mathematics

MDPI

Article

Comparison of Continuous-Time Partial Markov Chain Construction by Heuristic Algorithms for Purpose of Approximate Transient Analysis

Eimutis Valakevičius [1], Mindaugas Bražėnas [1] and Tomas Ruzgas [2,*]

[1] Department of Mathematical Modelling, Kaunas University of Technology, Studentų g. 50, LT-44239 Kaunas, Lithuania; eimutis.valakevicius@ktu.lt (E.V.); mindaugas.brazenas@ktu.lt (M.B.)
[2] Department of Applied Mathematics, Kaunas University of Technology, Studentų g. 50, LT-44239 Kaunas, Lithuania
* Correspondence: tomas.ruzgas@ktu.lt

Abstract: We investigate the construction of a partial absorbing continuous-time Markov chain (CTMC) using a heuristic algorithm aimed at approximate transient analysis. The accuracy of transient state probabilities is indicated by the probability of absorbing state(s) at the specified time moment. A key challenge is the construction of a partial CTMC that minimizes the probability of reaching the absorbing state(s). The generation of all possible partial CTMCs is too computationally demanding, in general. Thus, we turn to investigation of heuristic algorithms that chose to include one state at a time based on limited information (i.e., the partial chain that is already constructed) and without any assumptions about the structure of the underlying CTMC. We consider three groups of such algorithms: naive, based on state characterization by the shortest path (obtained by Dijkstra method) and based on exact/approximate state probabilities. After introducing the algorithms, we discuss the problem of optimal partial CTMC construction and provide several examples. Then we compare the algorithm performance by constructing the partial CTMCs for two models: car sharing system and a randomly generated CTMC. Our obtained numerical results suggest that heuristic algorithms using state characterization via the shortest path offer a balance between accuracy and computational effort.

Keywords: continuous-time Markov chain; approximate transient analysis; Dijkstra method; car sharing system

MSC: 60J22; 65C40

Academic Editor: Michael Voskoglou

Received: 9 December 2024
Revised: 4 January 2025
Accepted: 10 January 2025
Published: 16 January 2025

Citation: Valakevičius, E.; Bražėnas, M.; Ruzgas, T. Comparison of Continuous-Time Partial Markov Chain Construction by Heuristic Algorithms for Purpose of Approximate Transient Analysis. *Mathematics* **2025**, *13*, 274. https://doi.org/10.3390/math13020274

1. Introduction

Continuous-time Markov chains (CTMCs) provide analytical methods to compute the transient characteristics of models defined over a countable state space [1,2]. In practice, the state space of CTMCs becomes very large, especially when phase-type distributions [3] are used to model the time durations of non-exponential distributions. When an exact evaluation of the entire model is not feasible, various approximate analysis methods can be employed. One such approach is state aggregation [4–7]. Subsequently, transient probabilities can be computed using the well-acknowledged uniformization method [8–11].

We address the issue of constructing a partial CTMC, specifically focusing on the truncation problem. In a related study [12], the partial CTMC, referred to as the 'active set', is iteratively updated during the computation of transient probabilities using the adaptive

uniformization method. Alternatively, ref. [13] proposes a heuristic approach for partial CTMC construction that employs the Dijkstra algorithm [14].

This investigation continues our work from [15] and is closely related to the study by [13]. In our initial research [15], we explored the generation of partial CTMCs to find one that provides the most accurate approximation of the model's transient state probabilities. However, the sheer number of possible partial CTMCs is too large for practical applicability, even for small CTMCs. Consequently, we attempted to influence the order of partial CTMC generation to select a good partial CTMC without generating all possible ones. Despite these efforts, we were unable to achieve conclusive results or offer practical recommendations.

In this paper, we have a different criterion to evaluate the quality of constructed partial CTMCs. In [15], we aimed to find a partial CTMC that reaches the specified absorbing probability at the latest time moment t possible. In contrast, this paper focuses on finding partial CTMC with the smallest absorbing probability for the specified time moment t. This criterion requires less computational effort to evaluate. To address the practical aspects of partial CTMC construction, we investigate heuristic algorithms ranging from naive and fast to more complex. The first group of heuristic algorithms is straightforward to implement. The second group constructs partial CTMCs based on edge state characterization by the shortest path. The third group characterizes edge states based on all possible paths of fixed or infinite length.

In the recent scientific literature, most of the results are obtained by investigating transient analysis methods for particular CTMC classes, for example, queueing models [16–18]. The specific assumptions about the Markov chain structure allow to develop very effective analysis methods. However, in this paper we are interested in a more general approach that could be applied to CTMCs of complex structure (for example, to CTMCs of car sharing system). Considering that state probability evaluation is a computationally demanding task, in an ideal situation, we would like to come up with a procedure that is able to identify an optimal partial CTMC (i.e., with the smallest absorbing state(s) probability). Before investigating the problem of optimal partial CTMCs, in this paper, we try to find out what can be achieved without making assumptions about the CTMC structure and applying heuristic algorithms, ranging from naive to computationaly demanding.

Our main result is a thorough investigation of the application of various heuristic algorithms for partial CTMC generation. We conclude that heuristic algorithms based on the shortest path, considering their runtime and quality of constructed partial CTMCs, are the most useful class of algorithms to construct a good partial CTMC. However, the problem of optimal partial CTMC identification is quite complex, which we have demonstrated in a separate section of examples.

The rest of the paper is organized as follows: Section 2 provides general information and notation, and Section 3 presents the heuristic algorithms for partial CTMC construction. To illustrate the problem of constructing partial CTMCs using heuristic algorithms, we included several examples in Section 4. Section 5 describes the conducted numerical experiments and presents the results. The discussion and conclusions are given in Sections 6 and 7.

2. Background

Let the set of full CTMC states be denoted as

$$\boldsymbol{\mathcal{S}} = \left\{\vec{s}_1, \vec{s}_2, \ldots, \vec{s}_\ell, \ldots, \vec{s}_{|\boldsymbol{\mathcal{S}}|}\right\}, \tag{1}$$

where each state $\vec{s}_\ell$ is represented by a vector, and is associated with a unique index $\ell \in \{1, 2, \ldots\}$. For convenience, we refer to a particular state either by its vector repre-

sentation $\vec{s}_\ell$ or its index ℓ interchangeably. We assume that the high-level implicit model description is abstracted by a CTMC generator g, which is a mapping:

$$g(\ell) \to \{\ell_1, \ell_2, \ldots, \ell_{n_\ell}\}, \tag{2}$$

where n_ℓ is a number of transitions from state ℓ to state ℓ_k ($k = 1, 2, \ldots, n_\ell$) with transition rate $\lambda_{\ell,\ell_k} > 0$. A state $\vec{s}_\ell$ is said to be absorbing if $n_\ell = 0$, and for such a state, we have $g(\ell) \to \emptyset$.

If the high-level implicit model description is given by event formalism, the CTMC generator can be realized by Algorithm 1.

Algorithm 1 Algorithm of the CTMC generator. Input parameters: ℓ—the state to explore.

```
1:  function: g (ℓ)
2:      ℰ ← set of events applicable to state s⃗_ℓ
3:      𝒯 ← ∅
4:      for each k^th event e in ℰ do
5:          apply event e to state s⃗_ℓ to get state s⃗_{ℓ_k} and λ_{ℓ,ℓ_k}
6:          store transition rate λ_{ℓ,ℓ_k}
7:          𝒯 ← 𝒯 ∪ {ℓ_k}
8:      end for
9:      return 𝒯
10: end of function
```

If the state $\vec{s}_{\ell_k}$ determined in Algorithm 1, line 5 has not been previously discovered, it is assigned an arbitrarily chosen unique index $\ell_k \in \{1, 2, \ldots\}$; the generated transition rate λ_{ℓ,ℓ_k} is stored for a convenient access by other algorithm routines.

Once all transitions from the state ℓ have been explored, the rate λ_ℓ of the exponential distribution of the sojourn time is

$$\lambda_\ell = \sum_{k=1}^{n_\ell} \lambda_{\ell,\ell_k}. \tag{3}$$

The set of partial CTMC states of size K is denoted as

$$\boldsymbol{\mathcal{P}} = \left\{\vec{s}_{\ell_1}, \vec{s}_{\ell_2}, \ldots, \vec{s}_{\ell_K}\right\} \subset \boldsymbol{\mathcal{S}}. \tag{4}$$

For a given partial CTMC $\boldsymbol{\mathcal{P}}$, we define its edge state set as

$$\overline{\boldsymbol{\mathcal{P}}} = \left\{\vec{s}_{\ell_k} \mid \vec{s}_{\ell_k} \notin \boldsymbol{\mathcal{P}}, \exists \vec{s}_\ell \in \boldsymbol{\mathcal{P}} : \lambda_{\ell,\ell_k} > 0\right\}. \tag{5}$$

In the rest of the paper, we assume that edge state set $\overline{\boldsymbol{\mathcal{P}}}$ is implied by $\boldsymbol{\mathcal{P}}$ and g.

Let Q be the infinitesimal generator matrix of the partial CTMC with states $\boldsymbol{\mathcal{P}} \cup \overline{\boldsymbol{\mathcal{P}}}$, then the vector of state transient probabilities is

$$\vec{p}(t) = \left(p_{\ell_1}(t), p_{\ell_2}(t), \ldots, p_{\ell_{|\boldsymbol{\mathcal{P}}|}}(t), \ldots, p_{\ell_{|\boldsymbol{\mathcal{P}}|+|\overline{\boldsymbol{\mathcal{P}}}|}}(t)\right). \tag{6}$$

Given the initial probability vector $\vec{p}(0)$ with the only non-zero element $p_{\ell_1}(0) = 1$, the state probability vector at time moment t is

$$\vec{p}(t) = \vec{p}(0)e^{tQ}, \tag{7}$$

and the probability of edge states $\overline{\boldsymbol{\mathcal{P}}}$ (i.e., absorbing probability) is

$$p^{(\mathrm{a})}(t) = \sum_{\ell \in \overline{\boldsymbol{\mathcal{P}}}} p_\ell(t). \tag{8}$$

3. Algorithms

The partial absorbing CTMC $\boldsymbol{\mathcal{P}}$ with K transient states is considered optimal if it has the smallest absorbing probability $p^{(\mathrm{a})}(t)$ for the specified time moment t. A brute-force approach to finding the optimal partial CTMC is to generate all possible partial CTMCs and select the one with the smallest absorbing probability. However, this approach is impractical for two main reasons:

1. The number of possible partial CTMCs increases rapidly with the number of states K [15];
2. The computation of probabilities of absorbing states is computationally intensive.

Therefore, instead of attempting to find an optimal partial CTMC through exhaustive enumeration, we explore the application of heuristic algorithms. We construct a partial CTMC sequentially by including one edge state at a time until the specified number of states is reached, as outlined in Algorithm 2.

Algorithm 2 Partial CTMC construction by including one edge state at a time. Input parameters: ℓ_1—an initial state; $K < |\boldsymbol{\mathcal{S}}|$—the size of the partial CTMC.

1: **function: partialCTMC** (ℓ_1, K)
2: $\quad \boldsymbol{\mathcal{P}}^{(1)} \leftarrow \{\ell_1\}$ ▷ The edge state set $\overline{\boldsymbol{\mathcal{P}}}$ is implied by (5).
3: $\quad$ **for** i **from** 2 **to** K **do**
4: $\quad\quad \ell_i \leftarrow$ the chosen state from $\overline{\boldsymbol{\mathcal{P}}}^{(i-1)}$
5: $\quad\quad \boldsymbol{\mathcal{P}}^{(i)} \leftarrow \boldsymbol{\mathcal{P}}^{(i-1)} \cup \{\ell_i\}$
6: $\quad$ **end for**
7: $\quad$ **return** $\boldsymbol{\mathcal{P}}^{(K)}$
8: **end of function**

The choices of edge states (line 4 of Algorithm 2) determine the final partial CTMC and its absorbing probability $p^{(\mathrm{a})}(t)$. To characterize an edge state $\ell \in \overline{\boldsymbol{\mathcal{P}}}^{(i)}$, we use the already generated partial CTMC (with states $\boldsymbol{\mathcal{P}}^{(i)} \cup \overline{\boldsymbol{\mathcal{P}}}^{(i)}$) and the specified time moment t. Based on a criterion rule and edge state characterization, a heuristic choice is made. It is important to note that the class of heuristic algorithms, as we defined them, does not contain an algorithm capable of producing an optimal partial CTMC of any size $K < |\boldsymbol{\mathcal{S}}|$ in general (see Example 2).

To aid our discussion regarding algorithm complexity in Section 6, we introduce the following notations. The number of edge states, after including the ith state, is denoted by

$$\nu(i) = \left|\overline{\boldsymbol{\mathcal{P}}}^{(i)}\right|, \tag{9}$$

and the number of newly introduced edge states then is

$$\Delta(i) = \begin{cases} \nu(i) - \nu(i-1) + 1, & i > 1, \\ \nu(i), & i = 1. \end{cases} \tag{10}$$

The number of transitions from state ℓ_i is denoted by

$$\tau(i) = |g(\ell_i)|. \tag{11}$$

To begin, we consider two naive edge state choice heuristics. The first algorithm, which chooses an edge state randomly, is referred to as RND. The second algorithm, that chooses to include the edge states in the order they are discovered, is referred to as SRND.

For the specified time moment t, one option in the process of partial CTMC construction is to choose an edge state with the highest probability. Such an algorithm is referred to as EXP.

Effectively, by computing edge state probabilities, as in the EXP algorithm, the states are characterized based on all possible paths from the initial state ℓ_1. As suggested by [13], each edge state can be characterized by a single path instead. They apply the Dijkstra algorithm [14] for some distance measure and choose to include the edge state at the shortest distance from the initial state ℓ_1. The proposed algorithm is fast and constructs a sufficiently good partial CTMC for reliability studies. Inspired by [13], we continue to investigate the idea of edge state characterization by a single shortest path. We also consider the application of the Dijkstra method. However, it might be that other shortest path graph algorithms (like the Bellman–Ford algorithm [19]) could prove to be more efficient, if not in general, at least for some classes of CTMCs.

Given that the process starts in state ℓ_1, let the path $\ell_1, \ell_2, \ldots, \ell_K$ be the most probable path to a state ℓ_K, with probability

$$\prod_{k=1}^{K-1} p_{\ell_k,\ell_{k+1}}, \tag{12}$$

where $p_{\ell_k,\ell_{k+1}} = \lambda_{\ell_k,\ell_{k+1}}/\lambda_{\ell_k}$ is a transition probability from state ℓ_k to state ℓ_{k+1} in the discrete embedded Markov chain (DTMC). We refer to an algorithm that chooses to include the edge state to which the most probable path (from state ℓ_1) leads as P.

The expected time duration spent in each of the path states $\ell_1, \ell_2, \ldots, \ell_{K-1}$ is

$$\frac{1}{\lambda_{\ell_1}}, \frac{1}{\lambda_{\ell_2}}, \ldots, \frac{1}{\lambda_{\ell_{K-1}}}. \tag{13}$$

Assuming the process follows this path, the expected time duration before reaching the state ℓ_K is

$$\sum_{k=1}^{K-1} \frac{1}{\lambda_{\ell_k}}. \tag{14}$$

We refer to an algorithm that chooses to include the edge state with the smallest expected time duration before the first hit as T.

In algorithms P and T, edge state choice is made based on a path probability or time duration. Both of these characterizations are limited when applied separately. For example, a choice based on a path probability neglects time spent on the path. We suggest weighting the expected time duration by the inverse of the path probability

$$\frac{\sum_{k=1}^{K-1} \frac{1}{\lambda_{\ell_k}}}{\prod_{k=1}^{K-1} p_{\ell_k,\ell_{k+1}}}, K = 2, 3, \ldots. \tag{15}$$

It can be shown that characteristic (15) of a path $\ell_1, \ell_2, \ldots, \ell_K$ can be efficiently obtained if it has already been computed for a shorter path $\ell_1, \ell_2, \ldots, \ell_{K-1}$. The expression (15) for a path of length $K-1$ can be written as follows

$$\frac{\sum_{k=1}^{K-2} \frac{1}{\lambda_{\ell_k}}}{\prod_{k=1}^{K-2} p_{\ell_k,\ell_{k+1}}} = \frac{\sum_{i=1}^{K-2} \prod_{k=1,k\neq i}^{K-2} \lambda_{\ell_k}}{\prod_{k=1}^{K-2} \lambda_{\ell_k,\ell_{k+1}}} = \frac{a^{(K-1)}}{b^{(K-1)}}, \tag{16}$$

where $a^{(K-1)}$ and $b^{(K-1)}$ represent the enumerator and denominator. By rewriting the expression (15) for a path of length K, we obtain the following

$$\frac{\sum_{k=1}^{K-1}\frac{1}{\lambda_{\ell_k}}}{\prod_{k=1}^{K-1}p_{\ell_k,\ell_{k+1}}}=\frac{\sum_{i=1}^{K-1}\prod_{k=1,k\neq i}^{K-1}\lambda_{\ell_k}}{\prod_{k=1}^{K-1}\lambda_{\ell_k,\ell_{k+1}}}=\frac{\lambda_{\ell_{K-1}}\sum_{i=1}^{K-2}\prod_{k=1,k\neq i}^{K-2}\lambda_{\ell_k}+\prod_{k=1,k\neq K-1}^{K-1}\lambda_{\ell_k}}{\prod_{k=1}^{K-1}\lambda_{\ell_k,\ell_{k+1}}}= \tag{17}$$
$$=\frac{\lambda_{\ell_{K-1}}a^{(K-1)}+\prod_{k=1}^{K-2}\lambda_{\ell_k}}{\lambda_{\ell_{K-1},\ell_K}b^{(K-1)}}=\frac{a^{(K)}}{b^{(K)}}.$$

Therefore, the characteristic (15) for a path of length K can be evaluated recursively by computing the values $a^{(K)}$ and $b^{(K)}$ by

$$\begin{aligned}a^{(k)}&=\lambda_{\ell_{k-1}}a^{(k-1)}+c^{k-1},\\ b^{(k)}&=\lambda_{\ell_{k-1},\ell_k}b^{(k-1)},\\ c^{(k)}&=\lambda_{\ell_{k-1}}c^{(k-1)}\ \text{ for } k=2,3,\ldots,K,\end{aligned} \tag{18}$$

starting with the initial values of $a^{(1)}=0$, $b^{(1)}=1$, and $c^{(1)}=1$. The algorithm that chooses to include the edge state with the smallest value of (15) is referred to as PT.

We summarize the shortest path-based algorithms by restating the general algorithm given in [13]. Let $dist(\boldsymbol{d}_\ell)$ be the distance from the initial state ℓ_1 to state ℓ, where $\boldsymbol{d}_\ell$ is a data structure (i.e., scalar, vector, etc.) associated with the state ℓ. Then, a partial CTMC can be constructed by choosing to include the edge states that are at the smallest distance from the initial state ℓ_1, as outlined in Algorithm 3.

Algorithm 3 Partial CTMC construction algorithm [13], based on state characterization by the shortest path. Input parameters: ℓ_1—an initial state, $K<|\boldsymbol{\mathcal{S}}|$—the size of the partial CTMC, g—CTMC generator.

function: partialCtmcDijkstra(ℓ_1,K,g)
  $\boldsymbol{\mathcal{P}}\leftarrow\{\ell_1\}$ ▷ The edge state set $\overline{\boldsymbol{\mathcal{P}}}$ is implied by (5).
  while $|\boldsymbol{\mathcal{P}}|<K$ **do**
    $\ell\leftarrow$ state in $\overline{\boldsymbol{\mathcal{P}}}$ with the smallest value of $dist(\boldsymbol{d}_\ell)$
    $\boldsymbol{\mathcal{P}}\leftarrow\boldsymbol{\mathcal{P}}\cup\{\ell\}$
    $\boldsymbol{\mathcal{T}}\leftarrow g(\ell)$
    for each ℓ_k **in** $\boldsymbol{\mathcal{T}}\setminus\boldsymbol{\mathcal{P}}$ **do**
      $\boldsymbol{d}^*_{\ell_k}\leftarrow extend(\ell,\ell_k)$
      if $dist\left(\boldsymbol{d}^*_{\ell_k}\right)<dist\left(\boldsymbol{d}_{\ell_k}\right)$ **then**
        $\boldsymbol{d}_{\ell_k}\leftarrow\boldsymbol{d}^*_{\ell_k}$
      end if
    end for
  end while
  return $\boldsymbol{\mathcal{P}}$
end of function

The distance data $\boldsymbol{d}^*_{\ell_k}$ of the path from state ℓ_1 through the included state ℓ to state $\ell_k\in g(\ell)$ is computed by a function $extend(\ell,\ell_k)$. The already known distance data $\boldsymbol{d}_{\ell_k}$ (of the shortest path) to state ℓ_k is replaced by $\boldsymbol{d}^*_{\ell_k}$, in case

$$dist\left(\boldsymbol{d}^*_{\ell_k}\right)<dist\left(\boldsymbol{d}_{\ell_k}\right). \tag{19}$$

The implementations of functions $dist(\ldots)$ and $extend(\ldots)$ depend on the specific algorithm and are given in Table 1. For a newly discovered state ℓ, its distance data $\boldsymbol{d}_\ell$ is initialized with the default value, as shown in Table 1.

Table 1. The implementation details of state distance data for algorithms D, P, T, and PT.

Alg.	Distance Data, d_ℓ	Initial Dist. Data	Default Dist. Data	Distance, $dist(d_\ell)$	Extension, $extend(\ell, \ell_k)$
D	(i_ℓ)	$i_{\ell_1} \leftarrow 0$	$i_\ell \leftarrow +\infty$	i_ℓ	$i^*_{\ell_k} \leftarrow i_\ell + 1$
P	(q_ℓ)	$q_{\ell_1} \leftarrow 1$	$q_\ell \leftarrow 0$	$\frac{1}{q_\ell}$	$q^*_{\ell_k} \leftarrow q_\ell p_{\ell,\ell_k}$
T	(d_ℓ)	$d_{\ell_1} \leftarrow 0$	$d_\ell \leftarrow +\infty$	d_ℓ	$d^*_{\ell_k} \leftarrow d_\ell + \frac{1}{\lambda_\ell}$
PT	(a_ℓ, b_ℓ, c_ℓ)	$a_{\ell_1} \leftarrow 0$ $b_{\ell_1} \leftarrow 1$ $c_{\ell_1} \leftarrow 1$	$a_\ell \leftarrow +\infty$ $b_\ell \leftarrow 1$ $c_\ell \leftarrow 0$	$\frac{a_\ell}{b_\ell}$	$a^*_{\ell_k} \leftarrow \lambda_\ell a_\ell + c_\ell$ $b^*_{\ell_k} \leftarrow \lambda_{\ell,\ell_k} b_\ell$ $c^*_{\ell_k} \leftarrow \lambda_\ell c_\ell$

The EXP algorithm computes exact edge state probabilities by considering all possible paths from state ℓ_1. The actual precise edge state probabilities are not that important as long as a state with the highest probability is selected; therefore, we suggest making a choice based on an approximate state probability that is obtained by considering paths from ℓ_1 of finite length. Such an approximation of edge state probabilities can be computed in the corresponding discretized Markov chain (DTMC), as shown in Algorithm 4, which we refer to as DSC.

Algorithm 4 A partial CTMC construction algorithm based on CTMC discretization. Input parameters: ℓ_1—an initial state, $K < |\boldsymbol{\mathcal{S}}|$—the size of the partial CTMC, g—CTMC generator, t—the specified time moment, m—the factor for the number of discretization steps.

function: partialCtmcDsc(ℓ_1, K, g, t, m)
  $\boldsymbol{\mathcal{P}} \leftarrow \{\ell_1\}$ ▷ The edge state set $\overline{\boldsymbol{\mathcal{P}}}$ is implied by (5).
  $\boldsymbol{\mathcal{L}} \leftarrow \emptyset$
  $i_{\ell_1} \leftarrow 0$
  $i_\ell \leftarrow 1$ **for** $\ell \in g(\ell_1)$ ▷ Initial distances to the edge states.
  while $|\boldsymbol{\mathcal{P}}| < K$ **do**
    $steps^{(1)} \leftarrow \lceil t \cdot \max\{\lambda_\ell \mid \ell \in \boldsymbol{\mathcal{P}}\} \rceil$
    $steps^{(2)} \leftarrow \max\{i_\ell | \ell \in \overline{\boldsymbol{\mathcal{P}}}\} \cdot m$
    $steps \leftarrow \max\left\{steps^{(1)}, steps^{(2)}\right\}$
    $\lambda^{(\mathrm{dsc})} \leftarrow steps/t$
    $D \leftarrow$ probability matrix of discretized (with rate $\lambda^{(\mathrm{dsc})}$) Markov chain $\boldsymbol{\mathcal{P}} \cup \overline{\boldsymbol{\mathcal{P}}}$
    $\left(p_{\ell_1}, p_{\ell_2}, \ldots, p_{\ell_{|\boldsymbol{\mathcal{P}}|+|\overline{\boldsymbol{\mathcal{P}}}|}}\right) \leftarrow D^{steps}\vec{e}_1$
    $\ell^{(\max)} \leftarrow$ state $\ell \in \overline{\boldsymbol{\mathcal{P}}}$ with the maximum value of p_ℓ
    $\boldsymbol{\mathcal{P}} \leftarrow \boldsymbol{\mathcal{P}} \cup \left\{\ell^{(\max)}\right\}$
    $\boldsymbol{\mathcal{L}} \leftarrow \boldsymbol{\mathcal{L}} \cup \left\{\ell^{(\max)}\right\}$
    while $|\boldsymbol{\mathcal{L}}| > 0$ **do** ▷ Update the values of i_ℓ.
      $\ell \leftarrow$ any removed state from $\boldsymbol{\mathcal{L}}$
      $\boldsymbol{\mathcal{T}} \leftarrow g(\ell)$
      for each ℓ_k in $\boldsymbol{\mathcal{T}}$ **do**
        $i^*_{\ell_k} \leftarrow i_\ell + 1$
        if $i^*_{\ell_k} < i_{\ell_k}$ **then**
          $i_{\ell_k} \leftarrow i^*_{\ell_k}$
          $\boldsymbol{\mathcal{L}} \leftarrow \boldsymbol{\mathcal{L}} \cup \{\ell_k\}$
        end if
      end for
    end while
  end while
  return $\boldsymbol{\mathcal{P}}$
end of function

The matrix $D = \{d_{i,j}\}$ of transition probabilities of the discretized DTMC (line 11 of Algorithm 4) has elements

$$d_{i,j} = \begin{cases} \dfrac{\lambda_{\ell_i,\ell_j}}{\lambda^{(\mathrm{dsc})}}, & i \neq j, \\ 1 - \dfrac{\lambda_{\ell_i}}{\lambda^{(\mathrm{dsc})}}, & i = j. \end{cases} \tag{20}$$

In the line 12 of Algorithm 4 $\vec{e}_1$ stands for vector of size $|\boldsymbol{\mathcal{P}}| + |\overline{\boldsymbol{\mathcal{P}}}|$ with the first element being one and the rest zeroes.

The advantage of the Dijkstra method (applied in Algorithm 3) is that the distance from the included state ℓ to every state $\ell_k \in g(\ell)$ needs to be updated only once. This is a consequence of choosing to include a state ℓ reachable by the shortest path. However, in Algorithm 4, where the distance from state ℓ_1 is measured in the minimum number of transitions, the edge state ℓ (i.e., with the highest value of p_ℓ) is not necessarily an edge state reachable by the shortest path (i.e., there might exist a state $\hat{\ell} \in \overline{\boldsymbol{\mathcal{P}}}$, such that $i_{\hat{\ell}} < i_\ell$). As a consequence, the distances from more than one state might need to be updated, which is realized in lines 16–26 of Algorithm 4.

In the following sections, we include the value of m in the algorithm name; for example, DSC2 stands for DSC algorithm with $m = 2$.

4. Examples

In this section, we provide several examples to elucidate the problem of partial CTMC construction. Example 1 clarifies the distinction between the terms 'edge state' and 'absorbing state'. We demonstrate that a heuristic may not construct an optimal partial CTMC in general, in Example 2. Example 3 shows partial CTMCs of various sizes constructed for the specified time moment. Partial CTMCs of fixed size are constructed while allowing the time moment to vary, in Example 4. We use the CTMC of Example 2 to show the first iteration of Algorithm 4 of the DSC method in Example 5.

In order to provide a more objective evaluation of the construction of partial CTMCs using the heuristic algorithms, we also identified the optimal partial CTMCs. For a small CTMC, it is not overly complicated to generate all possible partial CTMCs and select the one with the smallest absorbing probability $p^{(\mathrm{a})}(t)$. We refer to this brute-force algorithm [15] as GEN.

Example 1. *In a graphical depiction of CTMC states, we use the following color coding: white for transient or recurrent states, gray for edge and black for absorbing states. To illustrate the distinction between the edge and absorbing states, let us consider a full CTMC* $\boldsymbol{\mathcal{S}} = \{1,2,3,4\}$ *given in Figure 1a. We construct a partial CTMC starting from the initial state 1, thus having* $\boldsymbol{\mathcal{P}}^{(1)} = \{1\}$ *and* $\overline{\boldsymbol{\mathcal{P}}}^{(1)} = \{2,3\}$ *(Figure 1b). Next, we can choose some state* $\ell \in \overline{\boldsymbol{\mathcal{P}}}^{(1)}$ *to be included. Assuming we choose state 3, we obtain* $\boldsymbol{\mathcal{P}}^{(2)} = \{1,3\}$ *and* $\overline{\boldsymbol{\mathcal{P}}}^{(2)} = \{2,4\}$ *(Figure 1c). Only after the inclusion of state 2 is it considered as an absorbing state in the partial CTMC (Figure 1d).*

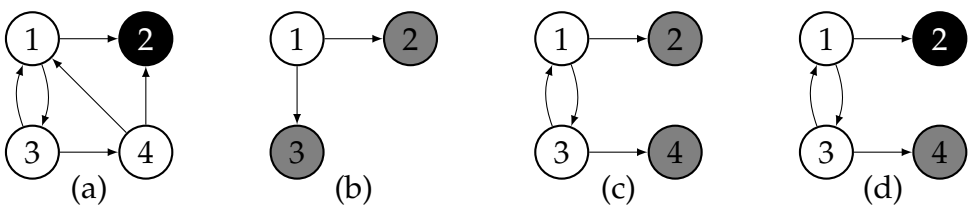

Figure 1. An example of partial CTMC construction: (**a**) the full CTMC, (**b**) the initial partial CTMC, (**c**,**d**) the partial CTMC after inclusion of states 3 and 2.

Example 2. *Starting from the initial state 1 (i.e., in CTMC* $\boldsymbol{\mathcal{P}} \cup \overline{\boldsymbol{\mathcal{P}}} = \{1,2,3\}$*), the probabilities of states 2 and 3 at time moment* $t = 1$ *are* 0.397 *and* 0.596*, respectively. The absorbing probablity* $p^{(\mathrm{a})}(t)$ *for two partial CTMCs* $\{1,2\}$ *and* $\{1,3\}$ *are* 0.813 *and* 0.929*. Out of all possible partial CTMCs of size 3, the one with the smallest value of* $p^{(\mathrm{a})}(t)$ *is* $\{1,3,4\}$ *(Figure 2d), which is also given by the EXP algorithm.*

Thus, in the construction of a partial CTMC of size 2, the choice of the EXP algorithm to include state 3 is incorrect, but this choice is correct for the construction of a partial CTMC of size 3.

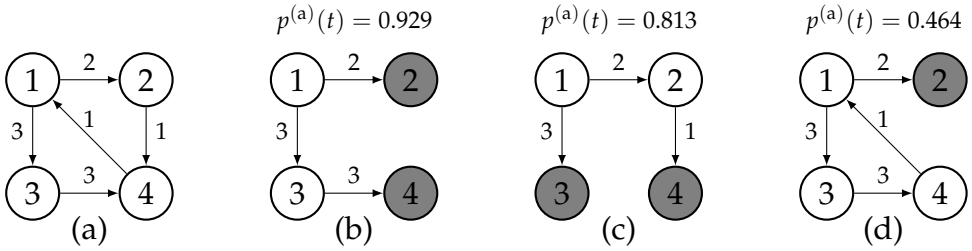

Figure 2. Example: (**a**) the full CTMC; (**b**) the constructed partial CTMC of size 2 by the EXP algorithm; (**c**,**d**) the partial CTMCs of sizes 2 and 3 with the smallest absorbing probability $p^{(\mathrm{a})}(1)$.

Example 2 can be used to prove the following statement. For a general CTMC $\boldsymbol{\mathcal{S}}$, there cannot exist a heuristic algorithm that follows Algorithm 2 and can construct an optimal partial CTMC of any size $K < |\boldsymbol{\mathcal{S}}|$. As shown in Example 2, the EXP algorithm fails to construct the optimal partial CTMC of size 2. If another heuristic algorithm would choose to include state 2, it would fail to construct the optimal partial CTMC of size 3.

This observation raises the following question. Let $\boldsymbol{\mathcal{P}}^{*(K)}$ be one of the optimal partial CTMCs of size K. What are the assumptions (about CTMC $\boldsymbol{\mathcal{S}}$ and the initial state ℓ_1) under which it could be proved that there exist the optimal partial CTMCs such that

$$\boldsymbol{\mathcal{P}}^{*(2)} \subset \boldsymbol{\mathcal{P}}^{*(3)} \subset \ldots \subset \boldsymbol{\mathcal{P}}^{*(|\boldsymbol{\mathcal{S}}|-1)} \; ? \tag{21}$$

Example 3. *Let us examine the CTMC provided in Figure 3.*

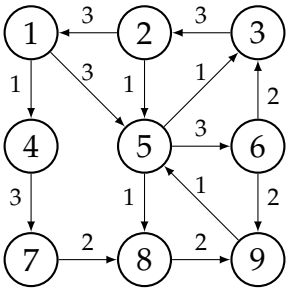

Figure 3. An example CTMC to demonstrate partial CTMC construction of sizes $2,3,\ldots,8$ for time moment $t = 1$.

Starting from the initial state 1, we construct partial CTMCs of sizes $2,3,\ldots,8$ *for time moment* $t = 1$*. The state choice sequences of heuristic algorithms are given in Table 2. Meanwhile, the optimal partial CTMC states need to be explicitly listed for each size of partial CTMC and are provided in Table 3.*

For instance, as shown in Table 2, the partial CTMC of size 5 constructed by the EXP algorithm is $\{1,5,6,3,4\}$*. Upon further analysis of the results provided in Table 2, we observe that algorithm pairs EXP, DSC and T, D produce quite similar state choice sequences. It appears that the heuristic algorithms (Tables 2 and 3) managed to construct an optimal partial CTMC only of size 2.*

For further comparison, we provide Table 4, where absorbing probabilities $p^{(a)}(t)$ are given for optimal partial CTMCs. Additionally, we identified the heuristic algorithms that constructed partial CTMCs with the smallest value of $p^{(a)}(t)$.

As indicated in Table 4, the optimal partial CTMCs (except for size 2) have smaller absorbing probabilities. If we set this fact aside, among the heuristic algorithms, there is no single one that constructed a partial CTMC with the smallest value of $p^{(a)}(t)$ for sizes $2, 3, \ldots, 8$. This is an evident consequence of the fact that optimal CTMCs do not satisfy the relation (21).

Table 2. The order of state choices when partial CTMC is constructed up to a size of 8 by the heuristic algorithms.

Alg.	State Choice Sequences
P	1, 5, 6, 4, 7, 8, 9, 3
T	1, 4, 5, 3, 8, 6, 7, 9
PT	1, 5, 4, 6, 7, 3, 8, 9
D	1, 4, 5, 7, 8, 6, 3, 9
EXP	1, 5, 6, 3, 4, 2, 7, 8
DSC1	1, 5, 6, 3, 2, 4, 7, 8

Table 3. The optimal partial CTMCs of sizes $2, 3, \ldots, 8$ as constructed by algorithm GEN.

Chain Size	Number of Chains	State Choice Sequences
2	2	1, 5
3	5	1, 4, 7
4	11	1, 5, 6, 9
5	16	1, 5, 6, 9, 8
6	16	1, 5, 6, 9, 3, 2
7	11	1, 5, 6, 9, 8, 3, 2
8	5	1, 5, 6, 9, 8, 3, 2, 4

Table 4. Comparison of absorbing probability $p^{(a)}(t)$ of optimal partial CTMCs and the smallest value of $p^{(a)}(t)$ as given by one (or several) heuristic algorithms.

Chain Size	GEN, Min $p^{(a)}(t)$	Other Alg., Min $p^{(a)}(t)$	Other Alg.
2	0.947	0.947	P, PT, EXP, DSC1
3	0.869	0.886	P, EXP, DSC1
4	0.731	0.799	EXP, DSC1
5	0.589	0.607	DSC1
6	0.426	0.565	EXP, DSC1
7	0.265	0.346	P
8	0.223	0.244	P, T, PT, D

Example 4. *To assess the performance of the EXP algorithm in comparison with the brute-force algorithm GEN, we generated partial CTMCs of size 5 for the various time moments. These partial CTMCs were constructed for the CTMC illustrated in Figure 4, starting from the initial state 2.*

We numerically determined approximate time intervals (up to time moment $t = 2$) where the same (optimal) partial CTMCs were identified (Table 5).

The absorbing probability $p^{(a)}(t)$ of each partial CTMC, obtained by algorithm GEN, is plotted in Figure 5, in red. It is noteworthy that the smallest value of absorbing probability $p^{(a)}(t)$ is obtained by different optimal partial CTMCs. The same can be said about the partial CTMCs obtained by the EXP algorithm (Figure 6).

As illustrated in Figure 7, the EXP algorithm fails to construct an optimal partial CTMC starting from time moment $t \approx 0.78$, which is an expected outcome.

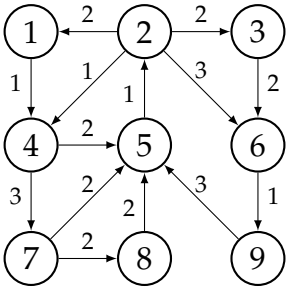

Figure 4. An example CTMC to demonstrate the importance of time moment t for the construction of partial CTMC with the aim to minimize absorbing probability $p^{(\mathrm{a})}(t)$.

Table 5. Time intervals (up to $t = 2$) of partial CTMCs as identified by algorithms GEN and EXP (the states are ordered).

Alg.	Time Interval	States, $\mathcal{P}$
GEN	$(0.000, 0.368)$	1, 2, 3, 4, 6
	$(0.368, 0.780)$	1, 2, 3, 6, 9
	$(0.780, 2.000)$	2, 3, 5, 6, 9
EXP	$(0.000, 0.368)$	1, 2, 3, 4, 6
	$(0.368, 1.260)$	1, 2, 3, 6, 9
	$(1.260, 2.000)$	1, 2, 5, 6, 9

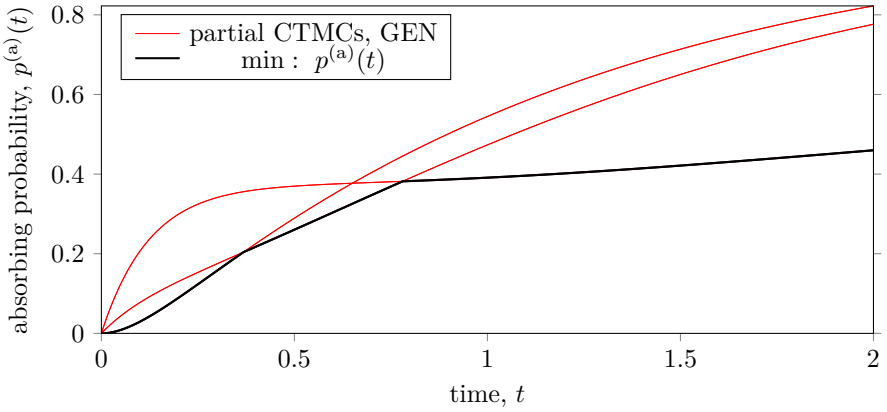

Figure 5. The plots of absorbing probability $p^{(\mathrm{a})}(t)$ for partial CTMCs constructed by algorithm GEN. Each of these partial CTMCs is optimal in their specific time interval (given in Table 5).

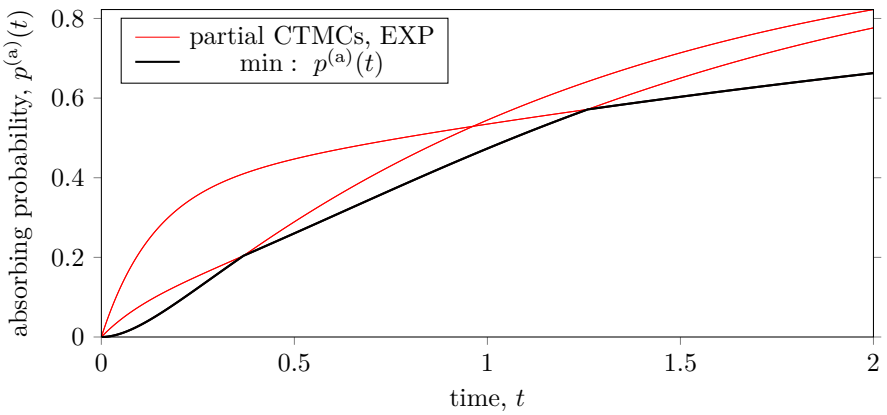

Figure 6. The plots of absorbing probability $p^{(\mathrm{a})}(t)$ of partial CTMCs constructed by the EXP algorithm. For each time interval specified in Table 5, one of the partial CTMCs gives the smallest value of $p^{(\mathrm{a})}(t)$.

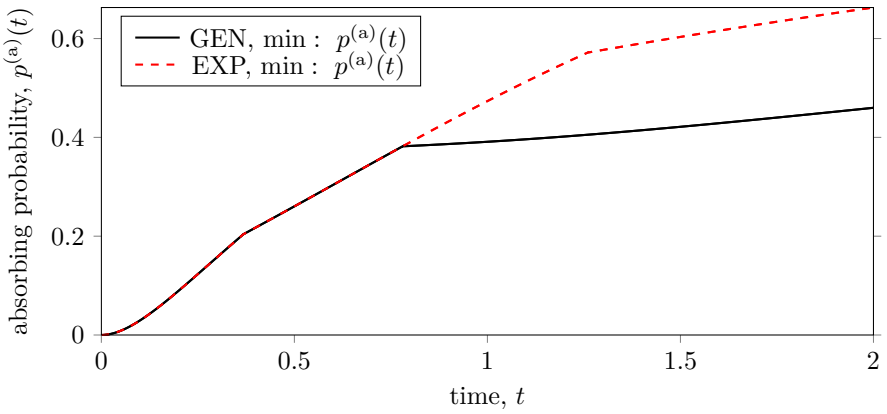

Figure 7. Comparison of $p^{(a)}(t)$ of partial CTMCs constructed by algorithms GEN and EXP.

Example 5. *By using the CTMC given in Example 3 we show one iteration of the DSC algorithm (Algorithm 4) for time moment $t = 1$ and starting from the initial state 1.*

As show in Figure 8a, before the first iteration we have $\boldsymbol{\mathcal{P}} = \{1\}$, $\overline{\boldsymbol{\mathcal{P}}} = \{4,5\}$ and the distances (measured in the number of transitions) initialized for the edge states are $i_4 = 1$, $i_5 = 1$. The rate matrix of CTMC $\boldsymbol{\mathcal{P}} \cup \overline{\boldsymbol{\mathcal{P}}}$ is

$$Q = \begin{pmatrix} -4 & 1 & 3 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}. \tag{22}$$

Next, we identify the minimum number (assuming $m = 1$) of steps: 4, and therefore the discratization rate is $\lambda^{(\mathrm{dsc})} = steps/t = 4$. Then, we obtain the probability matrix D of DTMC according to (20) and compute approximate state probabilities as $D^4\vec{e}_1$

$$\begin{pmatrix} p_1(t) \\ p_4(t) \\ p_5(t) \end{pmatrix} \approx \begin{pmatrix} 0 & 0.25 & 0.75 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}^4 \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 0.25 \\ 0.75 \end{pmatrix}; \quad \begin{pmatrix} p_1(t) \\ p_4(t) \\ p_5(t) \end{pmatrix} = \begin{pmatrix} 0.018316 \\ 0.245421 \\ 0.736263 \end{pmatrix}.$$

As a result, the state 5 is chosen to be included in $\boldsymbol{\mathcal{P}}$ (Figure 8b). For comparison, we have computed the exact state probabilities by (7). Before starting the next iteration, we need to update the distances to the edge states, which happen to be $i_3 = 2$, $i_4 = 1$, $i_6 = 2$, $i_8 = 2$. The rest of the state choices are shown in the last row of Table 2.

The distances to the edge states need to be maintained in order to assure that the calculated number of steps is sufficient to obtain the non-zero edge state approximate probabilities.

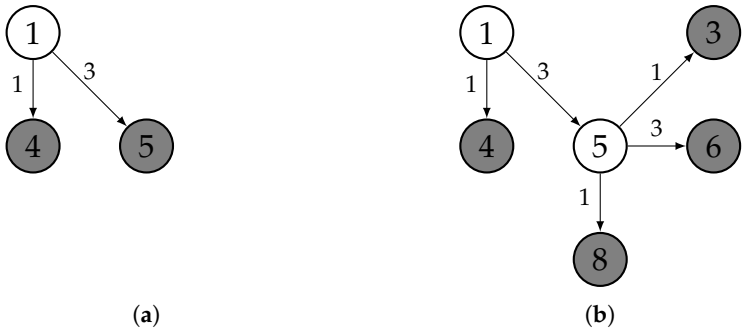

Figure 8. Example: (**a**) the partial CTMC before the first iteration; (**b**) the partial CTMC after the first iteration.

5. Results

We conducted performance comparisons of heuristic algorithms on two distinct models. The first model represents a car sharing system, constructed using the second modeling approach detailed in [20]. In this model, travel durations are simulated using randomly generated phase-type distributions of size 2, the client arrival rates and the routing probabilities are randomly selected. For the second model, we generated a random strongly connected CTMC using an algorithm given in [21].

In the initial part of the experimentation, we constructed partial CTMCs starting from each state (for the specified time moment t). The average and standard deviation of absorbing probabilities $p^{(a)}(t)$ are provided in Tables 6 and 7. Next, we compared the relative performance of heuristic algorithms based on rankings. Specifically, for each initial state, we constructed partial CTMCs (in our case, 10 of them) using each algorithm and ranked them according to the obtained absorbing probability $p^{(a)}(t)$. The characteristics of these ranks are provided in Tables 8 and 9. The same methodology was applied to compare algorithm runtimes, with the results provided in Tables 10 and 11.

Table 6. Characteristics of the absorbing probability of partial CTMCs of size 500 constructed for a car sharing system model with 3 zones and 3 cars. The CTMC of the model has 1771 states and 12,474 transitions (the average transition rate is 0.4532).

Alg.	Time Moment $t = 1.0$		Time Moment $t = 2.0$		Time Moment $t = 3.0$	
	Mean	Std.	Mean	Std.	Mean	Std.
RND	0.35949	3.3×10^{-3}	0.59513	3.9×10^{-3}	0.73980	3.7×10^{-3}
SRND	0.00177	5.8×10^{-5}	0.02522	5.9×10^{-4}	0.08921	1.7×10^{-3}
P	0.00080	2.3×10^{-5}	0.00996	2.3×10^{-4}	0.03532	7.3×10^{-4}
T	0.00468	1.5×10^{-4}	0.04583	9.3×10^{-4}	0.13933	2.2×10^{-3}
PT	0.00065	2.0×10^{-5}	0.00952	2.3×10^{-4}	0.03612	7.8×10^{-4}
D	0.00268	8.7×10^{-5}	0.03335	7.7×10^{-4}	0.10935	2.0×10^{-3}
DSC1	0.00056	1.8×10^{-5}	0.00799	1.9×10^{-4}	0.02762	5.8×10^{-4}
DSC2	0.00049	1.6×10^{-5}	0.00754	1.9×10^{-4}	0.02730	5.8×10^{-4}
DSC4	0.00048	1.6×10^{-5}	0.00741	1.9×10^{-4}	0.02695	5.7×10^{-4}
EXP	0.00048	1.6×10^{-5}	0.00737	1.8×10^{-4}	0.02681	5.7×10^{-4}

Table 7. Characteristics of absorbing probability of partial CTMCs of size 500 constructed for a randomly generated CTMC with 1771 states and 12,474 transitions (average transition rate is ≈ 0.4999).

Alg.	Time Moment $t = 1.0$		Time Moment $t = 2.0$		Time Moment $t = 3.0$	
	Mean	Std.	Mean	Std.	Mean	Std.
RND	0.64704	2.8×10^{-3}	0.88974	1.9×10^{-3}	0.96114	1.2×10^{-3}
SRND	0.29933	2.2×10^{-3}	0.71929	2.4×10^{-3}	0.89558	1.6×10^{-3}
P	0.24178	2.0×10^{-3}	0.63515	2.5×10^{-3}	0.84265	1.8×10^{-3}
T	0.39175	2.6×10^{-3}	0.79468	2.3×10^{-3}	0.93183	1.4×10^{-3}
PT	0.23662	2.0×10^{-3}	0.64085	2.6×10^{-3}	0.85005	1.9×10^{-3}
D	0.30108	2.1×10^{-3}	0.72040	2.4×10^{-3}	0.89593	1.6×10^{-3}
DSC1	0.22856	1.9×10^{-3}	0.61906	2.5×10^{-3}	0.82786	1.8×10^{-3}
DSC2	0.22850	1.9×10^{-3}	0.61906	2.5×10^{-3}	0.82786	1.8×10^{-3}
DSC4	0.22819	1.9×10^{-3}	0.61908	2.5×10^{-3}	0.82786	1.8×10^{-3}
EXP	0.22799	1.9×10^{-3}	0.61917	2.5×10^{-3}	0.82810	1.8×10^{-3}

Table 8. Characteristics of algorithm ranks based on the absorbing probability $p^{(a)}(t)$ of the constructed partial CTMCs for a car sharing system model.

Alg.	Time Moment $t = 1.0$		Time Moment $t = 2.0$		Time Moment $t = 3.0$	
	Mean	Std.	Mean	Std.	Mean	Std.
RND	9.99887	8.0×10^{-4}	9.99831	9.8×10^{-4}	9.90344	7.1×10^{-3}
SRND	7.11858	1.6×10^{-2}	7.27216	1.3×10^{-2}	7.19593	1.5×10^{-2}
P	6.15359	1.3×10^{-2}	5.77527	1.3×10^{-2}	5.40147	1.3×10^{-2}
T	8.63919	1.6×10^{-2}	8.65500	1.4×10^{-2}	8.56578	1.6×10^{-2}
PT	4.94410	9.3×10^{-3}	5.05816	1.1×10^{-2}	5.40542	1.4×10^{-2}
D	8.02823	1.7×10^{-2}	8.05364	1.6×10^{-2}	7.94692	1.7×10^{-2}
DSC1	4.10954	8.7×10^{-3}	4.17504	1.2×10^{-2}	3.82665	1.0×10^{-2}
DSC2	2.99435	2.1×10^{-3}	2.98306	3.6×10^{-3}	3.00000	8.4×10^{-3}
DSC4	1.97177	4.2×10^{-3}	1.92151	6.9×10^{-3}	1.91078	9.8×10^{-3}
EXP	1.03162	4.3×10^{-3}	1.09373	7.5×10^{-3}	1.15415	9.9×10^{-3}

Table 9. Characteristics of algorithm ranks based on the absorbing probability $p^{(a)}(t)$ of the constructed partial CTMCs for a randomly generated CTMC.

Alg.	Time Moment $t = 1.0$		Time Moment $t = 2.0$		Time Moment $t = 3.0$	
	Mean	Std.	Mean	Std.	Mean	Std.
RND	9.43986	1.2×10^{-2}	8.52851	1.2×10^{-2}	7.92547	6.8×10^{-3}
SRND	6.88876	1.7×10^{-2}	6.05082	1.8×10^{-2}	5.51214	1.3×10^{-2}
P	5.41841	1.3×10^{-2}	3.61491	1.4×10^{-2}	2.97798	4.7×10^{-3}
T	8.43535	1.2×10^{-2}	7.53811	1.3×10^{-2}	7.01920	6.8×10^{-3}
PT	4.46358	1.2×10^{-2}	4.45680	1.3×10^{-2}	3.96894	4.8×10^{-3}
D	7.00000	1.7×10^{-2}	6.02597	1.7×10^{-2}	5.43704	1.3×10^{-2}
DSC1	3.16996	1.8×10^{-2}	1.69509	1.7×10^{-2}	1.24449	1.0×10^{-2}
DSC2	2.90288	1.4×10^{-2}	1.69509	1.7×10^{-2}	1.24449	1.0×10^{-2}
DSC4	1.92264	1.3×10^{-2}	1.72050	1.6×10^{-2}	1.24675	1.0×10^{-2}
EXP	1.30265	1.5×10^{-2}	1.86392	1.9×10^{-2}	1.72953	1.1×10^{-2}

Table 10. Characteristics of partial CTMC construction runtimes and their ranks for the car sharing system model.

Alg.	Runtime (in Seconds)		Runtime Rank	
	Mean	Std.	Mean	Std.
RND	0.02154	1.1×10^{-5}	5.38227	2.0×10^{-2}
SRND	0.01618	1.4×10^{-5}	1.13665	1.0×10^{-2}
P	0.02080	2.3×10^{-5}	4.98363	1.9×10^{-2}
T	0.01770	1.5×10^{-5}	2.78148	1.6×10^{-2}
PT	0.02049	2.2×10^{-5}	4.48899	1.9×10^{-2}
D	0.01727	1.7×10^{-5}	2.22699	1.6×10^{-2}
DSC1	0.32313	6.4×10^{-4}	7.00000	0.0
DSC2	0.35770	5.5×10^{-4}	8.00000	0.0
DSC4	0.44843	7.1×10^{-4}	9.00000	0.0
EXP	0.58833	1.7×10^{-3}	10.00000	0.0

Table 11. Characteristics of partial CTMC construction runtimes and their ranks for the randomly generated CTMC.

Alg.	Runtime (in Seconds)		Runtime Rank	
	Mean	Std.	Mean	Std.
RND	0.01020	9.3×10^{-6}	2.60305	2.5×10^{-2}
SRND	0.00845	6.6×10^{-6}	1.04856	7.9×10^{-3}
P	0.01139	8.4×10^{-6}	5.12705	2.4×10^{-2}
T	0.01117	1.1×10^{-5}	4.63185	2.4×10^{-2}
PT	0.01107	9.0×10^{-6}	4.42292	2.4×10^{-2}
D	0.01050	1.1×10^{-5}	3.16657	2.9×10^{-2}
DSC1	0.48800	8.3×10^{-4}	7.58046	1.2×10^{-2}
DSC2	0.48651	8.0×10^{-4}	7.42236	1.2×10^{-2}
DSC4	0.52076	4.1×10^{-4}	8.99718	1.3×10^{-3}
EXP	0.97456	2.5×10^{-3}	10.00000	0.0

As demonstrated in Tables 6 and 8 the EXP algorithm, even though the slowest, consistently constructed partial CTMCs with the smallest values of absorbing probability from nearly every initial state. Conversely, the fastest algorithm, SRND, constructed CTMCs that ranked around the seventh place on average. Algorithm PT strikes a balance between partial CTMC quality and construction runtime.

It is worth noting that one reason algorithms EXP and DSC are slower is due to the fact that edge state probabilities are recomputed from scratch to determine the next edge state to be included. In contrast, algorithms, based on the shortest path reuse or update the known distance data of edges states.

The main algorithm performance trends observed in the case of the car sharing system model persist for a randomly generated CTMC.

For the second part of numerical experimentation, we opted for the fastest algorithms to construct a larger partial CTMC comprising 200,000 states. The partial CTMC was constructed from a single initial state. As shown in Tables 12 and 13, we state the time it took to compute absorbing probability, also.

Table 12. Characteristics of absorbing probabilities of partial CTMCs of size 200,000 constructed by the heuristic algorithms SRND, P, and PT for a car sharing system model with 10 zones and 100 cars. The partial CTMCs are constructed starting from the state, which represent 10 cars in the first zone. The full CTMC of the system has $\binom{309}{100} \approx 1.49 \times 10^{83}$ states.

Alg.	Absorbing Probability, $p^{(a)}(t)$ at:			Construction Runtime (in Seconds)	Evaluation Runtime (in Seconds)
	$t = 1.0$	$t = 2.0$	$t = 3.0$		
SRND	0.00201	0.03831	0.14732	70.60064	6.52133
P	0.00062	0.01435	0.06765	5597.13894	16.43587
PT	0.00057	0.01406	0.06764	5234.25113	15.73841

Table 13. Characteristics of absorbing probabilities of partial CTMCs of size 200,000 constructed by the heuristic algorithms SRND, P, and PT for a randomly generated CTMC of size 1,000,000.

Alg.	Absorbing Probability, $p^{(a)}(t)$ at:			Construction Runtime (in Seconds)	Evaluation Runtime (in Seconds)
	$t = 1.0$	$t = 2.0$	$t = 3.0$		
SRND	0.16594	0.63560	0.86841	12.71308	3.91580
P	0.10415	0.48092	0.75997	691.63537	3.47252
PT	0.09552	0.48086	0.76745	688.24140	3.45486

Tables 12 and 13 show that the performance of algorithms P and PT is quite comparable. The algorithm SRND stands out for its significantly smaller runtime. The runtime is affected by the number of edge states present at the moment the choice to include an edge state is made. To gain some insight, we plotted the number of edge states $\nu(i)$ observed after including the ith state in Figures 9 and 10.

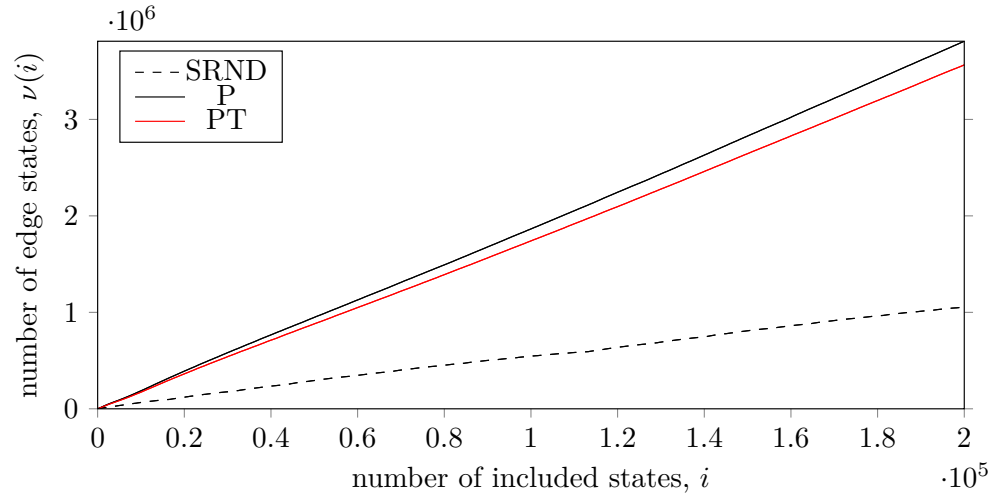

Figure 9. Number of edge states observed in the case of the car sharing system model.

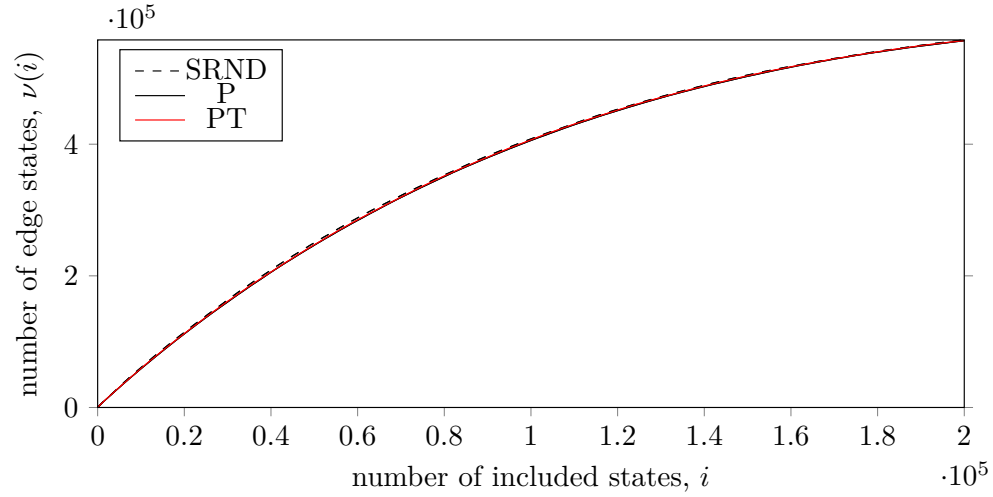

Figure 10. Number of edge states observed in the case of randomly generated CTMC.

As seen in Figure 9, the number of edge states increases steadily, while in case of the randomly generated CTMC (Figure 10), the number of edge states increase at a diminishing rate. It is also observed that the state choices may (Figure 9) or may not (Figure 10) have a strong influence on the number of edge states. Further, we quantify the tendencies discussed in the above figures. The mean of newly introduced edge states (after inclusion of ℓ_i state) and the mean and maximum of transitions from the included state ℓ_i are given in Tables 14 and 15.

Table 14. The characteristics explaining the change in the number of edge states in the case of the car sharing system model.

Alg.	Mean $\Delta(i)$	Mean $\tau(i)$	Max. $\tau(i)$
SRND	6.270	30.896	64
P	20.051	41.436	102
PT	18.824	40.360	102

Table 15. The characteristics explaining the change in the number of edge states in case of randomly generated CTMC.

Alg.	Mean $\Delta(i)$	Mean $\tau(i)$	Max. $\tau(i)$
SRND	3.792	7.029	28
P	3.786	7.001	28
PT	3.787	7.006	28

The characteristics given in Tables 14 and 15 support the previously discussed differences in partial CTMC construction for these two models. In addition, we can see that about half of the transitions from the included state are to previously discovered states, on average.

6. Discussion

6.1. Complexity

The complexity of Algorithm 2 depends on a number of factors. First, it depends on the properties of the underlying CTMC. Assuming that CTMC $\boldsymbol{\mathcal{S}}$ (of finite or infinite size) is such that the number of transitions from any state $\vec{s} \in \boldsymbol{\mathcal{S}}$ is bounded, i.e., there exists a positive integer M that

$$|g(\vec{s})| \leq M, \tag{23}$$

the number of edge states increase linearly (i.e., $\nu(i) \leq Mi$, for $i = 1, 2, \ldots, K$), at most. This assumption implies that complexity of CTMC generator g is $\mathcal{O}(1)$, which corresponds to a finite number of possible events in the model description by the event formalism. Second, we need to condier the complexity of choosing the ith state from $\nu(i)$ edge states. In the case of the first group of algorithms (RND, SRND), the state choice does not require much computation resulting in overall complexity of $\mathcal{O}(K)$. As for the second group of algorithms (P, PT, D, T), based on the edge state characterization by the shortest path, assuming that identification of edge state at the shortest distance is backed by a priority queue (in Algorithm 3), the worst case complexity is $\mathcal{O}(MK^2 \log(MK)) = \mathcal{O}(K^2 \log(K))$. In the case of the third group of algorithms (EXP, DSC), the computation of edge state exact/approximate probabilities involves $\phi(i)$ matrix-vector multiplications for each iteration i, where matrix size is

$$\left|\boldsymbol{\mathcal{P}}^{(i)}\right| + \left|\overline{\boldsymbol{\mathcal{P}}}^{(i)}\right| = i + \nu(i) \leq K + MK. \tag{24}$$

If in the EXP algorithm, edge state probabilities are computed by the uniformization [8] method, the number of matrix-vector multiplications is determined by the stiffness [22] of CTMC. In the case of the DSC algorithm (assuming $m = 1$), the number of matrix-vector multiplications is given by the highest shortest distance (measured in the number of transitions) from the initial state to an edge state. Without trying to go much into the details, we assume that $\phi(i)$ is bounded by a positive integer L (i.e., $\phi(i) \leq L$, for $i = 1, 2, \ldots, K$). If the underlying CTMC is sparse (i.e., (23) holds and for a finite CTMC we have $M << |\boldsymbol{\mathcal{S}}|$, as well), the complexity of edge state probability computation is $\mathcal{O}(L(K + MK)) = \mathcal{O}(K^2)$. The edge state probabilities are recomputed before choosing an edge state to include, resulting in an overal worst case complexity of $\mathcal{O}(K^3)$. Without the assumption (23), the order of complexity increases by several orders, depending on the group of algoritms.

As our numerical results suggest, the shortest path based algorithms can be very efficient, considering that edge state characterization is reused through iterations and sometimes updated, if a shorter path is found. The naive algorithm SRND can be still

considered, due to its implementation simplicity, especially if there are resources for constructing and evaluating a larger partial CTMC.

6.2. Limitations and Remarks

The biggest limitation of these heuristic algorithms is that they cannot find an optimal partial CTMC (as we showed that in Example 2), in general. On the other hand, if a structure of CTMC is simple enough, one might derive a much more efficient algorithm to identify (even optimal) partial CTMC.

We showed by Example 4 that optimal partial CTMC depends on the time moment t. Thus, we might expect a good heuristic algorithm to consider the time moment t in some way, but that is carried out only by the high computational complexity algorithms EXP and DSC. These algorithms could be considered for practical application if edge state probabilities could be evaluated with smaller computational effort (i.e., by reusing intermediate results from the previous iterations).

The heuristic algorithm template (i.e., Algorithm 2) implies that to identify a good partial CTMC of size K, one of size $K-1$ needs to be identified. As a result, the heuristic algorithms try to solve a harder problem than necessary, i.e., not only which states need to be included, but when, as well. One way to overcome this formulation issue would be to choose to include a particular subset of edge states in each iteration.

7. Conclusions

We explored three groups of heuristic algorithms for the construction of partial CTMC with a desired small absorbing probability. From the first group of naive algorithms (RND, SRND), SRND stands out for its simplicity and speed. Among the algorithms in the second group (P, T, PT, and D), which are based on characterization by the shortest path, both P and PT constructed partial CTMCs of even smaller absorbing probabilities at the expense of increased runtime. On the other hand, the third group of algorithms (DSC, EXP), which are based on characterization by all paths of finite or infinite length, constructed partial CTMCs with even smaller values of absorbing probabilities compared to P and PT, albeit at a significantly higher runtime cost. In practical applications, the algorithms SRND, P, and PT could prove to be usefull, especially if the structure of underlying CTMC is not well researched and more advanced transient analysis methods are not yet available.

Author Contributions: Conceptualization, T.R. and E.V.; methodology, E.V.; software, M.B.; validation, M.B. and E.V.; formal analysis, E.V.; investigation, M.B.; writing—original draft preparation, M.B.; writing—review and editing, T.R. and E.V.; visualization, M.B.; supervision, T.R. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Data Availability Statement: No new data were created or analyzed in this study. Data sharing is not applicable to this article.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Bhattacharya, R.N.; Waymire, E.C. *Stochastic Processes with Applications*; John Wiley & Sons: Hoboken, NJ, USA, 1990.
2. Norris, J.R. *Markov Chains*; Cambridge Series in Statistical and Probabilistic Mathematics; Cambridge University Press: Cambridge, UK, 1998; pp. I–XVI, 1–237.
3. Neuts, M.F. *Matrix-Geometric Solutions in Stochastic Models: An Algorithmic Approach*; Courier Corporation: North Chelmsford, MA, USA, 1994.

4. Bobbio, A.; Trivedi, K. An Aggregation Technique for the Transient Analysis of Stiff Markov Chains. *IEEE Trans. Comput.* **1986**, *C-35*, 803–814. [CrossRef]
5. Buchholz, P.; Sanders, W. Approximate computation of transient results for large Markov chains. In Proceedings of the First International Conference on the Quantitative Evaluation of Systems, 2004. QEST 2004. Proceedings, Enschede, The Netherlands, 27–30 September 2004; pp. 126–135. [CrossRef]
6. Bazan, P.; German, R. Approximate analysis of stochastic models by self-correcting aggregation. In Proceedings of the Second International Conference on the Quantitative Evaluation of Systems (QEST'05), Turin, Italy, 19–22 September 2005; pp. 134–143. [CrossRef]
7. Bazan, P.; German, R. Approximate transient analysis of large stochastic models with WinPEPSY-QNS. *Comput. Netw.* **2009**, *53*, 1289–1301. [CrossRef]
8. Stewart, W.J. *Introduction to the Numerical Solution of Markov Chains*; Princeton University Press: Princeton, NJ, USA, 1994.
9. van Moorsel, A.P.A.; Sanders, W.H. Adaptive uniformization. *Commun. Stat. Stoch. Model.* **1994**, *10*, 619–647. [CrossRef]
10. van Moorsel, A.P.A.; Sanders, W.H. Transient solution of Markov models by combining adaptive and standard uniformization. *IEEE Trans. Reliab.* **1997**, *46*, 430–440. [CrossRef]
11. Sidje, R.B.; Burrage, K.; MacNamara, S. Inexact Uniformization Method for Computing Transient Distributions of Markov Chains. *SIAM J. Sci. Comput.* **2007**, *29*, 2562–2580. [CrossRef]
12. Andreychenko, A.; Sandmann, W.; Wolf, V. Approximate adaptive uniformization of continuous-time Markov chains. *Appl. Math. Model.* **2018**, *61*, 561–576. [CrossRef]
13. Brameret, P.; Rauzy, A.; Roussel, J. Automated generation of partial Markov chain from high level descriptions. *Reliab. Eng. Syst. Saf.* **2015**, *139*, 179–187. [CrossRef]
14. Dijkstra, E.W. A note on two problems in connexion with graphs. *Numer. Math.* **1959**, *1*, 269–271. [CrossRef]
15. Bražėnas, M.; Valakevičius, E. Investigation of continuous time Markov chain approximate transient analysis via subgraph generation in the case of a car sharing system model. *Expert Syst. Appl.* **2024**, *237*, 121595. [CrossRef]
16. Rubino, G. Transient analysis of Markovian queueing systems: A survey with focus on closed-forms and uniformization. In *Advanced Trends in Queueing Theory*; Anisimov, V., Limnios, N., Eds.; ISTE @ Wiley: London, UK, 2020; Volume 2, pp. 1–35.
17. Vishnevsky, V.M.; Vytovtov, K.; Barabanova, E. Transient Behavior of a Two-Phase Queuing System with a Limitation on the Total Queue Size. *Autom. Remote Control.* **2024**, *85*, 46–59. [CrossRef]
18. Bu, Q. Transient Analysis for a Queuing System with Impatient Customers and Its Applications to the Pricing Strategy of a Video Website. *Mathematics* **2024**, *12*, 2030. [CrossRef]
19. AbuSalim, S.W.; Ibrahim, R.; Saringat, M.Z.; Jamel, S.; Wahab, J.A. Comparative Analysis between Dijkstra and Bellman-Ford Algorithms in Shortest Path Optimization. *IOP Conf. Ser. Mater. Sci. Eng.* **2020**, *917*, 012077. [CrossRef]
20. Bražėnas, M.; Valakevičius, E. Approximation of Non-Markovian Car Sharing Systems Models by Markovian One. In Proceedings of the Intelligent Systems and Applications, Amsterdam, The Netherlands, 1–2 September 2022; Arai, K., Ed.; Springer: Cham, Switzerland, 2023; pp. 458–474.
21. Maurer, P.M. Generating Strongly Connected Random Graphs. In Proceedings of the 2017 International Conference on Modeling, Simulation and Visualization Methods (MSV'17), London, UK, 11–14 July 2017; Arabnia, H.R., Deligiannidis, L., Tinetti, F.G., Eds.; CSREA: Las Vegas, NV, USA, 2017; pp. 3–6.
22. Trivedi, A.R.K. Numerical transient analysis of markov models. *Comput. Oper. Res.* **1988**, *15*, 19–36. [CrossRef]