Proceedings of the 2021 Winter Simulation Conference
S. Kim, B. Feng, K. Smith, S. Masoud, Z. Zheng, C. Szabo, and M. Loper, eds.

SPSC: AN EFFICIENT, GENERAL-PURPOSE EXECUTION POLICY FOR STOCHASTIC SIMULATIONS

Yu-Lin Huang
Gildas Morvan
Frédéric Pichon
David Mercier

Université d'Artois, UR 3926
Laboratoire de Génie Informatique et d'Automatique de l'Artois
F-62400 Béthune, FRANCE

ABSTRACT

In this paper, we present a stochastic simulation execution policy named SPSC from its different steps: Simulation, Partitioning, Selection, Cloning. It does not require prior knowledge on the model to be applied and allows one to estimate efficiently the probabilities of possible simulation outcomes, so called solutions. It is evaluated on three different multi-agent-based simulations. Results show that SPSC could be a relevant alternative to the Monte Carlo method.

1 INTRODUCTION

Stochastic simulations are widely used in application domains where no deterministic laws can be used to predict the future state of the system. The results of such simulations being variable by nature, it is mandatory to replicate their execution in order to retrieve experimentally the distribution of possible simulation outcomes, so called solutions and then obtain their probabilities. In the general case, the Monte-Carlo (MC) method is classically used (Rubinstein and Kroese 2016). Nevertheless, if computing resources are limited or solutions of interest are rare, MC can be inefficient (Rubino et al. 2009).

In this paper, we present a simulation execution policy called SPSC, based on the clustering, selection and cloning of replication states, in order to improve the quality of the results for a fixed computing cost while being as generic as possible. The general idea of this policy is to constrain the evolution of the replications by periodically selecting and cloning those potentially leading to interesting results for the decision maker. These ideas were shortly introduced in Huang et al. (2019). Here, we extend that original presentation, with refined analyses and experiments.

After formalizing the problem, we present some existing solutions in Section 2. Our approach, as well as its implementation are detailed in Section 3. Finally, the policy is validated on two academic agent-based models (prey-predator and virus transmission), then on a more complex traffic simulation. Experiments and results are discussed in Section 4. Finally, we conclude by presenting the main perspectives of this work.

2 PREAMBLE

2.1 Problem formalization

The state of a stochastic simulation is fully characterized by the set of its state variables. Nevertheless, it can be very large. Thus, to be interpreted by a decision maker, this raw state is generally reduced to a set of meaningful *observables* that sums it up, *e.g.*, for a population of individuals, the observables could be its size, the mean age of the individuals, the sex-ratio, etc.

978-1-6654-3311-2/21/$31.00 ©2021 European Union

Observables at time step t can be represented by a random vector X_t. We denote $\mathscr{D}$ the domain of vector X_t. Decision makers are often interested by some sets of specific values that the vector X_t can take. Such a set $\mathscr{S}$ is called a *solution* which is by nature a subset of domain $\mathscr{D}$, *i.e.*, $\mathscr{S} \subseteq \mathscr{D}$.

The main goals of a stochastic simulation are to determine if a solution $\mathscr{S}$ is possible and its probability $\mathbb{P}(X_t \in \mathscr{S}) = \theta_{\mathscr{S}}$ at time step t (in general the last time step). In the general case, obtaining an analytical expression of the distribution of X_t is virtually impossible. Hence the study of such a solution consists in building an estimator $\hat{\theta}_{\mathscr{S}}$ of $\theta_{\mathscr{S}}$ over a set of replications executed according to a specific policy.

2.2 Monte Carlo simulation

The Monte Carlo method (MC) is a classical execution policy to address this problem. It consists in simulating a number n of replications (*i.e.* obtaining n independent and identically distributed samples) and building an estimator (generally the Maximum likelihood estimator) $\hat{\theta}_{\mathscr{S}}$ of $\theta_{\mathscr{S}}$ defined as:

$$\hat{\theta}_{\mathscr{S}} = \frac{1}{n}\sum_{i=1}^{n} \mathbb{1}_{\mathscr{S}}(X_t^i) \tag{1}$$

where $\mathbb{1}_{\mathscr{S}}$ is the indicator function of solution $\mathscr{S}$ and X_t^i the value of observables in the i^{th} replication (Rubinstein and Kroese 2016). Of course, the number n has to be large enough for the estimator to be good. The minimal number of replications to obtain a good estimator $\hat{\theta}_{\mathscr{S}_i}$ of the probability $\mathbb{P}(X_{t,i} \in \mathscr{S}_i) = \theta_{\mathscr{S}_i}$ of some solution $\mathscr{S}_i$ where $X_{t,i}$ is one observable of the vector X_t is denoted $n(X_{t,i})$. Suppose a desired relative error ε for the estimator $\hat{\theta}_{\mathscr{S}_i}$ at confidence level $1-\alpha$:

$$\mathbb{P}(\frac{|\hat{\theta}_{\mathscr{S}_i} - \theta_{\mathscr{S}_i}|}{\theta_{\mathscr{S}_i}} \leq \varepsilon) \geq 1-\alpha. \tag{2}$$

The minimal value for $n(X_{t,i})$ to verify (2) can be approximated by applying the following algorithm (Banks et al. 2010):

1. Simulate n_0 replications. (n_0 observations $X_{t,i}^1, \ldots, X_{t,i}^{n_0}$)
2. Compute
$$n(X_{t,i}) = \lceil (\frac{Z_{1-(\frac{\alpha}{2})} \cdot \mathscr{S}_i}{\varepsilon \cdot \overline{X}_{t,i}})^2 \rceil \tag{3}$$
where $Z_{1-(\alpha/2)}$ is the $100(1-\alpha/2)$ quantile of the normal distribution, $\mathscr{S}_i$ stands for the sample standard deviation over the n_0 observations and $\overline{X}_{t,i}$ is the sample mean value over the n_0 observations. The conventional values for n_0, ε and α are respectively 150, 0.05 and 0.05.

2.3 Related execution policies

Other execution policies have been proposed to be used in specific contexts. The most relevant are briefly presented in the following.

If simulations rely on a Markov model and the solution of interest is unique and *rare*, methods such as splitting or importance sampling can be used.

The idea behind the *Splitting* methods consists in decomposing the target rare solution $\mathscr{S}$ into an event chain $E_0, ..., E_n$ which satisfies $\mathscr{S} = E_n \subset E_{n-1} \subset ... \subset E_0$. Initially, N replications from t_0 to t are simulated. Then, those reaching the event E_0 at time $t_t < t$ are cloned. Afterward, these cloned replications are simulated from t_t to t and those reaching E_1 are cloned once again. The process is repeated until $E_n = \mathscr{S}$ is reached (Villén-Altamirano and Villén-Altamirano 1994; L'Ecuyer et al. 2009). Finally, $\hat{\theta}_{\mathscr{S}}$ is computed from the product of conditional probabilities:

$$\hat{\theta}_{\mathscr{S}} = P(E_0)\prod_{i=1}^{n} P(E_{i+1}|E_i). \tag{4}$$

Importance Sampling (IS) relies on the substitution of the distribution of random variables. The basic idea is to obtain samples generated from a different distribution leading more often to the solution of interest (Glynn and Iglehart 1989). The probability estimation over these samples is biased, however it is possible to reconstruct an unbiased estimator combining the biased one and the relation between the two distributions.

The *polyagent* approach has been introduced by Parunak et al. to quickly characterize the mean trajectory of a stochastic multi-agent-based simulation (Parunak and Brueckner 2006). Periodically, agents generate a set of *ghosts*, *i.e.*, virtual agents that mimic their behavior and mark the explored trajectories with pheromones. Then, the agent goes up the gradient of the calculated pheromone field until it reaches its maximum, which defines its new state.

The idea behind *simulation cloning* is that if different replications generate the same trajectory, then, it is more efficient to simulate one replication and clone it at *decision points*, *i.e.*, when processing a non-deterministic event, to explore alternative futures (Hybinette and Fujimoto 1997; Hybinette and Fujimoto 2001). This approach allows one to perform "what-if" scenarios, *e.g.*, to determine the consequences of different scheduling algorithms in manufacturing simulations.

The policies presented in this section can be powerful and efficient, but being specific to a modeling paradigm or a particular problem, it is not straightforward to apply them in the general case. For instance, although splitting seems simple, we need to know the chain of events leading to the target solution. IS relies on Markov models and requires not only a full knowledge of the random variable distribution to skew but also low-level access to the pseudo-random number generator. The polyagent approach can only be applied on multiagent-based simulations and target one solution: the most probable. In the simulation cloning approach, decisions points must be known and their number should be limited to prevent combinatorial explosion.

3 SPSC: A GENERAL-PURPOSE EXECUTION POLICY

The proposed execution policy has been named according to the successive steps of the process: **S**imulation, **P**artitioning, **S**election and **C**loning.

3.1 General principle

Instead of executing the replications from t_0 to t_f as MC does, the temporal interval $[t_0, t_f]$ is cut into m sub-intervals $[t_{(i)}, t_{(i+1)}]$, $i = 0, \ldots, m-1$, such as:

$$[t_0, t_f] = \bigcup_{i=0}^{m-1} [t_{(i)}, t_{(i+1)}] \tag{5}$$

and

$$[t_{(i)}, t_{(i+1)}] \cap [t_{(j)}, t_{(j+1)}] = \begin{cases} t_{(i+1)} & \text{if } j = i+1, \\ [t_{(i)}, t_{(i+1)}] & \text{if } j = i, \\ \emptyset & \text{else,} \end{cases} \tag{6}$$

with $t_{(0)} = t_0$ and $t_{(m)} = t_f$.

Then, for each sub-interval $[t_{(i)}, t_{(i+1)}]$, the following steps, illustrated by Figure 1, are applied:

Simulation: Execute N replications from $t_{(i)}$ to $t_{(i+1)}$.

Partitioning: At time step $t_{(i+1)}$, retrieve the observables from the states of each replication then partition these N states into K subsets according to these observables.

Selection: Select one or more representative replications in each subset. Then, delete the non selected replications.

Cloning: Clone the selected replications to reach N replications in total (to control the computational cost).

Once the replications reach the final time step t_f, the results can be analyzed to obtain an estimation of the probability of the solutions.

3.2 Estimation of the probability of solutions

As shown in Section 2.1, our main goals are the detection of possible solutions at final time step t_f and the estimation of their probabilities. We recall here some notations: the observables of a replication at t_f are represented by a random vector X_{t_f}, the interesting solution is denoted $\mathscr{S}$, its probability $\mathbb{P}(X_{t_f} \in \mathscr{S})$ and the probability estimator $\hat{\theta}_{\mathscr{S}}$.

Contrary to MC, the Maximum likelihood estimator cannot be applied to SPSC. Indeed, some replications are removed during the simulation and replaced by clones. Thus, we introduce an estimator suitable in our case, based on the decomposition of the target probability and its reconstruction.

Decomposition of the probability of interest

The theoretical probability $\mathbb{P}(X_{t_f} \in \mathscr{S})$ can be decomposed using the law of total probability. Considering an intermediate time step j, we can rewrite this probability as:

$$\mathbb{P}(X_{t_f} \in \mathscr{S}) = \sum_{\mathscr{S}_j \in \mathscr{P}_j} \mathbb{P}(X_{t_f} \in \mathscr{S} | X_j \in \mathscr{S}_j)\mathbb{P}(X_j \in \mathscr{S}_j) \tag{7}$$

where $\mathscr{P}_j$ is a partition of the state variable space X_j.

More generally, by assuming that the studied system has the Markov property, *i.e.*, X_i depends only on X_{i-1}, the decomposition can then be extended:

$$\mathbb{P}(X_{t_f} \in \mathscr{S}) = \sum_{\substack{\mathscr{S}_{m-1} \in \mathscr{P}_{m-1} \\ \vdots \\ \mathscr{S}_1 \in \mathscr{P}_1}} \prod_{i=0}^{m-1} \mathbb{P}(X_{i+1} \in \mathscr{S}_{i+1} | X_i \in \mathscr{S}_i) \tag{8}$$

where $\mathscr{P}_i$, $i = 1, ..., m-1$ is the partition of the state variable space X_i, $\mathscr{S}_T = \mathscr{S}$ and $\mathscr{S}_0$ are the initial states of the simulation.

Reconstitution of probability estimators

In this Section, we present a method allowing to compute the estimator of $\mathbb{P}(X_{t_f} \in \mathscr{S})$ inspired by the probability decomposition presented in Equation (8).

Once SPSC iterations are over, we have for each intermediate time step $t_{(i)}$ a partition $\mathscr{P}_{(i)}$ for the state variable space $X_{t_{(i)}}$. For an element $\mathscr{S}_{(i)}$ of the partition $\mathscr{P}_{(i)}$, after the selection and cloning steps, n_i replications have then been cloned. From these n_i cloned replications, some (n_{i+1}) will be associated to an element $\mathscr{S}_{(i+1)} \in \mathscr{P}_{(i+1)}$ after the partitioning step at time step $t_{(i+1)}$. The quantities n_i and n_{i+1} can then be used to estimate the conditional probability $\mathbb{P}(X_{t_{(i+1)}} \in \mathscr{S}_{(i+1)} | X_{t_{(i)}} \in \mathscr{S}_{(i)})$ by:

$$\hat{P}(X_{t_{(i+1)}} \in \mathscr{S}_{(i+1)} | X_{t_{(i)}} \in \mathscr{S}_{(i)}) = \frac{n_{i+1}}{n_i} \tag{9}$$

Eventually, we can define a new estimator $\hat{\theta}_{\mathscr{S}}$ for $\mathbb{P}(X_{t_f} \in \mathscr{S})$ from Equation (9), using a decomposition similar to Equation (8):

$$\hat{\theta}_{\mathscr{S}} = \sum_{\substack{\mathscr{S}_{(m-1)} \in \mathscr{P}_{(m-1)} \\ \vdots \\ \mathscr{S}_{(1)} \in \mathscr{P}_{(1)}}} \prod_{i=0}^{m-1} \hat{P}(X_{t_{(i+1)}} \in \mathscr{S}_{(i+1)} | X_{t_{(i)}} \in \mathscr{S}_{(i)}) \tag{10}$$

where $\mathscr{S}_{(m)} = \mathscr{S}$ and $\mathscr{S}_{(0)} = \mathscr{S}_0$.

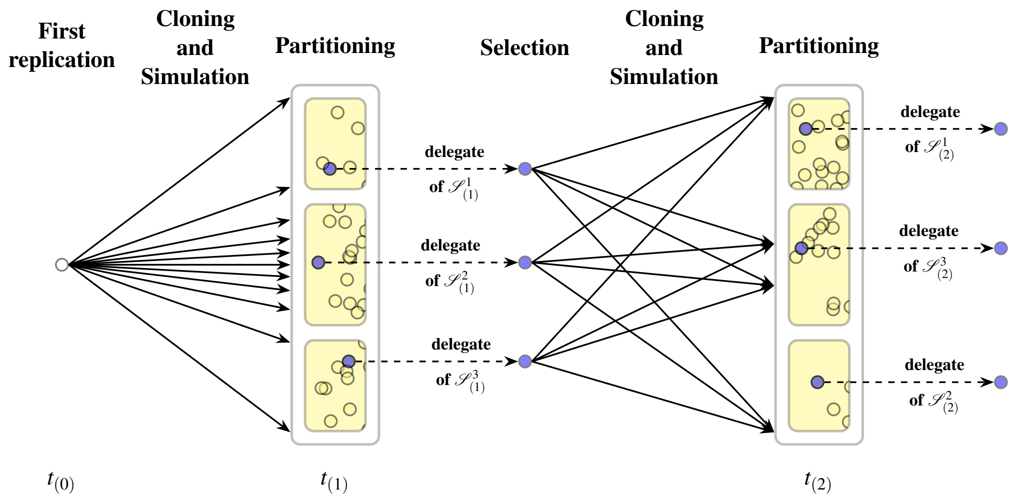

First replication
Cloning and Simulation
Partitioning
Selection
Cloning and Simulation
Partitioning
delegate of $\mathscr{S}^1_{(1)}$
delegate of $\mathscr{S}^2_{(1)}$
delegate of $\mathscr{S}^3_{(1)}$
delegate of $\mathscr{S}^1_{(2)}$
delegate of $\mathscr{S}^3_{(2)}$
delegate of $\mathscr{S}^2_{(2)}$
$t_{(0)}$
$t_{(1)}$
$t_{(2)}$

Figure 1: Illustration of the two first cuttings of SPSC ($[t_{(0)}, t_{(2)}]$).

3.3 Implementation

To conduct our experiments, we implemented SPSC on the top of the Similar agent-based simulation engine (Morvan et al. 2011).

SPSC itself was developed in Python to benefit from the various machine learning libraries available in this programming language such as sklearn (Pedregosa et al. 2011) and Keras (Chollet 2015).

In the following we present in detail the chosen implementation of the different steps of SPSC:

Simulation: N replications are executed.

Partitioning: To partition the states of these N replications in the space of the observables, we use an existing unsupervised learning algorithm assigning each replication to a subgroup. Our approach was tested with four well-known clustering algorithms (Aggarwal and Reddy 2013; Jain, Murty, and Flynn 1999): *kmeans*, *kmedoids*, *agglomerative*, and *DBSCAN* (Ester, Kriegel, Sander, and Xu 1996; Schubert, Sander, Ester, Kriegel, and Xu 2017) using 3 values for the number of partitions k: $5, 10$ and 15.

Selection: From any element $\mathscr{S}_{(i)} \in \mathscr{P}_{(i)}$, we select the most representative replication of each cluster which is the nearest to the center using euclidean distance.

Cloning: After partitioning and selection, k delegates are obtained to be cloned. To come back to N replications in total, we clone each delegate $\lfloor \frac{N}{k} \rfloor$ times. If k does not divide N, we select randomly the remainder number $N - k * \lfloor \frac{N}{k} \rfloor$ of delegates to produce one more clone per selected delegate.

The time interval $[0, T]$ is homogeneously split into $m = 1, 2, 3$ and 4 sub-intervals.

We can note that, compared to MC, SPSC implies additional computations in the partitioning and cloning steps. However, this extra cost is negligible compared to the simulation one, which allows us to compare the performance between MC and SPSC using the same number of replications.

4 EXPERIMENTS

In this section, we present use cases of SPSC on three different models. For each one of them, we follow this general guideline:

1. Identify the solutions of interest.
2. Estimate the reference probability value (as we do not have analytical solutions).
3. Run simulation batches using MC and SPSC with the same amount of resources.
4. Statistically compare the results according to the metrics presented in Section 4.1

4.1 Policies evaluations and comparisons

To evaluate a stochastic simulation execution policy and its associated probability estimator, two performance measures can be computed from a batch of replications: the detection capabilities of the policy, *i.e.*, its ability to reach a possible solution and the precision of its probability estimator.

Detection To evaluate the detection capabilities of an execution policy for a specific solution $\mathscr{S}$, the following indicator function is used:

$$Detect(\mathscr{S}) = \begin{cases} 1 & \text{if one of the replication reaches } \mathscr{S} \\ 0 & \text{else.} \end{cases} \tag{11}$$

For a set of solutions $\mathbb{S}$, we measure the detection of all the solutions simultaneously:

$$Detect(\mathbb{S}) = \begin{cases} 1 & \text{if } \forall \mathscr{S} \in \mathbb{S}, Detect(\mathscr{S}) = 1 \\ 0 & \text{else.} \end{cases} \tag{12}$$

Precision To evaluate the precision of a probability estimator for a given policy on all the solutions simultaneously, the mean of the relative errors of the solutions is considered:

$$Err(\mathbb{S}) = \frac{1}{\#\mathbb{S}} \sum_{\mathscr{S}\in\mathbb{S}} \left|\frac{\hat{P}(\mathscr{S}) - P_{ref}(\mathscr{S})}{P_{ref}(\mathscr{S})}\right|, \tag{13}$$

where $P_{ref}(\mathscr{S})$ is the reference probability of the solution $\mathscr{S}$, and $\#\mathbb{S}$ is the cardinal of $\mathbb{S}$.

Now, in order to compare the performance of two different policies, SPSC and MC in this article, multiple batches are executed for each policy. The previous metrics are computed for each batch and we retain in particular the mean value per policy, subsequently denoted by $\overline{Detect_{SPSC}}$ and $\overline{Err_{SPSC}}$ for SPSC, and $\overline{Detect_{MC}}$ and $\overline{Err_{MC}}$ for MC. Thus, to compare the actual gain of a policy over another, we use the following indicators:

Relative detection rate gain in % To evaluate the gain of detection rate of SPSC over MC, the following measure is used:

$$\mathscr{G}_{\text{Detect}}(\mathbb{S}) = \frac{\overline{Detect_{SPSC}}(\mathbb{S}) - \overline{Detect_{MC}}(\mathbb{S})}{\overline{Detect_{MC}}(\mathbb{S})} \times 100 \tag{14}$$

Relative precision gain in % To evaluate the gain of precision of estimations of SPSC over MC, the following measure is used:

$$\mathscr{G}_{\text{Precision}}(\mathbb{S}) = \frac{\overline{Err_{MC}}(\mathbb{S}) - \overline{Err_{SPSC}}(\mathbb{S})}{\overline{Err_{MC}}(\mathbb{S})} \times 100 \tag{15}$$

Relative precision gain without failure in % To evaluate the precision of the estimation of SPSC over MC when all solutions are successfully detected, we consider the precision gain slightly modified from (15):

$$\mathscr{G}'_{\text{Precision}}(\mathbb{S}) = \frac{\overline{Err_{MC}}'(\mathbb{S}) - \overline{Err_{SPSC}}'(\mathbb{S})}{\overline{Err_{MC}}'(\mathbb{S})} \times 100 \tag{16}$$

where $\overline{Err}'(\mathbb{S})$ represents the mean value of $Err(\mathbb{S})$ computed from batches that verify $Detect(\mathbb{S}) = 1$.

4.2 Prey-predator

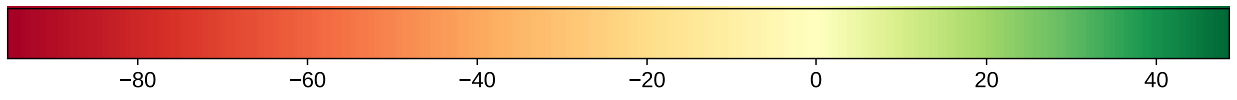
We consider an agent-based prey-predator model similar to the ”Wolf-Sheep” model shipped with NetLogo (Wilensky 1999) as a first example. The observables of this model are the number of living predators and preys. The following possible solutions can be considered:

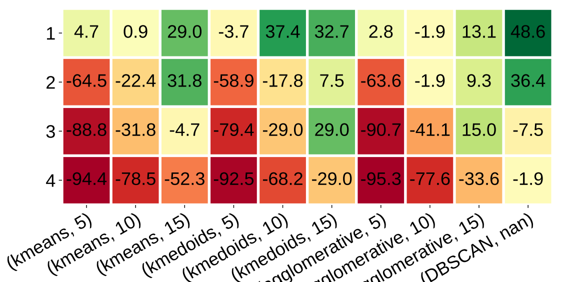
- $\mathscr{S}_1$: One species survives, extinction of prey and predator species, only vegetation remains .
- $\mathscr{S}_2$: Two species survive, extinction of the predator species, only preys and vegetation remain.
- $\mathscr{S}_3$: The three species survive and form a stable ecosystem with pseudo-periodical oscillations of their populations.

First, we run 900000 replications using MC with $t_f = 1000$ in order to provide reference values for the comparisons:

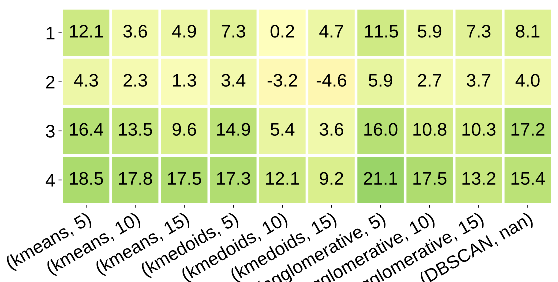
$$P_{ref}(\mathscr{S}_1) \approx 0.0065, P_{ref}(\mathscr{S}_2) \approx 0.0126, P_{ref}(\mathscr{S}_3) \approx 0.981. \tag{17}$$

Then, we run 1000 batches of $N = 50$ replications for each policy (MC and different configurations of SPSC). The detection rate and the precision gains of SPSC over MC are shown in Figure 2.

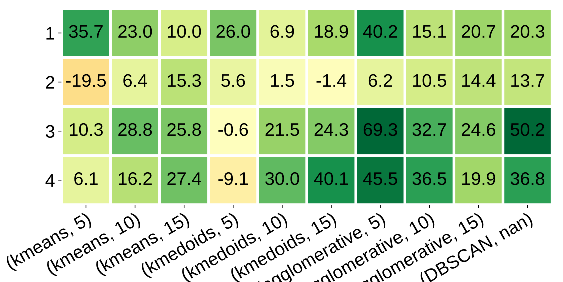




(a) Relative detection rate gain $\mathscr{G}_{\text{Detect}}(\mathbb{S})$ of SPSC over MC



(b) Relative precision gain $\mathscr{G}_{\text{Precision}}$ of SPSC over MC



(c) Relative precision gain without failure $\mathscr{G}'_{\text{Precision}}$ of SPSC over MC

Figure 2: Relative detection and precision gains of SPSC over MC in % for different configurations tested on the prey-predator model. The ordinate indicates the number m of sub-intervals and the abscissa represents the couple of clustering method used and number k of clusters used.

4.3 Virus

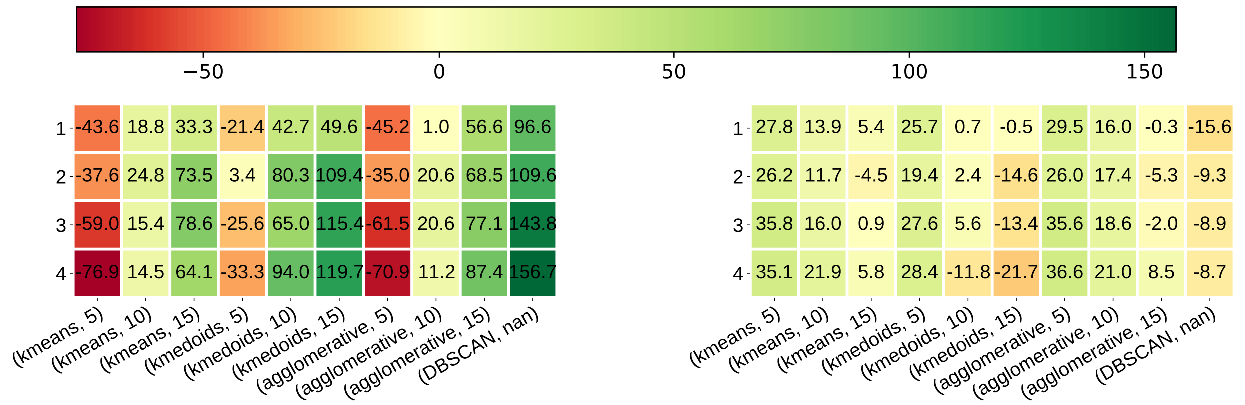
We then consider an agent-based model of virus spreading, similar to the one shipped with NetLogo as a second example. The observables of this model are the number of infected and healthy agents. The following possible solutions can be considered:

- $\mathscr{S}_1$: the virus disappears completely from the environment,
- $\mathscr{S}_2$: the virus remains in the environment.

We used the same settings as in the previous example: reference probabilities were obtained from 500000 replications with $t_f = 250$.

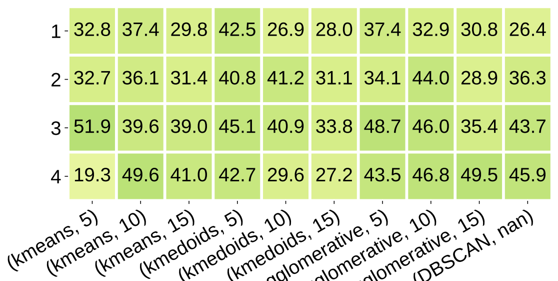
$$P_{ref}(\mathscr{S}_1) \approx 0.002644, \qquad P_{ref}(\mathscr{S}_2) \approx 0.997356$$

Then, as previously, 1000 batches of $N = 50$ replications are executed for each policy (MC and different configurations of SPSC). The detection rate and the precision gains of SPSC over MC are shown in Figure 3.



(a) Relative detection rate gain $\mathscr{G}_{\text{Detect}}(\mathbb{S})$ of SPSC over MC

(b) Relative precision gain $\mathscr{G}_{\text{Precision}}$ of SPSC over MC



(c) Relative precision gain without failure $\mathscr{G}'_{\text{Precision}}$ of SPSC over MC

Figure 3: Relative detection and precision gains of SPSC over MC in % for different configurations tested on the virus model. The axes are the same as those in Figure 2.

4.4 Microscopic traffic simulation

In this section SPSC is evaluated on a more complex use case: a microscopic traffic simulation in which agents move in the road network from an origin to a destination following a roadmap and driving rules.

An illustration of the network used in our experiments is given in Figure 4. Possible origins of vehicles are O_1, O_2, O_3 and O_4 and possible destinations are D_1 and D_2. Roadmaps are randomly generated from possible paths between origins and destinations. Agents behave according to the car-following model IDM (Treiber et al. 2000) and the French driving rules.

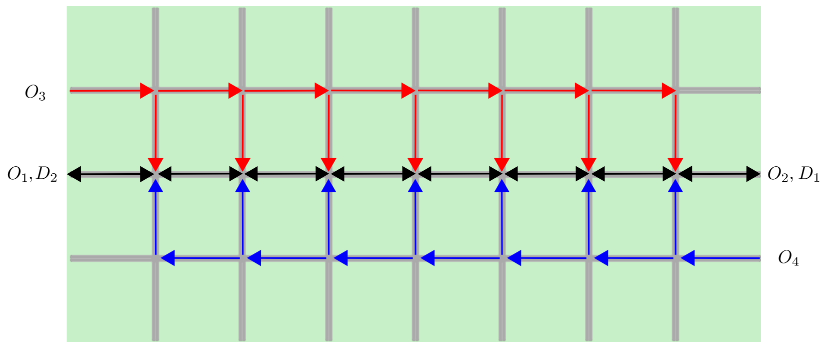


Figure 4: Experimental setup for the traffic simulation. Possible origins of vehicles are O_1, O_2, O_3 and O_4 and possible destinations are D_1 and D_2.

We consider, for each lane, information that allows us to determine the congestion level (*i.e.* mean speed and lane occupation). Therefore, the observables retrieved from a simulation state are multidimensional spatial data. Using such raw data for the partition step could lead to bad results. Thus, an autoencoder is trained beforehand on MC data in order to provide low dimensional characteristics for the partition step (Rumelhart et al. 1985; Goodfellow et al. 2016).

We assume that the local traffic state is the main indicator for the decision maker. We split the network into 3 sectors: north, center and south (corresponding respectively to red, black and blue arrows in Figure 4). We consider 2 possible traffic states within a sector: *fluid*, *i.e.* at most 2 lanes exceed experimental critical density and *congested*, *i.e.* at least 7 lanes exceed experimental critical density. Hence, we can define a set $\mathbb{S}$ of 8 solutions (2 traffic states for 3 independent sectors):

- $\mathscr{S}_1$: fluid $\times$ fluid $\times$ fluid
- $\mathscr{S}_2$: fluid $\times$ fluid $\times$ congested
- $\mathscr{S}_3$: fluid $\times$ congested $\times$ fluid
- $\mathscr{S}_4$: fluid $\times$ congested $\times$ congested
- $\mathscr{S}_5$: congested $\times$ fluid $\times$ fluid
- $\mathscr{S}_6$: congested $\times$ fluid $\times$ congested
- $\mathscr{S}_7$: congested $\times$ congested $\times$ fluid
- $\mathscr{S}_8$: congested $\times$ congested $\times$ congested

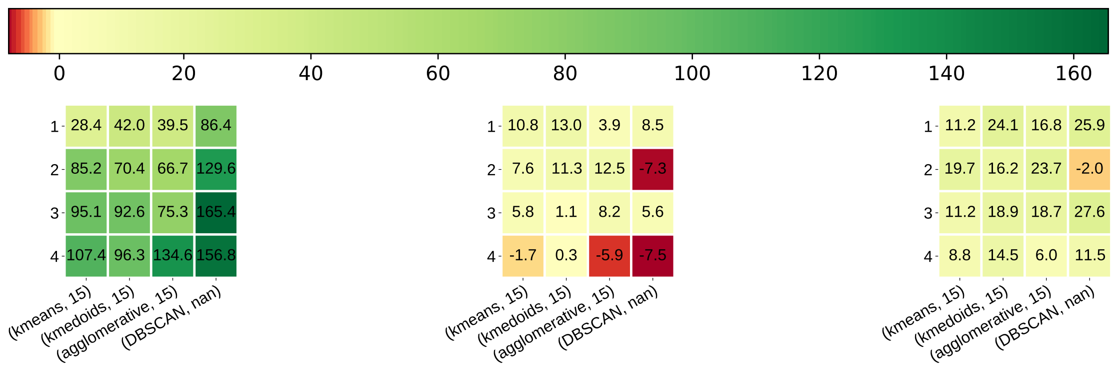
The reference values of the probability of these solutions are estimated by executing 175000 replications using MC with $t_f = 72000$:

$$\begin{array}{llll} P_{ref}(\mathscr{S}_1) \approx 0.00692 & P_{ref}(\mathscr{S}_2) \approx 0.00023 & P_{ref}(\mathscr{S}_3) \approx 0.0048 & P_{ref}(\mathscr{S}_4) \approx 0.00114 \\ P_{ref}(\mathscr{S}_5) \approx 0.00019 & P_{ref}(\mathscr{S}_6) \approx 0.00022 & P_{ref}(\mathscr{S}_7) \approx 0.00146 & P_{ref}(\mathscr{S}_8) \approx 0.00437 \end{array} \tag{18}$$

Then, to compare the performances of MC and SPSC (with the number of clusters k fixed to 15), 500 batches of $N = 50$ replications are executed for each policy. Since it is virtually impossible to detect all the possible solutions with only 50 replications, we modify slightly the detection metric by measuring the capability of detecting at least 2 different solutions. The detection and precision gains of SPSC over MC are shown in Figure 5.

4.5 Discussion

From the results obtained on the three examples tested here, we can conclude that in some configurations, SPSC outperforms MC in terms of the first considered metric: the detection rate (See Figures 2a, 3a, 5a).



(a) Relative detection rate gain $\mathscr{G}_{\text{Detect}}(\mathbb{S})$ of SPSC over MC

(b) Relative precision gain $\mathscr{G}_{\text{Precision}}$ of SPSC over MC

(c) Relative precision gain without failure $\mathscr{G}'_{\text{Precision}}$ of SPSC over MC

Figure 5: Relative detection and precision gains of SPSC over MC in % for different configurations tested on the microscopic traffic simulation. The axes are the same as those in Figure 2.

When dealing with multiple solutions, it works extremely well with a high number of clusters k. In contrast, using a small k can lead to the non detection of some solutions. An explanation for this phenomenon is that when using a clustering algorithm with a small k, some specific features of the data points could potentially be hidden if the point is assigned to a large cluster. It has to be noted that the DBSCAN algorithm, that searches automatically for potential outliers and quantifies the optimal number of clusters outperforms other clustering algorithms according to this metric.

The number m of sub-intervals seems to have an impact on the detection rate. For the prey-predator and virus models, the detection rate decreases in general while increasing m. Contrarily with the transport model, the detection rate is better with high value of m. A possible reason is that the prey-predator and virus models have short time interval ($t_f \leq 1000$). Thus, replication clones might not have enough time to evolve to interesting solutions. Conversely, the time interval for the transport model is relatively large ($t_f = 72000$) and replication clones have enough time to evolve even with $m = 4$.

On the other hand, the relative errors obtained by SPSC are lower than MC in most of the cases (See Figures 2b, 3b, 5b). A tendency that can easily be observed is that bad detection rates are related to good errors and vice versa. This can be explained by the fact that the value of the reference probability is very small, which leads to a paradoxical situation: the non detection of such a solution (the probability is estimated by 0 in this case) results in a very small relative error. However, when considering the precision gain without failure (i.e., when the execution policy did not fail to detect the solution), it is clear that SPSC allows one to obtain better estimates (See Figures 2c, 3c, 5c). Once again, DBSCAN turns to be one of the best clustering algorithms among those tested in these experiments.

5 CONCLUSIONS AND PERSPECTIVES

We have presented in this paper an execution policy named SPSC which can be more efficient than MC with limited resources. Contrary to existing specialized methods discussed in Section 2.3, SPSC is generic and easy to apply, since the simulations are treated as black boxes and no specific information about the possible solutions is needed. The only operational constraint imposed by SPSC on the simulation environment is the ability to clone replications.

Some configurations (using different partitioning algorithms and k) of SPSC were tested with three different stochastic multi-agent-based simulations. The results comparison shows that, using the appropriate configuration, SPSC detects the possible solutions and estimate their probabilities better than MC, especially when the clustering algorithm DBSCAN is used for the partition step. On the other hand, in this study, replications were cloned with the same rate. Investigating the impact of different cloning policies on the quality of the results would be interesting.

The microscopic traffic simulation presented in Section 4.4 was based on a simple network topology. From an experimental point of view, our main perspective is to validate SPSC on a realistic multi-modal network at the city scale. From a more theoretical point of view, an extension of SPSC, using modern statistical inference tools (*e.g.*, based on belief functions (Kanjanatarakul et al. 2016)) and probabilistic clustering algorithms (Ross 2010) can be envisaged as we deal with small samples at each intermediate time step.

ACKNOWLEDGEMENT

The ELSAT2020 project is co-financed by the European Union with the European Regional Development Fund, the French state and the Hauts-de-France Region Council.

REFERENCES

Aggarwal, C. C., and C. K. Reddy. (Eds.) 2013. *Data Clustering: Algorithms and Applications*. Boca Raton, Florida: Chapman and Hall/CRC Press.

Banks, J., J. Carson II, B. Nelson, and D. Nicol. 2010. *Discrete-event system simulation*. Fifth ed. Upper Saddle River, New Jersey: Pearson Education, Inc.

Chollet, F. 2015. “Keras. https://github.com/fchollet/keras”. accessed 19th August.

Ester, M., H.-P. Kriegel, J. Sander, and X. Xu. 1996. “A density-based algorithm for discovering clusters in large spatial databases with noise”. In *2nd International Conference on Knowledge Discovery and Data Mining*, edited by E. Simoudis, J. Han, and U. Fayyad, 226–231. Palo Alto, California: AAAI press.

Glynn, P. W., and D. L. Iglehart. 1989. “Importance sampling for stochastic simulations”. *Management science* 35(11):1367–1392.

Goodfellow, I., Y. Bengio, and A. Courville. 2016. *Deep learning*. Cambridge, Massachusetts: MIT press.

Huang, Y.-L., G. Morvan, F. Pichon, and D. Mercier. 2019. “SPSC: a new execution policy for exploring discrete-time stochastic simulations”. In *International Conference on Principles and Practice of Multi-Agent Systems*, edited by M. Baldoni, M. Dastani, B. Liao, Y. Sakurai, and R. Z. Wenkstern, 568–575. New York: Springer.

Hybinette, M., and R. Fujimoto. 1997. “Cloning: A novel method for interactive parallel simulation”. In *Proceedings of the 1997 Winter Simulation Conference*, edited by S. Andradóttir, K. J. Healy, D. H. Withers, and B. L. Nelson, 444–451. NW Washington, DC: Institute of Electrical and Electronics Engineers Computer Society.

Hybinette, M., and R. Fujimoto. 2001. “Cloning parallel simulations”. *ACM Transactions on Modeling and Computer Simulation (TOMACS)* 11(4):378–407.

Jain, A. K., M. N. Murty, and P. J. Flynn. 1999. “Data clustering: a review”. *ACM Computing Surveys* 31(3):264–323.

Kanjanatarakul, O., T. Denœux, and S. Sriboonchitta. 2016. “Prediction of future observations using belief functions: A likelihood-based approach”. *International Journal of Approximate Reasoning* 72:71–94.

L’Ecuyer, P., F. Le Gland, P. Lezaud, and B. Tuffin. 2009. “Splitting Techniques”. In *Rare Event Simulation using Monte Carlo Methods*. Hoboken, New Jersey: John Wiley & Sons.

Morvan, G., A. Veremme, and D. Dupont. 2011. “IRM4MLS: the influence reaction model for multi-level simulation”. In *Multi-Agent-Based Simulation XI*, Volume 6532 of *LNCS*, 16–27. New York: Springer.

Parunak, H., and S. Brueckner. 2006. “Concurrent Modeling of Alternative Worlds with Polyagents”. In *Proceedings of the 2006 International Conference on Multi-Agent-Based Simulation VII*, edited by L. Antunes and K. Takadama, 128–141. Berlin, Heidelberg: Springer-Verlag.

Pedregosa, F., G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. “Scikit-learn: Machine Learning in Python”. *Journal of Machine Learning Research* 12:2825–2830.

Ross, T. 2010. *Fuzzy logic with engineering applications*. Third ed. Hoboken, New Jersey: John Wiley & Sons.

Rubino, G., B. Tuffin et al. 2009. *Rare event simulation using Monte Carlo methods*, Volume 73. Hoboken, New Jersey: John Wiley & Sons.

Rubinstein, R. Y., and D. P. Kroese. 2016. *Simulation and the Monte Carlo method*. Third ed. Hoboken, New Jersey: John Wiley & Sons.

Rumelhart, D. E., G. E. Hinton, and R. J. Williams. 1985. “Learning Internal Representations by Error Propagation”. Technical report, California University, San Diego. La Jolla Institute for Cognitive Science.

Schubert, E., J. Sander, M. Ester, H. P. Kriegel, and X. Xu. 2017. “DBSCAN revisited, revisited: why and how you should (still) use DBSCAN”. *ACM Transactions on Database Systems (TODS)* 42(3):1–21.

Treiber, M., A. Hennecke, and D. Helbing. 2000. “Congested traffic states in empirical observations and microscopic simulations”. *Physical review E* 62(2):1805.

Villén-Altamirano, M., and J. Villén-Altamirano. 1994. “RESTART: a straightforward method for fast simulation of rare events”. In *Proceedings of the 1994 Winter Simulation Conference*, edited by J. Tew, M. Manivannan, D. Sadowski, and A. Seila, 282–289. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Wilensky, U. 1999. “NetLogo.http://ccl.northwestern.edu/netlogo/, Center for Connected Learning and Computer-Based Modeling, Northwestern University. Evanston, Illinois.”. accessed 19th August.

AUTHOR BIOGRAPHIES

YU-LIN HUANG is a Ph.D. student in computer science at Artois University. His research interests include agent-based simulation, high performance computing and machine learning. His email address is ylin.huang@univ-artois.fr. His website is https://www.lgi2a.univ-artois.fr/spip/fr/annuaire/yu-lin-huang.

GILDAS MORVAN is an associate professor in computer science at Université d’Artois where he obtained his Ph.D. in 2009. His research deals with agent-based modeling and multi-level simulation. He is the main developer and maintainer of the multiagent-based simulation platform SIMILAR (ttps://github.com/gildasmorvan/similar/). His email address is gildas.morvan@univ-artois.fr. His website is https://www.lgi2a.univ-artois.fr/~morvan/.

FREDERIC PICHON is a professor in computer science at Artois University and a member of the LGI2A. He received a Ph.D. degree in information and communication systems from the University of Technology of Compiègne, France, in 2009. His research focuses on uncertain reasoning and information fusion. His email address is frederic.pichon@univ-artois.fr. His website is https://www.lgi2a.univ-artois.fr/~pichon/.

DAVID MERCIER is a professor in computer science at Université d’Artois, France. He earned a Ph.D. in information and communication systems from the University of Technology of Compiègne, France, in 2006. His main research interests include information fusion and reasoning with uncertainty in particular with the Dempster-Shafer theory of evidence. His email address is david.mercier@univ-artois.fr. His website is https://www.lgi2a.univ-artois.fr/~mercier.