Profile, and Wavefront. BR metrics have been recently applied [36] to variable order evaluation. They are defined over a symmetric $N \times N$ matrix $\mathbf{A}$, related to the model. In this paper we adopt the definition of $\mathbf{A}$ in [5], where $N = |V|$ and $A_{i,j}$ is nonzero iff there is at least an event that connects variables i and j. Other ways of defining $\mathbf{A}$ can be found

in [36]. The i-th bandwidth $\beta_i(\mathbf{A})$ of $\mathbf{A}$ is defined as: $\beta_i(\mathbf{A}) = i - \min\{j \mid \mathbf{A}_{i,j} \neq 0\}$. The *bandwidth* (BW) and the *profile* (PROF) of $\mathbf{A}$ are then defined as:

$$\text{BW}_{\mathbf{A}} = \max\{\beta_i(\mathbf{A}) \mid 1 \leq i \leq N\}, \qquad \text{PROF}_{\mathbf{A}} = \sum_{i=1}^{N} \beta_i(\mathbf{A}).$$

Also important for BR methods is the *wavefront* of the j-th column, $\omega_j(\mathbf{A}) = \{k \mid k > j \wedge \exists l \leq j \text{ s.t. } \mathbf{A}_{k,l} \neq 0\}$. Three metrics are defined: the *average wavefront* (AVGWF), the *maximum wavefront* (MAXWF) and the *root-mean-square wavefront* (RMSWF):

$$\text{AVGWF}_{\mathbf{A}} = \frac{1}{N}\sum_{j=1}^{N} \omega_j(\mathbf{A}), \quad \text{MAXWF}_{\mathbf{A}} = \max_{1\leq j\leq N} \omega_j(\mathbf{A}),$$

$$\text{RMSWF}_{\mathbf{A}} = \sqrt{\sum_{j=1}^{N} \frac{\omega_j(\mathbf{A})^2}{N}}.$$

We use Boost-C++ [11] to compute these functions. Some of our 28 heuristics are connected to BR metrics: Sloan is a profile and wavefront minimization algorithm [41], CM and KING minimize the bandwidth [16,30], while GPS minimizes both the bandwidth and the profile [22]. It is therefore of interest to understand if minimizing these metrics reduces the DD size; a preliminary analysis can be found in [28].

The Sum of Unique and Productive Spans (SOUPS) metric. The metrics discussed so far can be fooled by introducing additional copies of an event e (which has then greater impact, even if these copies affect neither the state space nor the DD size) or a new event e' that tests but does not change the state, e.g., a transition with the same input and output arcs in a Petri net (which does not lead to new states, but affects the metric, incorrectly suggesting that the variables in $V(e')$ should be close to each other).

While these examples are of course extreme and easily recognizable, cases where multiple events have the same effect on some variables or where an event is affected by some variables but does not change them are common. The SOUPS metric is then a variation of NES that considers only the *unique* and *productive* portions of the event spans: common portions of the effect of different events are counted once, and only the portion of the span of an event leading to the creation of new substates is counted. This variation not only aims at solving the two abovementioned anomalies, but also reflects the way in which the cache in a DD library reduces the cost of DD operations. The algorithm to compute the SOUPS metric is somewhat involved and its description is beyond the scope of this paper, but its computational complexity is still low. More details are available in [42].

5 Correlation evaluation: the methodology

To study the correlation of metrics and DD size we need to define what a "good" metric is. The scope of our evaluation is delimited by our desire to employ metrics for two purposes: to choose the "best" order among a set of precomputed orders, and to select the best metric for new heuristics based on metric minimization. We then say that metric m is a *perfect* predictor of the (final) DD size if

$$m(l_1) \leq m(l_2) \;\Rightarrow\; \mathit{final}(l_1) \leq \mathit{final}(l_2) \tag{1}$$

for any $l_1, l_2 \in \mathfrak{S}(V)$. As far as we know, no efficiently computable metric satisfies (1); in other words, we can assess the quality of different variable orders only a-posteriori, by generating the state space and observing the size of its DD encoding, but this is not of practical use when wanting to analyze a given model. However, a metric m satisfying (1) *in most cases*, i.e., such that $m(l)$ has high correlation with $\mathit{final}(l)$, could lead to good orders *in most cases*.

Our objective is to evaluate to which extent a metric respects the monotonicity expressed by Eq. 1, and to this end we can use the *Spearman rank correlation coefficient* [25]. The Spearman rank CC estimates, for a metric m, the correlation among $m(l)$ and $\mathit{final}(m)$ by computing the weighted Pearson correlation $\rho_{X,Y,W}$ (see Appendix A for a complete definition), among the *ranked lists* of metric values (X) and DD sizes (Y) with sample weights W (equal to 1 if left specified). Thus, the Spearman CC quantifies how well a metric that has the i-th largest value of the metric correlates with the i-th largest value of the DD size.

Given a set of variable orders $\mathscr{V} \subseteq \mathfrak{S}(V)$, we indicate with $S_m(\mathscr{V}) = \{\langle m(l), \mathit{final}(l)\rangle \mid l \in \mathscr{V}\}$ the bivariate series of the metric value and the size of the DD, for each specific order $l \in \mathscr{V}$, and with $\hat{\rho}_m(\mathscr{V})$ the CC for metric m computed from $S_m(\mathscr{V})$.

We can define different ways of computing the correlation coefficient depending on the set of variable orders considered. Ideally, we would like to compute the exact coefficient ρ_m using the entire $\mathfrak{S}(V)$ dataset, but this is feasible only for small models, since $|\mathfrak{S}(V)| = |V|!$; thus, for larger models, we must resort to computing the correlation coefficient $\hat{\rho}_m(\mathscr{V})$ for some appropriately chosen subset $\mathscr{V} \subseteq \mathfrak{S}(V)$.

5.1 Data set selection

We consider five strategies to generate $\mathscr{V}$:

1. $\mathscr{V}_{\text{EXH}} = \mathfrak{S}(V)$**, the *exhaustive* set** of orders. This is feasible only for small models.
2. $\mathscr{V}_{\text{RND}}$**, a set of *random* orders**. The set of orders is generated by randomly assigning variables to order positions, based on standard algorithms to generate random

permutations. Ideally, we would like to generate a set of random orders that statistically represents the distribution of the DD size for the entire set $\mathscr{V}_{\text{EXH}}$, but this is unfeasible, since this distribution is not known.

3. $\mathscr{V}_{\text{SEL}}$**, a set of *selected* orders**. We consider the variable orders obtained by running the 28 heuristics listed in Section 2. These orders are qualitatively different from an average random order, as one would hope they tend to be "better than random".
4. $\mathscr{V}_{\text{IMPR}}$**, a set of *improved* orders**. Unfortunately, random orders are almost always quite poor for most models. Worse yet, as we will show, even a large number of random orders does not cover the range of DD sizes for most models, since we tend to altogether miss any of the good orders that result in small DDs. Therefore, we introduce a variation process that, given variable order $l^{(i)}$, attempts to find an *improved* variable order $l^{(i+1)} = h(l^{(i)})$ satisfying $\mathit{final}(l^{(i+1)}) < \mathit{final}(l)$, using a *mutation* function h. An *improved variable order sequence* $t(l) = l^{(0)}, l^{(1)}, \ldots, l^{(n)}$ is a sequence of improved variable orders, such that each order $l^{(i)}, i > 0$ in the sequence is known to be an improvement over $l^{(i-1)}$, and $l^{(0)} = l$. Given a set of initial orders $\mathscr{V}'$, $\mathscr{V}_{\text{IMPR}}$ is the set of orders $\{k \mid k \in t(l) \wedge l \in \mathscr{V}'\}$, that is, the orders appearing in any of the sequences starting from the elements of $\mathscr{V}'$. We indicate with $T_{\max}$ the maximum length among all computed sequences: $T_{\max} = \max_{l \in \mathscr{V}'} \{|t(l)|\}$.
5. $\mathscr{V}_{\text{BEST}}$**, the set of *best observed* orders**. This includes the last variable order generated for each variable order sequence $t(l)$ considered for the generation of $\mathscr{V}_{\text{IMPR}}$.

To generate $\mathscr{V}_{\text{IMPR}}$, we adopt a procedure similar to that proposed in [9] which, at each call, produces a different order where a randomly chosen variable v in position x is moved to a new random position $y \neq x$ without changing the relative order of the other variables. The following algorithm generates a sequence $t(l) = l^{(0)}, l^{(1)}, \ldots, l^{(n)}$, where function h is as in the sifting algorithm of [38]:

```
for i from 1 to max_sequence_length
  for n from 1 to max_attempts
    l' ← h(l^(i))    // New candidate variable order
    if final(l') < final(l^(i))
      l^(i+1) ← l'
      start new iteration of the outer loop
  could not improve l^(i), end the sequence
```
(2)

To contain computational costs, our experiments consider 10 000 random orders ($|\mathscr{V}_{\text{RND}}| = 10\,000$), while, for $\mathscr{V}_{\text{IMPR}}$, we populate $\mathscr{V}'$ with 1 000 random orders, and fix $\mathit{max_sequence_length} = 300$ and $\mathit{max_attempts} = 50$ (the generation of a single sequence $t(l)$ ends by either failing $\mathit{max_attempts}$ attempts at improving an order, or when the sequence reaches length $\mathit{max_sequence_length}$).

Figure 1 shows the bivariate series for the MCC model "SmallOperatingSystem-PT-MT0016DC0008" and the NES metric, described in Section 4. The model has nine places, $|V| = 9$, therefore there are $|\mathscr{V}_{\text{EXH}}| = 9! = 362\,880$ possible variable orders. The state space has 16 587 states. Plots in Figure 1 depict, left to right, the series for $\mathscr{V}_{\text{EXH}}$, $\mathscr{V}_{\text{RND}}$, $\mathscr{V}_{\text{IMPR}}$, and $\mathscr{V}_{\text{BEST}}$ respectively. The DD sizes are plotted on a logarithmic scale and, in all plots, red crosses identify the bivariate series for $\mathscr{V}_{\text{SEL}}$.

Plot [A] shows $S_{\text{NES}}(\mathscr{V}_{\text{EXH}})$, the bivariate series relating the metric NES with the final DD size for all possible variable orders. For this model, the DD size can differ by two orders of magnitude. Plot [B] shows the $S_{\text{NES}}(\mathscr{V}_{\text{RND}})$ bivariate series for 10 000 random samples[1]. Plot [C] shows the $S_{\text{NES}}(\mathscr{V}_{\text{IMPR}})$ bivariate series, starting from the first 1 000 random orders of $\mathscr{V}_{\text{RND}}$. In this case $|\mathscr{V}_{\text{IMPR}}|$ has 5 906 samples, generated using Algo. 2 by performing 85 474 attempts, thus computing 85 474 DDs. The value of $T_{\max}$ is 14. Plot [D] shows the $S_{\text{NES}}(\mathscr{V}_{\text{BEST}})$ bivariate series, which therefore includes 1 000 orders (the final points of the paths built by Algo. 2) plus the 28 orders computed by the considered heuristics. We observe that different plots have a different coverage of the red crosses, since the red crosses, resulting from heuristics for variable order computation, tend to be located closer to the minimal value of the DD size.

This phenomenon, and the relationships between the various orders, are better observed on models with a larger number of variables, although on such models the cost of building $|\mathscr{V}_{\text{EXH}}|$ DDs is too high, thus $\mathscr{V}_{\text{EXH}}$ is not part of the comparison. Figure 2 shows the bivariate series for the MCC model "CircularTrains-PT-012". This model has 24 places, thus its 6.24×10^{23} possible variable orders cannot be considered exhaustively. The model has many variables, but only 195 states and the size of the observed DDs varies from one hundred nodes to a few thousand nodes. In that case, an upper bound of the DD size can be computed as

$$\sum_{k=1}^{L} \min(N, \prod_{l=k}^{L-1} n_l, 2^{\prod_{l=1}^{k} n_l} - 1) = 3\,003 \text{ nodes}$$

although such a DD with 3 003 nodes could not exists for any variable order. In practice, the worst observed DD for this model has a size of 2 046 nodes. Again, red crosses identify the $S_{\text{NES}}(\mathscr{V}_{\text{SEL}})$ bivariate series. Plot [A] shows the $S_{\text{NES}}(\mathscr{V}_{\text{RND}})$ for 10 000 random samples. Each blue point is a pair $\langle m(l), \mathit{final}(l) \rangle$, with the value of the metric on the y axis, and the final DD size on the x axis (with a logarithmic scale). We can easily observe that random orders are concentrated in a "bad" area of the plot, whereas the $\mathscr{V}_{\text{SEL}}$ tend to be much better. In other words, Plot [A] shows a case where random orders do not provide a good coverage of the

[1] It could be slightly less than 10 000, since the same permutation may be randomly generated multiple times, but this is extremely unlikely except for small models.

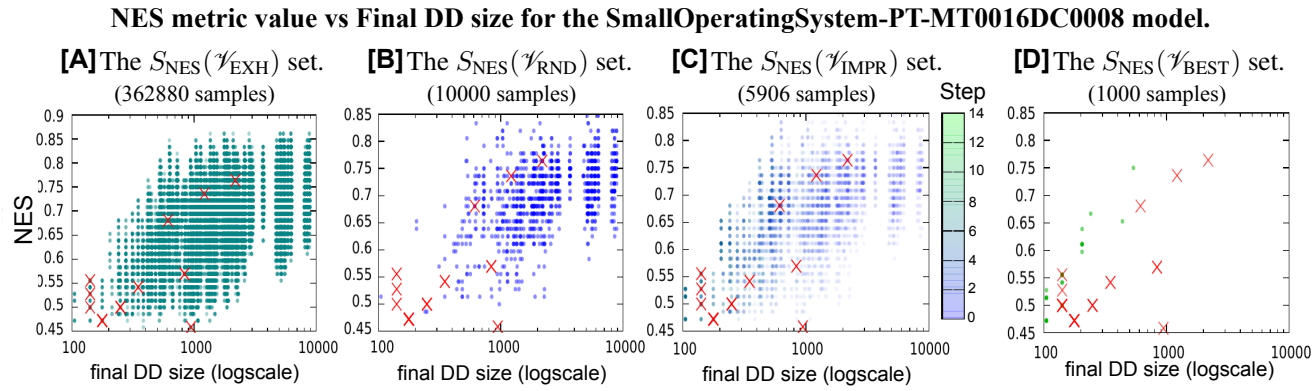


Fig. 1 The bivariate series for the various $\mathcal{V}$ orders for a small model.

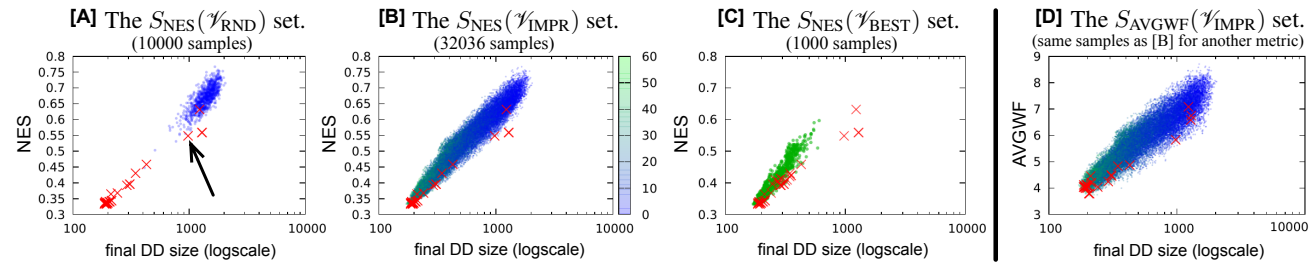


Fig. 2 The random and improved bivariate series for the CircularTrains-PT-012 model.

sample space, because almost all the random samples are "bad" orders. Observe that this might have undesired consequences when evaluating a heuristic: if we consider the set of orders in $\mathcal{V}_{RND}$ to evaluate the heuristic indicated with the arrow in plot [A], then we could conclude that this is a good heuristic, but the comparison with the DDs generated by either $\mathcal{V}_{IMPR}$ (plot [B]) or $\mathcal{V}_{SEL}$ (red crosses in all plots) or with $\mathcal{V}_{BEST}$ (plot[C]) indicate that this heuristic has built a fairly poor order. Thus, having an indication of the minimal size of a DD for a given model is important when comparing different heuristics.

In Section 7, heuristics will be compared on a "score": a measure of the size of the DD produced by a given heuristic relative to known minimum and maximum DD sizes for a given model. Our experiments often show that "good" orders are rare and hard to find by random sampling, thus we cannot build a statistically significant sample because we have no clue about the distribution of the orders and because we need to limit the number of random samples considered to 10 000 orders to contain computational costs (for this model, the ratio between $|\mathcal{V}_{EXH}|$ and $|\mathcal{V}_{RND}|$ is over 10^{19}). Plot [B] shows the plot for $S_{NES}(\mathcal{V}_{IMPR})$, which spans a wider portion of the sample space. The first 1 000 random orders generated for $\mathcal{V}_{RND}$ (shown in blue) are used as starting orders for the improved sequences, while improved random orders are shown in a color that progressively fades to green. In this model T_{max} is 60, and $|\mathcal{V}_{IMPR}| = 32\,036$. $\mathcal{V}_{IMPR}$ appears to be more representative than $\mathcal{V}_{RND}$ because it covers a wider area, and it is certainly more informative about "good" orders. However, for the purpose of approximating of the exact correlation, $\mathcal{V}_{IMPR}$ suffers from a bias problem, as discussed later in this section. Plot [C] shows the results for $S_{NES}(\mathcal{V}_{BEST})$, which includes 1 000 orders. The comparison with plot [A] indicates that, while only a few random orders produce DDs of less than 1 000 nodes, the (expensive) computation of $\mathcal{V}_{IMPR}$ allows us to retain a set of best orders producing DDs that are always below (or significantly below) 1 000 nodes.

Had we used a different metric, we would have obtained a different distributions of the samples along the y axis, but not on the x axis, since the final DD size is common to all bivariate series, independently of the metric. This is because the improvements made by Algo. 2 are not driven by any specific metric. For instance, plot [D] shows the bivariate series set $S^{AVGWF}(\mathcal{V}_{IMPR})$ for the AVGWF metric, and the same set of variable orders $\mathcal{V}_{IMPR}$. The plot has a different spread of samples from the plot [B] that uses the NES metric.

To better understand the relationships among the set of variable orders and the how CCs computed on those sets could be used, it is helpful to return to the "easy" model, for which we can consider all possible variable orders. The plots in Figure 3 [A], [B], [C], and [D] are the same as in Figure 1. Plots [E], [F] and [G] show the histogram of the distribution of the final DD sizes obtained from the $\mathcal{V}_{EXH}$, $\mathcal{V}_{RND}$, and $\mathcal{V}_{IMPR}$ datasets, respectively. For each of the three sets of variable orders, the corresponding histogram has the

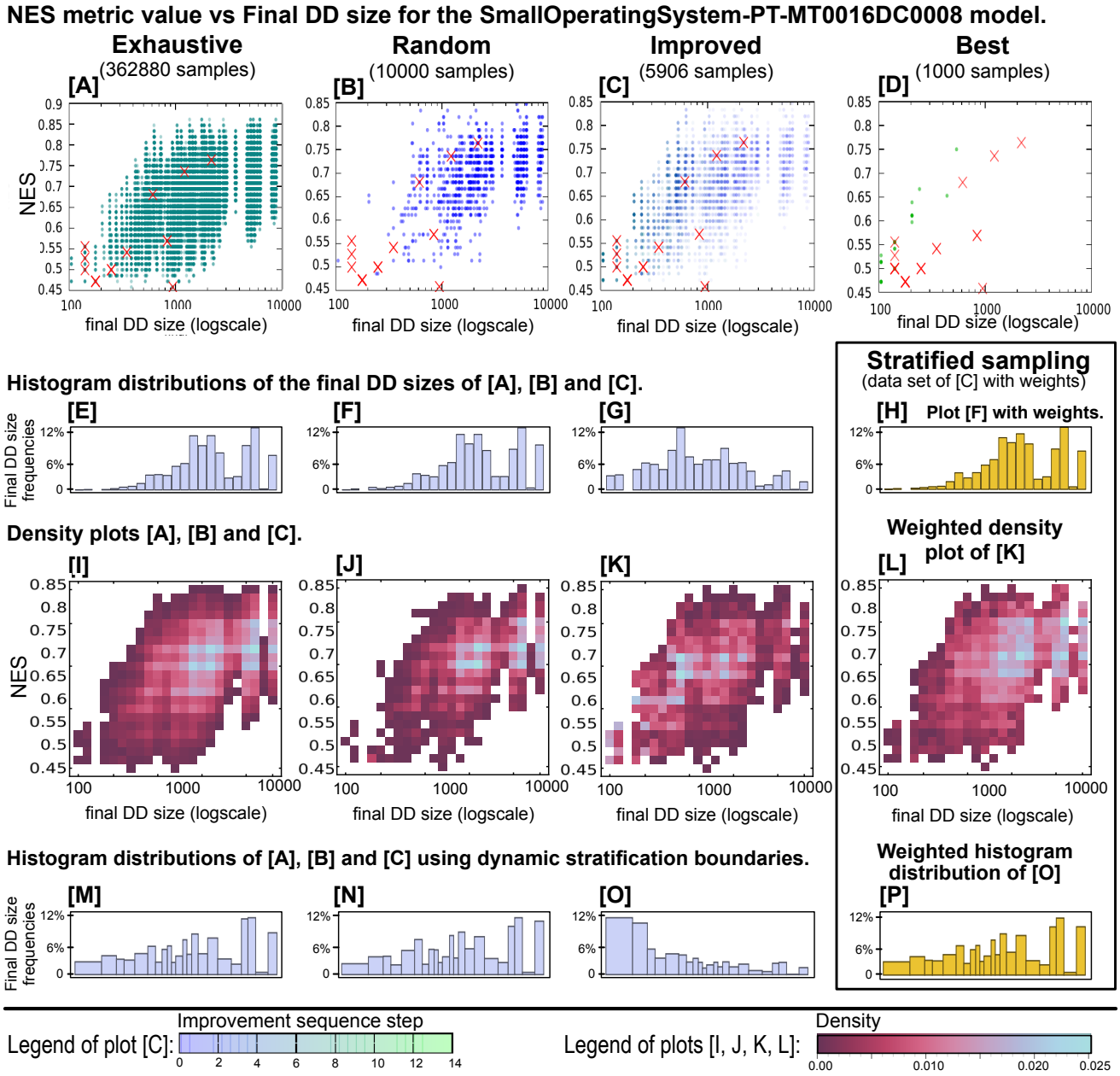


Fig. 3 A complete example of correlation evaluation on an "easy" model.

DD size on the x axis, partitioned in fixed size bins (on the same log-scale), and on the y axis the percentage of variable orders whose size falls in that bin. Plot [E] clearly shows that the distribution is not uniform: "good" variable orders (resulting in a small final DD size) are rare, while most variable orders result in a medium or large final size. Plot [F], built on a random subset of orders, is a close approximation of [E], which is built on the entire set of orders. Plot [G], built on a set of improved orders, is instead skewed towards good orders and, while this reflects the pleasant fact that improved orders are indeed better with respect to the goal of reducing the DD size, it does raise the issue of sample bias.

We stress that, even if the distributions [E] and [F] are similar, this does not imply that the CC computed on $\mathcal{V}_{\text{RND}}$ is a good approximation of the CC on the full set of orders $\mathcal{V}_{\text{EXH}}$.

To summarize, $\mathcal{V}_{\text{RND}}$ is purely random and thus perfectly fair, but suffers from a critical limitation: it is highly likely that the dataset $\mathcal{V}_{\text{RND}}$ has few or even no points in the left side of the plot (where the rare "good" orders lie). On the other hand, the improved dataset $\mathcal{V}_{\text{IMPR}}$ has a higher chance of including data points that span the entire sample space, but the improvement process employed to generate this set alters the sample distribution: the sample density of $\mathcal{V}_{\text{IMPR}}$ (plot [G] in the example) is quite different from that of the

exhaustive set, since the frequency of the "good" variable orders is artificially amplified. Plots [I], [J], and [K] show the density plot of the bivariate series for the NES metric depicted in plots [A], [B], and [C] respectively. The sample space is discretized in a grid of uniform-size squares. Each square in the plot shows the number of samples falling in it, using light blue for high density and dark red for a density close to zero, while white squares contain no samples. All plots indicate that there is a wide range of metric values for which the DDs have a rather large size (above 1000 vs. a best DD size below 100).

Stratified sampling of the improved dataset. We can reduce bias in the $\mathcal{V}_{\text{IMPR}}$ dataset by weighting its elements. To do so, we use *random stratified sampling* [24][29], which partitions the sample space of the DD size (i.e., the x axis) into a set $\mathcal{S}$ of *strata*, and then weights the samples in each stratum using a probability taken from a fair dataset. Let $b : \mathfrak{S}(V) \to \mathcal{S}$ be the function that gives the stratum containing $\mathit{final}(l)$. Let $H_{\mathcal{V}} : \mathcal{S} \to \mathbb{R}$ be the sample probability distribution over $\mathcal{S}$ for set $\mathcal{V}$, defined as $H_{\mathcal{V}}(s) = Pr\{l \in s \mid l \in \mathcal{V}\}$. The distribution $H_{\mathcal{V}_{\text{EXH}}}$ can be computed for small models such as the example of Figure 3 (histogram [E]), but not for larger models. Thus, we use the distribution $H_{\mathcal{V}_{\text{RND}}}$ (histogram [F]) of the random dataset $\mathcal{V}_{\text{RND}}$ to weight the elements of $\mathcal{V}_{\text{IMPR}}$: variable order $l \in \mathcal{V}_{\text{IMPR}}$ receives weight

$$W(l) \;=\; H_{\mathcal{V}_{\text{RND}}}\big(b(l)\big) \;/\; H_{\mathcal{V}_{\text{IMPR}}}\big(b(l)\big). \tag{3}$$

The histogram of the improved set $\mathcal{V}_{\text{IMPR}}$ weighted by W, shown in plot [H], is close to plot [F], but has more points in the "low-probability" regions. Plot [L] shows the density plot of the bivariate series [K] weighted using the frequencies of the random samples (from plot [F]). Stratified sampling reduces the bias observed in plot [K], where the frequencies of the improved samples is amplified.

Dynamic determination of strata boundaries. Using histogram bins of fixed size to compute distributions is a poor approach in practice, since there may be strata with few or no samples, which could lead to some weights in (3) being zero. This degenerate condition can be avoided by defining dynamic strata boundaries. We adopt the *Kozak method* [33], which computes the strata boundaries so that each stratum of $\mathcal{V}_{\text{RND}}$ has a "sufficient" number of samples inside. A detailed description of the method can be found in [33, 7].

Plot [N] shows distribution $H_{\mathcal{V}_{\text{RND}}}$ for the random samples using dynamically sized bins. The stratification boundaries are computed from the $\mathcal{V}_{\text{RND}}$ set. Plots [M,O] show the distributions $H_{\mathcal{V}_{\text{EXH}}}$ and $H_{\mathcal{V}_{\text{IMPR}}}$ using the same strata boundaries of [N]. The weights assigned to the improved samples result in a histogram density [P] similar to the random samples [N], and both are close to the exact frequencies [M], while, as before, the unweighted frequencies [O] are not.

This raises several questions, which we address next: What are the implications of the above observations on the definition and use of the CCs? How should one use the Spearman rank CC computed on $\mathcal{V}_{\text{EXH}}$, $\mathcal{V}_{\text{RND}}$, and $\mathcal{V}_{\text{IMPR}}$, and $\mathcal{V}_{\text{BEST}}$? Under which conditions can one use the bivariate series built on $\mathcal{V}_{\text{IMPR}}$ as representative of $\mathcal{V}_{\text{EXH}}$?

5.2 Definition and evaluation of the correlation coefficients.

Given a metric m, we can define five CCs. Each CC is computed by considering either a different bivariate series $S_m(\mathcal{V}_x)$ or different weights W. We can then define:

- *Exact correlation coefficient (ECC)*: defined on the bivariate series $S_m(\mathcal{V}_{\text{EXH}})$. All samples are treated alike (W is constant).
- *Random correlation coefficient (RCC)*: defined on the bivariate series $S_m(\mathcal{V}_{\text{RND}})$. All samples are treated alike (W is constant).
- *Improved correlation coefficient (ICC)*: defined on the $S_m(\mathcal{V}_{\text{IMPR}})$ bivariate series without weights. All samples are treated alike (W is constant).
- *Stratified correlation coefficient (SCC)*: also defined on $S_m(\mathcal{V}_{\text{IMPR}})$, but samples are weighted using the stratified sampling weights computed through Eq. (3).
- *"Best" correlation coefficient (BCC)*: defined on the bivariate series $S_m(\mathcal{V}_{\text{BEST}})$. All samples are treated alike (W is constant).

For their evaluation, we consider the 12 metrics described in Section 4. Each table reports the number of orders in each dataset (*Orders* row) and, for each metric, the Spearman CC computed for the five settings. The Spearman CCs are computed using the *wCorr* package in R [45], and stratified sampling uses the *stratification* library in R [7] to determine the optimal stratification boundaries with *Kozak method* [33]. We use 25 dynamic bins and an expected coefficient of variation of 0.1 for strata determination. Strata boundaries are estimated from the $\mathcal{V}_{\text{RND}}$ set. Table 1 shows the values of the five CCs for each of the 12 metrics for three "easy" models for which exhaustive exploration of the variable orders is possible. The last row reports the average over all metrics of the relative error with respect to the ECC, again, for each model. The typical interpretation of the Spearman CC is: $[1, 0.8]$ means very strong correlation, $[0.6, 0.8]$ strong correlation, $[0.4, 0.6]$ moderate correlation, $[0.2, 0.4]$ weak correlation, $[0, 0.2]$ very weak to non-existent correlation, while negative values mean anti-correlation. The relative error for a single metric m and a single set of variable orders, for example $\mathcal{V}_{\text{RND}}$, it is computed as $\frac{|RCC_m - ECC_m|}{ECC_m}$. The last row reports the average error relative to ECC: for example, a value of 0.124 for RCC means that, on average, the 12 metrics provide a ranking of the orders in $\mathcal{V}_{\text{RND}}$ whose average CC

Table 1 Comparison between the correlation coefficients on the three "easy" models.

Model:	ResAllocation					SmallOperatingSystem					SwimmingPool				
Correlation coefficient	Exact	Random	Improved	Best	Stratified	Exact	Random	Improved	Best	Stratified	Exact	Random	Improved	Best	Stratified
Orders: / Metric:	40350	10000	5121	1000	5121	362880	10000	5906	1000	5906	362880	10000	7057	1000	7057
PTS_P	0.8616	0.8680	0.9215	0.7754	0.8814	0.7038	0.7271	0.7919	0.8330	0.6802	0.4372	0.4628	0.4407	0.1594	0.4598
SOUPS	0.7368	0.7870	0.8756	0.7348	0.7758	0.5524	0.5735	0.6755	0.6642	0.5469	0.3140	0.3360	0.2620	0.1268	0.3128
PSF	0.7262	0.7380	0.8852	0.7821	0.7476	0.5969	0.6083	0.7791	0.6129	0.5701	0.2410	0.3216	0.5161	0.1029	0.3443
PTS	0.7127	0.7571	0.8486	0.6430	0.7591	0.4500	0.4989	0.6362	0.7832	0.4295	0.3845	0.4073	0.3731	0.2122	0.3905
NES	0.7021	0.7569	0.8510	0.6595	0.7564	0.4511	0.4864	0.6380	0.7514	0.4384	0.3110	0.3282	0.2780	0.1458	0.3081
WES1	0.6402	0.7128	0.7949	0.5906	0.6905	0.4112	0.4327	0.6013	0.7557	0.3824	0.2608	0.2691	0.2356	0.1258	0.2516
PROF	0.6225	0.6308	0.6664	0.3323	0.6501	0.3795	0.4328	0.5680	0.6337	0.3614	0.2236	0.2503	0.1390	-0.063	0.2461
SOT	0.4547	0.5613	0.5672	0.3264	0.5135	0.3036	0.3293	0.4092	0.4845	0.2850	0.2054	0.2356	0.1734	0.0524	0.2147
MAXWF	0.3443	0.3111	0.2589	-0.145	0.3583	0.3739	0.4424	0.6128	0.6673	0.4025	0.1552	0.1721	0.0194	-0.103	0.1469
AVGWF	0.3807	0.3611	0.2730	0.0001	0.4036	0.4598	0.5372	0.7103	0.7593	0.4889	0.1937	0.1985	0.0975	-0.023	0.1781
RMSWF	0.3889	0.3678	0.2730	-0.010	0.4098	0.4632	0.5392	0.7111	0.7668	0.4939	0.2008	0.2057	0.0876	-0.028	0.1835
BW	0.1158	0.1962	0.0353	-0.124	0.1578	0.0794	0.0938	0.2600	0.5259	0.0490	0.0685	0.0824	0.0991	-0.032	0.0831
Average relative error		**0.124**	**0.247**	**0.556**	**0.085**		**0.104**	**0.565**	**0.969**	**0.077**		**0.103**	**0.372**	**0.893**	**0.094**

Table 2 Comparison between the correlation coefficients on three large models.

Model:	MAPK				FMS				RwMutex			
Correlation coefficient	Random	Improved	Best	Stratified	Random	Improved	Best	Stratified	Random	Improved	Best	Stratified
PSF	0.7661	0.8437	0.8672	0.5956	0.8484	0.8109	0.8033	0.5604	0.2509	0.8991	0.3375	0.3263
PTS_P	0.5642	0.8570	0.8208	0.5920	0.6741	0.9093	0.8220	0.7446	0.5447	0.9450	0.5427	0.5747
PROF	0.4852	0.8716	0.8254	0.5744	0.5572	0.8691	0.7100	0.6713	0.5228	0.9472	0.5515	0.5420
SOUPS	0.4514	0.8873	0.7829	0.6754	0.5549	0.9140	0.7981	0.7462	0.5116	0.8532	0.2046	0.5258
RMSWF	0.4339	0.7862	0.7923	0.3366	0.4464	0.7873	0.6738	0.4774	0.4920	0.9062	0.3246	0.5106
AVGWF	0.4247	0.7865	0.8023	0.3364	0.4374	0.7847	0.6797	0.4689	0.4531	0.9119	0.3645	0.4694
NES	0.4008	0.8064	0.7898	0.4840	0.5168	0.8960	0.7837	0.7090	0.5133	0.8510	0.2202	0.5060
PTS	0.3945	0.7974	0.7915	0.4743	0.4578	0.8797	0.7718	0.6826	0.4526	0.8442	0.2809	0.4769
MAXWF	0.3779	0.7027	0.6496	0.2651	0.3945	0.7181	0.5514	0.4276	0.3942	0.7460	0.2375	0.4569
WES1	0.3770	0.7901	0.7780	0.4953	0.5124	0.8838	0.7837	0.7066	0.3971	0.6837	0.0635	0.3878
SOT	0.2612	0.6937	0.6432	0.4646	0.3894	0.7347	0.5451	0.5766	0.3608	0.5692	0.0496	0.3297
BW	0.1080	0.4833	0.6671	0.1264	0.1381	0.4170	0.5313	0.1890	0.0445	0.0730	-0.1169	0.0188
Avg w.r.t. RCC	-	**1.097**	**1.178**	**0.280**	-	**0.772**	**0.619**	**0.276**	-	**0.917**	**0.685**	**0.117**
Avg w.r.t. SCC	**0.247**	**0.940**	**1.013**	-	**0.226**	**0.456**	**0.335**	-	**0.175**	**0.995**	**0.972**	-

differ by 12.4% from the exact CC (built from the ranking based on $\mathscr{V}_{\mathrm{EXH}}$).

Table 2 shows the values of CC (other than ECC) for each of the 12 metrics for three "large" models. These are models with more than 20 variables, for which exhaustive exploration of the variable orders is not possible, thus ECC is unknown, the common situation in practical applications. Since ECC is not available, we compute the average relative error in two ways, relative to RCC (second to the last row) and relative to SCC (last row).

We can now provide a first answer to questions Q1 and Q2. The above analysis settings is what we propose for Q1, while for the predictive power of the metrics (Q2) we can already observe that metrics provide rather diverse correlations, ranging from very weak (e.g. 0.1158 for the metric BW on ECC in Table 1), to very strong (e.g. 0.8616 for the metric PTS^P on ECC in the same table), so their predictive power differs significantly. Moreover, the correlation is not uniform over the considered sets, meaning that some metrics have better prediction performance than others on specific set, which seems to indicate that the choice of a metric should be influenced by the type of application we are considering. This topic is more thoroughly investigated in the next sections. We also observe that the average error relative to ECC (last row of Table 1) and to either RCC or SCC (last two rows of Table 1) indicates that, for the considered examples, RCC and SCC are relatively close to ECC, and to each other.

6 Numerical assessment of metric correlations

The 12 metrics considered focus on different model aspects, thus we can expect them to have different predictive power for different models. For example, for the three models of Table 1, PTS^P is always the best metric for the exhaustive set (highest ECC), but the second best depends on the model. For the three models of Table 2, PTS^P is not always the best metric for the different CCs, but it is always among the best three for any set $\mathscr{V}$ in any of the three models. Clearly, it is

Table 3 The 43 models considered for the evaluation.

Model	Parameters	$\|V\|$	$\|E\|$	$\|\Pi\|$	$\|\mathcal{V}_{\text{IMPR}}\|$	Attempts	$T_{\max}$
AirplaneLD	10	89	88	36	97 030	394 485	99
Angiogenesis	1	39	64	8	64 700	945 577	99
AutoFlight	1a	32	30	11	39 303	317 431	70
BridgeAndVehicles	V4P5N2	28	52	7	47 853	373 191	88
CircadianClock	10	14	16	7	10 741	122 284	21
CircularTrains	12	24	12	42	32 036	283 278	54
CSRepetitions	2	23	28	6	28 558	252 503	53
DatabaseWithMutex	2	38	32	18	52 051	418 277	87
Dekker	10	50	120	40	81 342	606 663	136
Diffusion2D	D5N010	25	144	1	1 000	51 000	1
DotAndBoxes	2	66	164	280	74 474	479 716	129
DrinkVendingMachine	2	24	72	12	19 692	183 762	34
Eratosthenes	50	49	108	15	1 000	51 000	1
ERK	1	11	11	5	8 806	115 742	20
FlexibleBarrier	06a	75	154	8	119 439	827 434	179
FMS	2	22	20	6	22 443	209 058	41
GPPP	C1N1	33	22	19	51 084	398 458	85
GlobalResAllocation	3	33	4 791	75	53 471	425 817	102
HexagonalGrid	110	31	42	15	33 889	294 352	57
HypertorusGrid	d2k1p8b00	13	16	0	32 879	287 425	57
HouseConstruction	2	26	18	7	11 754	126 875	24
JoinFreeModules	3	16	25	4	21 054	207 656	41
Kanban	5	16	16	6	14 095	148 062	27
LamportFastMutEx	2	69	96	17	118 212	761 658	178
MAPK	8	22	30	7	27 969	251 491	63
NeighborGrid	d2n3m1c12	9	40	1	1 000	51 000	1
Parking	104	65	97	17	103 215	680 327	157
Peterson	2	102	126	14	167 482	952 859	221
PhaseVariation	D2CS10	14	65	5	15 072	157 206	32
Philosophers	5	25	25	10	14 899	151 257	72
PhilosophersDyn	3	30	84	12	45 708	363 574	77
Raft	2	28	52	4	34 789	297 549	56
Referendum	10	31	21	10	26 848	236 261	58
ResAllocation	R2C2	8	6	4	5 121	84 261	13
RobotManipulation	5	15	11	9	15 041	162 529	34
RwMutex	r10w10	50	40	30	64 059	461 675	109
SafeBus	3	57	91	265	106 788	739 699	170
SharedMemory	5	41	55	11	63 594	457 191	105
SimpleLoadBal	2	32	45	16	47 233	382 440	81
SmallOperatingSystem	MT16DC8	9	8	4	5 906	85 474	14
SwimmingPool	1	9	7	3	7 057	97 903	15
TokenRing	5	36	156	6	54 603	424 925	96
TriangularGrid	1200	16	12	9	16 132	157 377	38

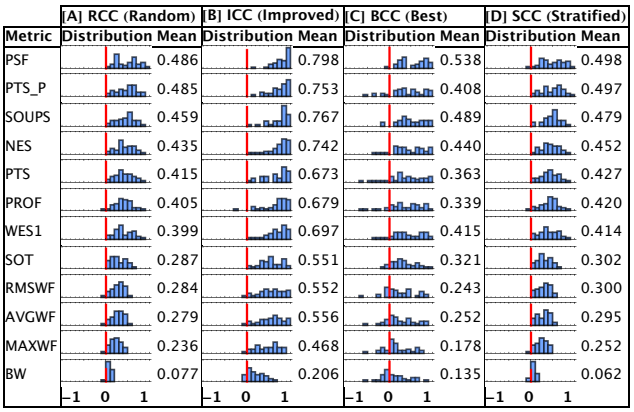


Fig. 4 Correlation coefficient distributions for the tested metrics. Higher is better. Red bar height is fixed to 15 for all rows.

Table 4 Comparison of the average CC vs the metrics.

RCC	metric	**ICC**	metric	**BCC**	metric	**SCC**	metric
0.486	PSF	0.798	PSF	0.538	PSF	0.498	PSF
0.485	PTS_P	0.767	SOUPS	0.489	SOUPS	0.497	PTS_P
0.459	SOUPS	0.753	PTS_P	0.440	NES	0.479	SOUPS
0.435	NES	0.742	NES	0.415	WES1	0.452	NES
0.415	PTS	0.697	WES1	0.408	PTS_P	0.427	PTS
0.405	PROF	0.679	PROF	0.363	PTS	0.420	PROF
0.399	WES1	0.673	PTS	0.339	PROF	0.414	WES1
0.287	SOT	0.556	AVGWF	0.321	SOT	0.302	SOT
0.284	RMSWF	0.552	RMSWF	0.252	AVGWF	0.300	RMSWF
0.279	AVGWF	0.551	SOT	0.243	RMSWF	0.295	AVGWF
0.236	MAXWF	0.468	MAXWF	0.178	MAXWF	0.252	MAXWF
0.077	BW	0.206	BW	0.135	BW	0.062	BW

not enough to compute the correlation between a metric and a single or a few Petri nets. If we want to pick a metric to rank variable orders for an unknown model, it is more useful to study the mean value and/or the distribution and value of the four CCs for each metric over a wide range of models. This section uses the methodology just introduced to evaluate the predictive power of the metrics by computing the distribution and the mean value of the CCs on 43 models, selected so that we can generate 10 000 random orders for each of them. The only modification to the methodology of the previous section is that now ECC cannot be evaluated because the number of variables in almost all models is too large to allow an exhaustive exploration of the variable orders.

Table 3 summarizes the experiments. For each of the 43 models, it lists the model name and parameters, the number of variables $|V|$, events $|E|$, and P-semiflows $|\Pi|$, the size of the improved dataset $|\mathcal{V}_{\text{IMPR}}|$, the number of DDs generated to compute $\mathcal{V}_{\text{IMPR}}$ ("Attempts"), and the maximum length $T_{\max}$ of the improved sequences. Recalling that Algorithm 2 was run with $max_sequence_length = 300$ and $max_attempts = 50$, and that we consider 1 000 initial orders, we can observe that the models have quite different behavior. Models have different size in terms of places and transitions (variables and events) and of their ratio. In no case the improvement algorithm stops due to the limit on the sequence length (300). The parameters used for computing the sets $\mathcal{V}_{\text{RND}}, \mathcal{V}_{\text{IMPR}}, \mathcal{V}_{\text{SEL}}$, and $\mathcal{V}_{\text{BEST}}$ are the same as in the previous section. For this analysis, we generated 14 476 702 DDs (sum of the "Attempts" column) to have 1 859 422 samples (sum of the $\mathcal{V}_{\text{IMPR}}$ column), plus another 430 000 DDs for each $\mathcal{V}_{\text{RND}}$, which is also used for strata boundaries determination. Three models of Table 3 have $|\mathcal{V}_{\text{IMPR}}| = 1\,000$, meaning that none of the attempts were able to produce a smaller DD: a closer inspection reveals that their DD size is insensitive to the variable order. These three models were therefore excluded from the rest of the analysis, since the Spearman correlation coefficient cannot be computed, thus leaving 40 models for the correlation analysis. The analysis was performed by running the state space generation of GreatSPN [3], with parallel independent runs on a 16 cores Xeon E5 machine at 2.4GHz, and it required about 10 days. State space generation is based on the Saturation algorithm implemented in the Meddly library [6].

Figure 4 shows the distribution and the mean value of the CCs for the 40 models. Rows correspond to each metric m and columns report distribution and mean for RCC, ICC, BCC, and SCC. The x axis corresponds to the CC values, from -1 to $+1$ partitioned into 20 bins, while the y axis describes the number of models whose CC falls into each bin.

The scale of the y axis is different for different metrics, but the red bar shown at $x = 0$ always always corresponds to 15, to allow a comparison among rows. For example, the distribution of RCC_{PSF} has significantly fewer than 15 models in each bin, while for RCC_{BW} metric the bin corresponding to the lowest positive correlation is well above 15. Rows are sorted in decreasing order of the average RCC_m. With respect to the predictive power, we can observe that ICC is the best, with most metrics having an average between 0.5 and 0.8 (moderate to strong correlation). This means that the use of the metrics is more likely to be effective when the set of orders observed has a good coverage of all possible orders, with a bias towards better orders, while metrics are less correlating when considering only random orders (RCC column) or only good orders (BCC column). As expected SCC, the stratified CC computed over $\mathscr{V}_{\text{IMPR}}$ is very close to RCC. Table 4 reports instead the average CCs of Figure 4, but with each column individually sorted in decreasing order.

Figure 4 and Table 4 answer Q2 based on a larger set of models than in the previous section. PSF is consistently the best metric (on average, it correlates more than any other metric for any of the four definitions of CC), while the second best is either SOUPS or PTS^P. Five metrics (SOT, RMSWF, AVGWF, MAXWF, and BW) are consistently in the lowest positions, with BW being always the worst. We are not aware of any published work highlighting this lack of correlation for BW, which is used by heuristics like KING and CM, and, partially, by GPS.

The PTS^P and PSF metrics exhibit high correlation with the final DD size: they exploit knowledge of the P-semiflows, which identify sets of connected variables, thus they good correlation is not totally unexpected.

The WES metric, proposed in [40] as an improvement to NES, is actually worse in terms of correlation, for all definitions of CC. Its performance in [40] was likely due to a limited or unfortunate choice of models.

PTS and PROF, used by FORCE and Sloan (very popular algorithms for variable order computation), are not the most correlating metrics, raising the question of whether these algorithms could be improved by using other metrics.

7 Applications

The previous sections investigated the metrics "per se". Now, we investigate instead the possible relationships between metrics and heuristics, to answer question Q3: Can knowledge of each metric correlation help improve the state space construction of DEDS? This first requires a way to "score" the performance of the different heuristics, so that we can investigate the behavior of the heuristics listed in Section 2 and their interplay with the considered metrics. Then, we define and evaluate 1) a modification of the FORCE heuristic, to consider a metric with higher correlation than the PTS used by FORCE, 2) a new heuristic based on the simulated annealing optimization of a given metric, and 3) a new metaheuristic based on metrics. All experiments are performed in the same setting: we use the 40 models from Table 3, and the same $\mathscr{V}$ sets as before.

7.1 A score to evaluate metrics and heuristics.

To study the use of metrics within heuristics or define metaheuristics, we need to be able to compare their behavior based on the DD size generated from a given variable order, where *the smaller the DD, the better the order*. Given a heuristic a producing a variable order l_a for a model $\mathscr{N}$, we can define a *score* for a as

$$\text{SC}_{\mathscr{N}}(a) = 1 - \frac{\log(\mathit{final}(l_a)) - \log(\min_{\mathscr{N}})}{\log(\max_{\mathscr{N}}) - \log(\min_{\mathscr{N}})}, \tag{4}$$

where min and max are the smallest and largest DD sizes built for $\mathscr{N}$ over $\mathscr{V}_{\text{EXH}}$. We choose logarithms in this formula because the DD sizes for the same model but different variable orders may exhibit exponential variations. In particular, this avoids the risk of deeming equally good two variable orders that result in a significantly different number of DD nodes, but such that this difference is instead very small relative to the min − max interval.

Of course, Equation 4 is not applicable in practice since, on most models, $\mathscr{V}_{\text{EXH}}$ is too large. We then use instead the set of Improved Random orders, giving rise to:

$$\text{SC}_{\mathscr{N}}(a) = 1 - \frac{\log(\mathit{final}(l_a)) - \log(\min\{R\})}{\log(\max\{R\}) - \log(\min\{R\})}, \tag{5}$$

with $R = \{\mathit{final}(l') \mid l' \in \mathscr{V}_{\text{IMPR}}\}$. For this analysis $\mathscr{V}_{\text{IMPR}}$ presents the interesting features of covering "good" orders with a higher probability than other sets (including $\mathscr{V}_{\text{RND}}$).

Eq. (5) defines a score "by heuristic and by model" relative to all observed DD sizes for that model. Eq. (5) defines a scale that goes from 1 (best score for the smallest observed DD) to 0 (worst score for the largest observed DD).

We consider the 14 heuristics briefly presented in Section 2, not including FORCE, which is the subject of a more complete evaluation in the next subsection. Each heuristic a has been run on the 40 models of Table 3: for each model $\mathscr{N}$, the heuristic a computes a variable order l_a resulting in a DD of size $\mathit{final}(l_a)$. We then compute the score $\text{SC}_{\mathscr{N}}(a)$ using Eq. (5). Column A in Figure 5 reports, for each heuristic a, the distribution of the $\text{SC}_{\mathscr{N}}(a)$, for each $\mathscr{N}$ in the set of 40 models of Table 3. Each row also reports the mean value of $\text{SC}_{\mathscr{N}}(a)$ over all $\mathscr{N}$. The average over all heuristics is at the bottom of the table, while the last plot shows the distribution and average score of the random orders ($\mathscr{V}_{\text{RND}}$),

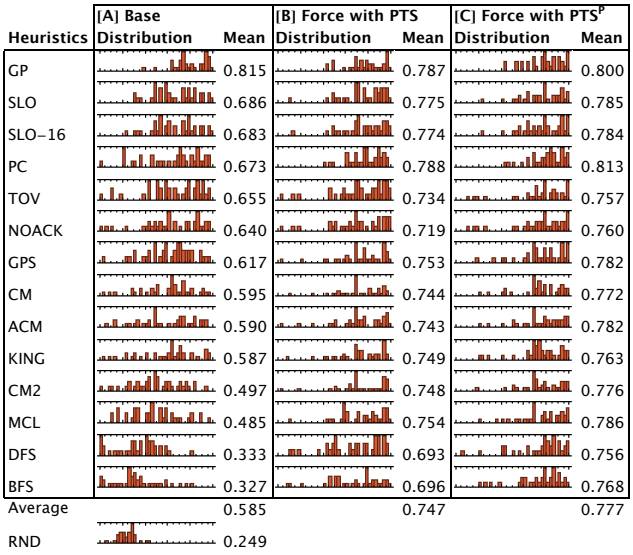


Fig. 5 Score distributions and averages of the tested variable ordering heuristics. Higher is better.

based on $10\,000 \times 40$ samples (unlike the distributions for a heuristic a, which is based on 40 scores, since a computes only one order per model).

Coming back to our question Q3, we can then draw the following conclusions. Among the considered heuristics, GP has the highest average score, confirming the observations in [4]. This is consistent with the fact that GP minimizes the PSF metric, which, from the analysis of Figure 4 and Table 4, has the highest correlation. *Sloan*, a wavefront and profile minimization algorithm, is the second best. Since our correlation analysis suggests that profile (PROF) is a better metric than the wavefront ones (AVGWF, MAXWF, and RMSWF), we conjecture that profile minimization is more important than wavefront minimization, and this could be used to design a new heuristic resulting in better variable orders. The BW reduction heuristics King and CM (all variations) show weaker performance, which is not surprising given that the BW metric has weak correlation. DFS and BFS, not based on any metric and considered only for comparison, are generally poor. Most heuristics are not very stable, in the sense that they sometimes produce exceptionally bad variable orders (bars on the left of the score distribution). We are not aware of any heuristic whose goal is to minimize the NES or SOUPS metric.

7.2 Metrics and FORCE with PTS^P.

We now focus on the popular *FORCE* heuristic for static variable ordering, which takes a variable order l_0 and modifies it iteratively, to minimize the PTS metric. FORCE was not included in the list of heuristics evaluated in the previous subsection, as it is known that its behavior heavily depends on the initial order l_0. The question we would like to answer now is: What is the impact of changing the metric used by FORCE in its iterative procedure? In other words, does a metric with higher correlation than PTS produce better orders? We do so for many different choices of the heuristic used to produce l_0; this allows us to gain insights into which heuristic should be used to produce the initial order for FORCE. We are not aware of other studies about the choice of l_0, and we believe this is an important practical contribution. This is, again, an instance of Question Q3.

Column B of Figure 5 reports analogous results as column A but, for each heuristic a, the orders used to compute the score distribution and the average are now produced running FORCE on an initial order produced by a. The results confirm our experience that FORCE is an effective heuristic: it is, on average over the 40 models, able to improve the initial order l_0 with the single exception of the GP case, a heuristic that already produces an average score of 0.815, quite high considering that overall average score produced by the 14 heuristics is 0.585 and that the average score over the 40 000 random orders is 0.249. The behavior of FORCE does indeed depend on the initial order, but it nevertheless generally produces reasonably good variable orders, especially because it is able to significantly improve poor performing orders (lower rows of the table).

Column C of Figure 5 reports instead the results of FORCE modified to use PTS^P instead of PTS, a more correlating metric according to the experiments of Figure 4. On all sets for which we computed the CC, PTS^P was always slightly better than PTS, and indeed the average score computed using FORCE with PTS^P is indeed slightly better than PTS. Moreover, not only the average over all models and all initial variable orders is better, but also the average of the scores of each single initial order (the average values in column C are always higher than than in column B). Note that it we cannot run FORCE with a metric significantly more correlating than PTS, since, among the metrics we consider, FORCE, by its nature, can only use PTS or PTS^P.

7.3 Heuristics based on metrics and simulated annealing

We now experiment how a metric m could be used inside the definition of a new heuristic that search an "optimal" order through a simulated annealing procedure [19], aimed at minimizing ther value of m, resulting in a heuristic we call m-annealing. Starting from an initial variable order for a given model $\mathcal{N}$, m-annealing evaluates possible alternative orders obtained by exchanging a pair of variables randomly selected and accepts or rejects the new order l with a decision procedure based the value of $m(l)$. Acceptance follows the rules of Boltzmann annealing with a temperature which is equal to the metric value, decreased at each step of

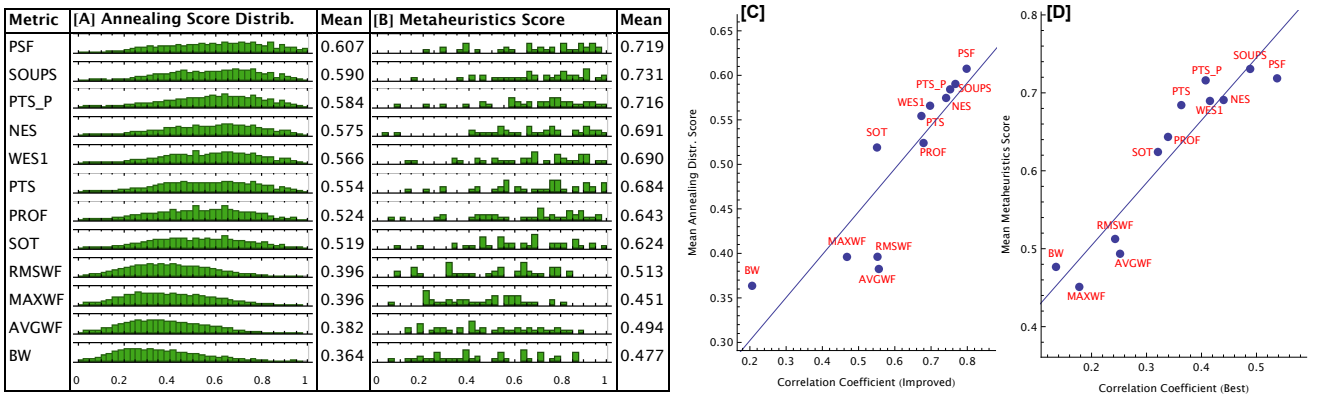


Fig. 6 Score distribution using a metric-guided simulated annealing.

$1/10000$ of its initial value, and a probability of acceptance which follows the algorithm in [19], pag.30-31.

To limit the possible bias generated by the choice of the initial order, we consider the usual set of 1 000 orders for $\mathcal{N}$, and, for each order, m-annealing produces a new order.

This procedure resembles the one used to build the $\mathcal{V}_{\text{IMPR}}$ set, but with the major difference that the selection of an order l is based on knowledge of $final(l)$, which requires building the DD, in the case of $\mathcal{V}_{\text{IMPR}}$, while m-annealing is based only on $m(l)$, which simply requires a metric evaluation.

We evaluate 12 m-annealing heuristics, one for each metric m. Figure 6[A] reports the distribution and the mean of the score for our set of 40 models, one row per heuristic. Each distribution is built from the scores of $40 \times 1\,000$ DDs. The relative value of the m-annealing heuristics is quite clear from the distributions, indicating that correlation has an impact: the m-annealing heuristics built out of the most correlating metrics, like PSF, SOUPS, and PTS^P, exhibit the highest scores. However, none of the m-annealing heuristics is very effective; for example, Figure 5[C] shows that FORCE, with heuristic PC for the generation of l_0, produces a better average score than the best one in Figure 6[A] (0.813 for FORCE vs. 0.607 for PFS-annealing).

The above analysis investigates the influence of metric correlation on the heuristic score. A practical application of m-annealing requires also a way to select one order among the 1 000 ones produced for a given model, that is to say it is necessary to define a meta-heuristic that drives the selection of the single order to use for the DD construction. The evaluation of the use of metrics in meta-heuristic is indeed the next problem we tackle, but we anticipate here the case in which the best order is deemed to be the one with the lowest m value among the 1 000 produced by m-annealing. Figure 6[B] presents the results: the distribution in each row m is now over the 40 samples, one per model, selected according to the lowest value of m. If multiple orders tie for the the same best metric value, we take the average score. A comparison of the mean values reported in the two columns indicates, as expected, that a metric-based meta-heuristic tends to select better orders. It is also remarkable that the best two, SOUPS and PSF, invert their position in the rank: PSF produces better orders than SOUPS, but SOUPS is more effective than PSF in selecting the best ones.

But what is the actual impact of the CCs? If we were to choose a-priori one of the 12 m-heuristics, is selecting the one with the highest CC our best bet? And when the selected one happens not to be the best for a particular model, how bad can it be? Remember that the CCs were computed w.r.t. the final DD sizes, while scores are meant to provide a "relative" evaluation of the heuristics, so the relationship between the two is not obvious. The plots on Figure 6 allow us to visualize this answer. Figure 6[C] plots the average score of each m-annealing heuristic (on the y-axis) against ICC_m (on the y-axis); Figure 6[D] plots it instead against BCC_m; the values for ICC_m and BCC_m are those reported in Figure 4. We choose ICC when the average score is computed over all the orders produced by the simulated annealing procedure (values of column Mean in Figure 6[A]) and BCC when the average score is that of the meta-heuristic (Figure 6[B]). This is a reasonable choice, since the meta-heuristic performs a selection, as does the construction of $\mathcal{V}_{\text{BEST}}$, which selects the final elements of the traces encountered during the construction of $\mathcal{V}_{\text{IMPR}}$. For these plots, again, metric m "is better" than metric m' if $\text{SC}_m > \text{SC}_{m'}$. Then, the question is whether higher scores correspond to higher CC, which can be translated on the plots as: is it true that $y_m > y_{m'} \Rightarrow x_m > x_{m'}$? For plot [C] this is true for the 5 highest scores, while PTS has a higher score than PROF, but ICC_{PROF} is greater than ICC_{PTS}. The other peculiarity is that SOT, AVGWF, and RMSWF have about the same ICC, but SOT has a much higher score.

With respect to question Q3 (Can knowledge of each metric correlation help improve the state space construction of DEDS?), we can conclude that, while the m-annealing heuristics (each complemented by the m meta-heuristic) are no better than the best instance of FORCE(see Figure 5, knowl-

edge of the correlation does make a difference, as a more correlating metric m produces a more effective m-heuristic. In particular, if we choose one of the best 5 metrics according to ICC, the difference in average score of the m-heuristics is less than 10%, while if we choose one of the best 5 metrics on BCC, the difference in average score of the m-meta-heuristics is less than 5%.

7.4 Use of metrics for meta-heuristics

We now focus on the question of whether more correlating metrics are better for the definition of a meta-heuristic. This is an instance of question Q3, and a positive answer would have significant practical implications, as this is the typical situation a user has to face: how to select a single variable order among a set of few (in the case of $\mathcal{V}_{\text{SEL}}$) or many (in the case of $\mathcal{V}_{\text{RND}}$) possible candidates. For this evaluation, we define 12 meta-heuristics, one for each metric m: in each meta-heuristic, metric m is used to select the best order to use for a given model, among a set of generated orders. Again, we consider four sets of variable orders: $\mathcal{V}_{\text{RND}}$, $\mathcal{V}_{\text{IMPR}}$, $\mathcal{V}_{\text{BEST}}$, and $\mathcal{V}_{\text{SEL}}$; from each, we select the best order according to the meta-heuristic, build the DD, and compute a score. In other words, we consider the use of a m-meta-heuristic over the different sets we introduced. For each metric m, Figure 7[A] reports the distribution and the average of the score for the DD of 40 models, where each DD is built using the order selected by the $m_{\mathcal{V}_{\text{RND}}}$-meta-heuristic. For example, the distribution reported for SOUPS in column [A] is built using, for each of the 40 models, the order with the best SOUPS value in $\mathcal{V}_{\text{RND}}$. The last row reports instead the distribution and the mean of the score over all orders in the corresponding set and all 40 models. Thus, while the distributions on the first 12 rows are built out of 40 scores, one per model, the distributions of the Baseline row are built over all the scores in the corresponding set. These distributions are our baseline for comparison, since in practical applications and without a meta-heuristic, we would have taken one order out of that set. For column [B], the distribution on the last row is built out of the orders in $\mathcal{V}_{\text{IMPR}}$ (a total of 1 859 422 samples for the 40 models); for column [C], out of of the orders in $\mathcal{V}_{\text{BEST}}$ ($40\times1\,000$ samples); and for column [D], out of the orders in $\mathcal{V}_{\text{SEL}}$ ($14\times3\times40$ samples). As expected following the observations in Section 5.1, the scores of the $\mathcal{V}_{\text{RND}}$ orders are bad (the distribution is concentrated on the left), while the distribution of $\mathcal{V}_{\text{IMPR}}$ tends to have both bad and good orders and the the one for $\mathcal{V}_{\text{BEST}}$ contains quasi-optimal variable orders. The baseline of column [D] is built out of the $14\times3\times40$ orders produced by 14 heuristics plus 28 versions of FORCE with PTS or PTS^P, and it does not show a clear distribution, possibly because of the low number of samples but also because the heuristics, at least in our experience, exhibit a very variable behavior.

The m-meta-heuristic is better than the baseline in all cases except for the $BW_{\mathcal{V}_{\text{SEL}}}$-meta-heuristic (0.687 vs. 0.703). Most importantly, the best meta-heuristics significantly improve the score over the baseline.

The plots in Figure 7[E,F,G,H] allow us to examine reason the usefulness of the indications provided by the various CCs. Each plot refers to the relationship between the value of the CC for a metric m and the score of the meta-heuristic based on m on the $\mathcal{V}_{CC}$ Each point in the plots corresponds to a meta-heuristic, where its y value is the score and its x axis is the value of CC. Figures [E] and [F] plot the Mean of column [A] against RCC and SCC. As already observed, SCC and RCC have a similar behavior and for both of them the highest scores are obtained for metrics with higher correlation, while the lowest four correlating metrics produce the lowest scores. Figures [G] and [H] plot the Mean of column [B] and [C] against ICC and SCC, respectively, as done for the simulated annealing study. We observe again that the most correlating metric produces the best score, although the order is not strictly preserved, for example plot [H] shows the same inversion between PSF and SOUPS that was present in the plot of Figure 6[D].

With respect to question **Q3**, we now investigate whether the ranking of a metric according to its CC computed over a set of orders is effective to select the best order among that set. The analysis of Figure 7 involves sets of order that may not be significant or available in practice: $\mathcal{V}_{\text{RND}}$ is easy to generate, but, as we have seen, its coverage of good orders is questionable; $\mathcal{V}_{\text{IMPR}}$ and $\mathcal{V}_{\text{BEST}}$ are typically not available when employing a meta-heuristic (their definition requires building the DDs for a very large number of orders); $\mathcal{V}_{\text{SEL}}$ is relatively easy to compute but includes a small number of variable orders as input to the meta-heuristics.

We thus propose to evaluate the 14 meta-heuristics on a different set of orders, that can be built in practice. The requirement is to have more orders than $\mathcal{V}_{\text{SEL}}$ and a better coverage of good orders than $\mathcal{V}_{\text{RND}}$. This approach also allows us to investigate the behavior of the metrics in a context different from the one where the CCs were computed (although still on the same set of models). We call this set $\mathcal{V}_{\text{FORCE}}$, generated as follows: we start from the same 1 000 initial orders considered for the construction of $\mathcal{V}_{\text{IMPR}}$, and apply on each of them the standard FORCE, thus producing 1 000 (possibly) different orders.

Figure 8 (left) reports the results with the same format as in Figure 7. These results can be compared with those where we use FORCE directly (Figure 5) and with those for $\mathcal{V}_{\text{RND}}$ and $\mathcal{V}_{\text{SEL}}$ in Figure 7. The average score of the best m-meta-heuristic (the one based on SOUPS) is 0.842, higher than 0.813, the best score of the heuristics reported in Figure 5 (FORCE with PTS^P with an initial order computed using the PC heuristic). While the number of generated orders is higher for Figure 8 than for Figure 5, this additional cost is

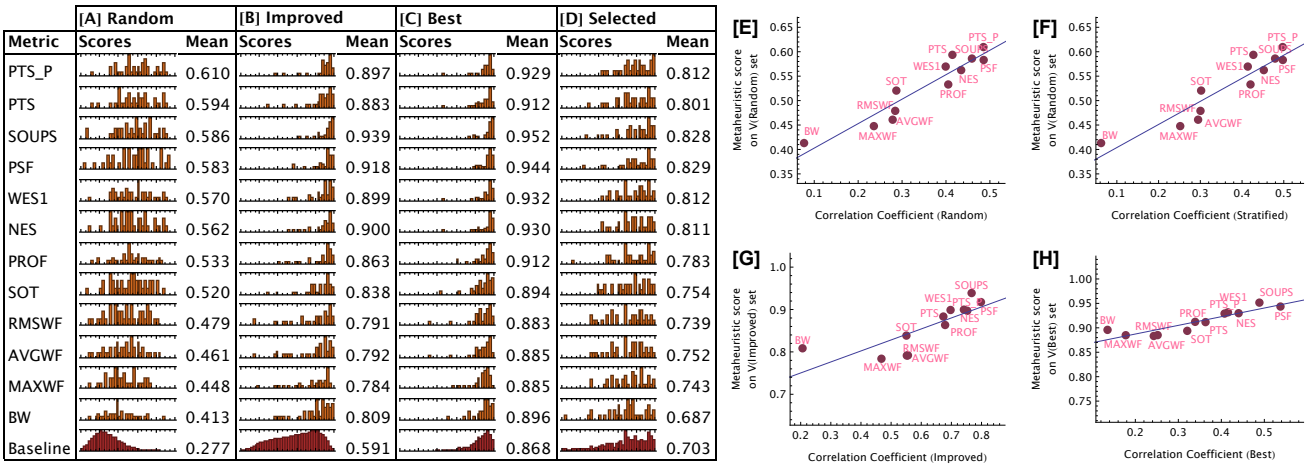


Fig. 7 Score distribution using a metric-based meta-heuristics.

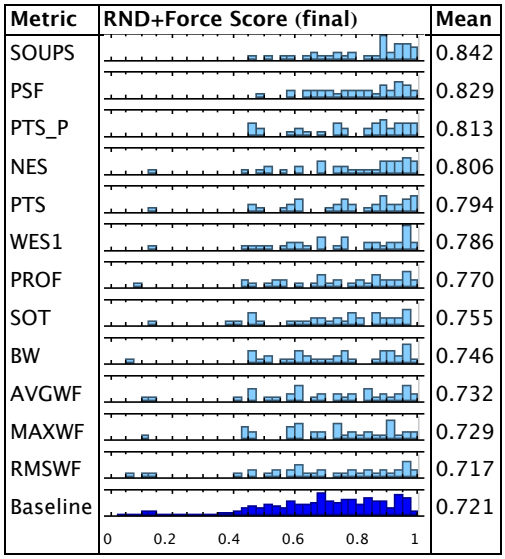


Fig. 8 Score distribution using a metric-guided selection from orderings obtained using FORCE.

small compared to the savings due to even a small improvement in the size of the DD being built for a model.

8 Conclusions

In this paper we have studied the predictive power of variable order metrics and we have experimented how this knowledge can be used for producing new heuristics for good variable orders.

We have defined a methodology to compute several correlation coefficients for decision diagrams' metrics. By combining a technique for variable order improvement (to avoid using only "poor" random orders) with stratified sampling (to reduce the sample bias), we obtained different sets of orders (from purely random to a set of the best observed orders) on which we have computed various coefficient of correlations to evaluate the predictive power of the metrics in different contexts. We considered ten existing metrics, chosen since they directly or indirectly play a significant role in some popular variable order heuristic algorithms, plus two newly defined one (PSF and PTS^P). We then applied this methodology to study the predictive power of the 12 metrics on the variable orders of a representative set of 40 Petri net models. The 40 are a subset of the 77 used in the 2017 Model Checking Contest at the annual Petri net conference. We have chosen the models (and the associated parameters), for which it was possible to generate the sets of orders within the given resource limits. The computation of the CC for the 12 metrics on the 40 models required the construction of 14 476 702 DDs variable orders and the construction of the associated DDs: indeed a good coverage of the order space (that includes very good orders, if any, and very bad ones, if any) is a requirement to compute significant indices. Our experiments show that the correlation among the final DD size and the value of the tested metrics may significantly differ from metric to metric, as there are metrics like BW with "very weak to non-existing" correlation, up to metrics like SOUPS and PSF that exhibit a "strong" correlation. Moreover the metrics with the highest CC on one set of orders may not be the best one on another set of orders, nevertheless if we consider the three best metrics on the four types of CCs considered (CC on $\mathscr{V}_{\text{RND}}, \mathscr{V}_{\text{IMPR}}, \mathscr{V}_{\text{BEST}}$, and stratified CC on $\mathscr{V}_{\text{IMPR}}$) we observe that 11 of the 12 best positions are occupied by three metrics (SOUPS, PSF, and PTS^P).

We have provided evidence of the positive impact of highly correlating metrics on variable order heuristics by defining and evaluating two new heuristics and a meta-heuristic. To do so we have defined a notion of "score" to be able to compare the DDs produced on the whole set of 40 models and associated thousands of orders. The first heuristic is a modification of FORCE in which the standard PTS metric

that FORCE optimizes is substituted by an higher correlating metric (PTS^P). The experiments show that the use of a more correlating metric produces better orders with better scores and even if the improvement is limited, FORCE is not only more effective but also more stable when it minimizes PTS^P instead of PTS.

The second heuristic is a metric-based simulated annealing in which a metric m is used to drive a simulating annealing optimization procedure, actually 1000 procedures started from 1000 different initial orders for each one of the 40 models. The same metric m is then used to select the best order for each model. The resulting scores are lower than the ones obtained with the FORCE based heuristic, but they clearly show that the annealing procedures that use the best correlating metrics produce the best set of orders.

Finally we have defined a meta-heuristic in which a metric m is used to drive the selection of the "best" variable order among a set of variable orders produced with different techniques: the set of orders that are defined in the methodology for the CC evaluation, plus a new set of "improved" orders built using FORCE from an initial set of $1,000$ random orders. Again, we observe that when an highly correlating metric is used as the basis for the meta-heuristics, we get better scores (in particular the three best average scores are obtained, again, for SOUPS, PSF, and PTS^P). Since the construction of improved set of orders based on FORCE does not require the construction of any DD (unlike the construction of $\mathcal{V}_{\text{IMPR}}$), the meta-heuristic based on a combination of random order improved with FORCE and selected using an highly correlating metric is a meta-heuristic that *can be applied in practice* for the construction of the DD of the state space of large models.

We stress that, while all models used for our experiments are Petri nets from the Petri net Model Checking Contest, our methodology to evaluate correlation is fully general. Moreover most of the 12 metrics and the defined heuristics can be applied to many other discrete-state formalisms, as long as their behavior is captured by a set of asynchronous events and can be analyzed using DDs. This is true even for the metrics PSF and PTS^P, which rely on knowledge of the P-semiflows, a notion defined specifically for Petri nets, but which can be easily extended to other formalisms, since a P-semiflow simply identifies a set of variables that must satisfy a linear constraint.

The study of the CC of various metrics paves the way to different lines of future work. A first research question is whether metrics can be used to drive dynamic reordering heuristics. While it is true that dynamic reordering works at the library level, where typically the information on the structure of the model is not directly available, nevertheless it would be interesting to see whether an extension to include such knowledge and the metrics computation can lead to more effective dynamic reordering procedures.

A second research question is to improve the current set of metrics: indeed none of the metrics reaches a very high correlation (what is usually defined as "very strong" correlation). On the other side there are always cases (although very few) in which metrics with very low correlation are able to select good orders. The challenge is then whether it is possible to define a metric that can take into account the positive characteristics of different metrics to boost the behavior of the constituent metrics, while avoiding the risk of resulting in an average, unsatisfactory, behavior.

Appendix: Weighted Spearman correlation

We summarize the formulas used to compute the weighted Spearman correlation coefficient from a weighted bivariate series (X,Y,W) used by the "wCorr" package [45]. A more detailed explanation can be found in [46]. The Pearson correlation coefficient $\sigma_P(X,Y)$ is defined as $\frac{\sigma_{X,Y}}{\sigma_X \sigma_Y}$ where $\sigma_{X,Y}$ is the covariance between X and Y, and σ_X and σ_Y are the standard deviations of X and Y, respectively. The weighted Pearson correlation coefficient is defined as:

$$\sigma_P(X,Y,W) = \frac{\sum_{i=1}^{n} [w_i(x_i-\bar{x})(y_i-\bar{y})]}{\sqrt{\sum_{i=1}^{n} (w_i(x_i-\bar{x})^2) \sum_{i=1}^{n} (w_i(y_i-\bar{y})^2)}}, \quad (6)$$

where $\bar{x}$ and $\bar{y}$ are the weighted means of X and Y:

$$\bar{x} = \frac{1}{\sum_{i=1}^{n} w_i} \sum_{i=1}^{n} w_i x_i, \qquad \bar{y} = \frac{1}{\sum_{i=1}^{n} w_i} \sum_{i=1}^{n} w_i y_i,$$

using w_i as the weights. The Spearman coefficient $\sigma_S(X,Y)$ is defined as the Pearson coefficient over the *ranks* X' and Y' of X and Y, i.e., the values of X and Y are replaced with their relative position. The *weighted* Spearman coefficient $\sigma_S(X,Y,W)$ is computed as the weighted Pearson coefficient $\sigma_P(X',Y',W)$ where X' and Y' are the *weighted ranks* X' and Y' of X and Y, defined so that the j-th element of X has rank

$$rank_j = \sum_{\substack{x_k \in X' \\ x_k < x_j}} w_k \; + \; \frac{n_j+1}{2} \bar{w}_j S,$$

where n_j is the number of entries in X having value x_j, and $\bar{w}_j$ is the average weight of those entries. The weighted ranks Y' of Y are defined analogously. Note that $\sigma_S(X,Y,W)$ equals $\sigma_S(X,Y)$ when all weights W are equal.

References

1. Aloul, F.A., Markov, I.L., Sakallah, K.A.: FORCE: A fast and easy-to-implement variable-ordering heuristic. In: Proc. of GLSVLSI, pp. 116–119. ACM, NY (2003)
2. Amparore, E.G.: A New GreatSPN GUI for GSPN Editing and CSL^{TA} Model Checking. In: QEST, pp. 170–173. Springer (2014)

3. Amparore, E.G., Balbo, G., Beccuti, M., Donatelli, S., Franceschinis, G.: 30 Years of GreatSPN, chap. In: Principles of Performance and Reliability Modeling and Evaluation: Essays in Honor of Kishor Trivedi, pp. 227–254. Springer, Cham (2016)
4. Amparore, E.G., Beccuti, M., Donatelli, S.: Gradient-based variable ordering of decision diagrams for systems with structural units. In: Automated Technology for Verification and Analysis, pp. 184–200. Springer (2017)
5. Amparore, E.G., Donatelli, S., Beccuti, M., Miner, A.: Decision diagrams for Petri nets: which variable ordering? (2018). Accepted for pubblication on Transactions on Petri Nets and Other Models of Concurrency, Springer
6. Babar, J., Miner, A.: Meddly: Multi-terminal and edge-valued decision diagram library. In: Quantitative Evaluation of Systems, International Conference on, pp. 195–196. IEEE Computer Society, Los Alamitos, CA, USA (2010)
7. Baillargeon, S., Rivest, L.P.: The construction of stratified designs in R with the package stratification. Survey Methodology **37**(1), 53–65 (2011)
8. Berthomieu, B., RIBET, P.O., Vernadat, F.: The tool tina ? construction of abstract state spaces for petri nets and time petri nets (2004)
9. Bollig, B., Löbbing, M., Wegener, I.: On the effect of local changes in the variable ordering of ordered decision diagrams. Information Processing Letters **59**(5), 233 – 239 (1996)
10. Bollig, B., Wegener, I.: Improving the variable ordering of OBDDs is NP-complete. IEEE Trans. Comput. **45**(9), 993–1002 (1996)
11. The Boost-C++ library. `http://www.boost.org/`
12. Bryant, R.E.: Graph-based algorithms for boolean function manipulation. IEEE Transactions on Computers **35**, 677–691 (1986)
13. Ciardo, G., Jones, R.L., Miner, A.S., Siminiceanu, R.: Logical and stochastic modeling with SMART. Perf. Eval. **63**, 578–608 (2006)
14. Ciardo, G., Lüttgen, G., Siminiceanu, R.: Saturation: An efficient iteration strategy for symbolic state-space generation. In: TACAS'01, pp. 328–342 (2001)
15. Cimatti, A., Clarke, E.M., Giunchiglia, F., Roveri, M.: NUSMV: A new symbolic model verifier. In: 11th Int. Conf. on Computer Aided Verification, pp. 495–499. Springer (1999)
16. Cuthill, E., McKee, J.: Reducing the bandwidth of sparse symmetric matrices. In: Proc. of the 1969 24th National Conference, pp. 157–172. ACM, New York (1969)
17. van Dijk, T., Hahn, E.M., Jansen, D.N., Li, Y., Neele, T., Stoelinga, M., Turrini, A., Zhang, L.: A comparative study of bdd packages for probabilistic symbolic model checking. In: X. Li, Z. Liu, W. Yi (eds.) Dependable Software Engineering: Theories, Tools, and Applications, pp. 35–51. Springer International Publishing, Cham (2015)
18. van Dijk, T., van de Pol, J.: Sylvan: multi-core framework for decision diagrams. International Journal on Software Tools for Technology Transfer **19**(6), 675–696 (2017)
19. Du, K.L., Swamy, M.N.S.: Search and Optimization by Metaheuristics. Basel, Springer (2016)
20. Fujita, M., Matsunaga, Y., Kakuda, T.: On variable ordering of binary decision diagrams for the application of multi-level logic synthesis. In: Proceedings of the conference on European design automation, EURO-DAC'91, Amsterdam, The Netherlands, 1991, pp. 50–54 (1991)
21. Garavel, H., Lang, F., Mateescu, R., Serwe, W.: Cadp 2011: a toolbox for the construction and analysis of distributed processes. International Journal on Software Tools for Technology Transfer **15**(2), 89–107 (2013)
22. Gibbs, N.E., Poole Jr, W.G., Stockmeyer, P.K.: An algorithm for reducing the bandwidth and profile of a sparse matrix. SIAM Journal on Numerical Analysis **13**(2), 236–250 (1976)
23. Heiner, M., Rohr, C., Schwarick, M., Tovchigrechko, A.A.: MARCIE's secrets of efficient model checking. In: Transactions on Petri Nets and Other Models of Concurrency XI, pp. 286–296. Springer, Heidelberg (2016)
24. Hocevar, D.E., Lightner, M.R., Trick, T.N.: A study of variance reduction techniques for estimating circuit yields. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems **2**(3), 180–192 (1983)
25. Hollander, M., Wolfe, D.A.: Nonparametric statistical methods. Wiley-Interscience (1999)
26. Jorn Lind-Nielsen: BuDDy Manual. `http://http://buddy.sourceforge.net/manual/main.html` (2003)
27. Kam, T., Villa, T., Brayton, R.K., Sangiovanni-Vincentelli, A.: Multi-valued decision diagrams: theory and applications. Multiple-Valued Logic **4**(1), 9–62 (1992)
28. Kamp, E.: Bandwidth, profile and wavefront reduction for static variable ordering in symbolic model checking. Tech. rep., University of Twente (June, 2015)
29. Keramat, M., Kielbasa, R.: A study of stratified sampling in variance reduction techniques for parametric yield estimation. IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing **45**(5), 575–583 (1998)
30. King, I.P.: An automatic reordering scheme for simultaneous equations derived from network systems. Journal of Numerical Methods in Eng. **2**(4), 523–533 (1970)
31. Kordon, F., Garavel, H., Hillah, L.M., Hulin-Hubard, F., Berthomieu, B., Ciardo, G., Colange, M., Dal Zilio, S., Amparore, E., Beccuti, M., Liebke, T., Meijer, J., Miner, A., Rohr, C., Srba, J., Thierry-Mieg, Y., van de Pol, J., Wolf, K.: Complete Results for the 2017 Edition of the Model Checking Contest. `http://mcc.lip6.fr/2017/results.php` (2017)
32. Kordon, F., Garavel, H., Hillah, L.M., Paviot-Adet, E., Jezequel, L., Hulin-Hubard, F., Amparore, E., Beccuti, M., Berthomieu, B., Evrard, H., Jensen, P.G., Botlan, D.L., Liebke, T., Meijer, J., Srba, J., Thierry-Mieg, Y., van de Pol, J., Wolf, K.: MCC2017 - The Seventh Model Checking Contest. Accepted for publication at TopNoC, Springer (2017)
33. Kozak, M.: Optimal stratification using random search method in agricultural surveys. Statistics in Transition **6**(5), 797–806 (2004)
34. Kwiatkowska, M., Norman, G., Parker, D.: PRISM: Probabilistic Model Checking for Performance and Reliability Analysis. Performance Evaluation **36**(4), 40–45 (2009)
35. McMillan, K.L.: Symbolic Model Checking. Kluwer Academic Publishers, Norwell, MA, USA (1993)
36. Meijer, J., van de Pol, J.: Bandwidth and wavefront reduction for static variable ordering in symbolic reachability analysis. In: NASA Formal Methods, 2016, pp. 255–271. Springer, Cham (2016)
37. Murata, T.: Petri nets: Properties, analysis and applications. Proceedings of the IEEE **77**(4), 541–580 (1989)
38. Rudell, R.: Dynamic variable ordering for ordered binary decision diagrams. In: Proceedings of the 1993 IEEE/ACM International Conference on Computer-aided Design, ICCAD '93, pp. 42–47. IEEE Computer Society Press, Los Alamitos, CA, USA (1993). URL `http://dl.acm.org/citation.cfm?id=259794.259802`
39. Schwarick, M., Heiner, M., Rohr, C.: Marcie - model checking and reachability analysis done efficiently. In: Quantitative Evaluation of Systems (QEST), 2011 Eighth International Conference on, pp. 91–100 (2011)
40. Siminiceanu, R.I., Ciardo, G.: New metrics for static variable ordering in decision diagrams. In: 12th Int. Conf. TACAS 2006, pp. 90–104. Springer, Heidelberg (2006)
41. Sloan, S.W.: An algorithm for profile and wavefront reduction of sparse matrices. International Journal for Numerical Methods in Engineering **23**(2), 239–251 (1986)
42. Smith, B., Ciardo, G.: SOUPS: A variable ordering metric for the saturation algorithm. In: Proceedings of the 2018 International Conference on Application of Concurrency to System Design (2018). To appear

43. Somenzi, F.: Efficient manipulation of decision diagrams. STTT **3**(2), 171–181 (2001)
44. Thierry-Mieg, Y.: Symbolic model-checking using its-tools. In: TACAS, *Lecture Notes in Computer Science*, vol. 9035, pp. 231–237. Springer (2015)
45. The wCorr library by Ahmad Emad and Paul Bailey. `https://cran.r-project.org/web/packages/wCorr/wCorr.pdf`
46. The wCorr formulas. `https://cran.r-project.org/web/packages/wCorr/vignettes/wCorrFormulas.html`