State-Space Size Estimation By Least-Squares Fitting

Nicholas J. Dingle William J. Knottenbelt [*]

Abstract

We present a method for estimating the number of states in the continuous time Markov chains (CTMCs) underlying high-level models using least-squares fitting. Our work improves on existing techniques by producing a numerical estimate of the number of states rather than classifying the state space into on of three types. We demonstrate the practicality and accuracy of our approach on a number of CTMCs generated from three Generalised Stochastic Petri Net (GSPN) models with up to 11 million states.

1 Introduction

A vital component of many correctness and performance analysis techniques is the explicit enumeration of the state space underlying a high-level model such as a Petri net or a process algebra specification. An early indication of likely state space size is beneficial in a number of ways. For example, it provides the user with a good idea of the computational resources (CPU time, number of CPUs, memory, disk space etc.) that will be required to complete the analysis process. If the estimate suggests that currently allocated resources are inadequate, the process can be restarted early with increased resources The latter is particularly useful in utility computing environments where charges are levied on a CPU hour basis. Alternatively, a large estimate may persuade the modeller to apply an alternative exploration or analysis strategy (e.g. using probabilistic methods [9, 11], or "on-the-fly" [2] approaches) or to revisit the level of abstraction employed in the model.

However, with the exception of specialised models with restricted structure [12], there are few known techniques for estimating state space sizes from high level models in the literature. An important recent work, and the inspiration for our present investigation, is the paper of Pelánek and Šimeček [10] in which various methods for estimating state-space sizes are discussed. The methods fall into two main categories: those based on state sampling and those based on the attributes of breadth-first state-space exploration.

The sampling-based approach works by taking two samples of the state space, each containing s states, and comparing the states in each. Some number, x, will appear in both and the ratio x/s is then used to classify the size of

[*]Department of Computing, Imperial College London, 180 Queen's Gate, London SW7 2BZ, United Kingdom. Email: {njd200,wjk}@doc.ic.ac.uk

```
  begin
      $A = \emptyset$
      $E = s_0$
      $F$.insert($s_0$)
      while $F$(not empty) do begin
          $F$.remove($s$)
          foreach $s' \in$ succ($s$) do begin
              if $s' \notin E$ do begin
                  $F$.insert($s'$)
                  $E = E \cup s'$
              end
              $A = A \cup \mathrm{id}(s) \rightarrow \mathrm{id}(s')$
          end
      end
end
```

Figure 1: Breadth-first search algorithm for state-space exploration [8].

the total state space into one of three categories. The samples can be generated by breadth-first search, depth-first search or a random walk.

The estimation from the attributes of breadth-first search also aims to classify the overall state-space size into the same three categories based the number of states in the first k levels of the search. Estimation by human judgment, by classification trees and by neural networks was conducted, as well as an investigation into combining these classification methods with the results from the random-sampling approach to improve accuracy.

The major limitation of [10] is that the authors explicitly avoid estimating the total number of states and instead confine themselves to placing the estimated total state-space size into one of three categories (i.e. those models which can be handled easily, those which may require state-space reduction or parallel generation and those which are too large). In contrast, this paper presents a method for dynamically estimating the number of states in the underlying state-space of a high-level model. Our method uses least-squares fitting from the number of states currently observed during the breadth-first state generation process. We demonstrate the accuracy of this technique on a number of state spaces generated from three high-level Generalised Stochastic Petri Net (GSPN) models.

The remainder of this paper is organised as follows: Section 2 briefly presents the breadth-first state-space generation algorithm used the DNAmaca [7] steady-state analysis tool, before Section 3 describes the method of least-squares fitting. Section 4 then introduces the three GSPN models considered and demonstrates how the fitting method can be used to predict accurately the total number of reachable states whilst the state-generation process is underway. Finally, Section 5 concludes and suggests directions for future work.

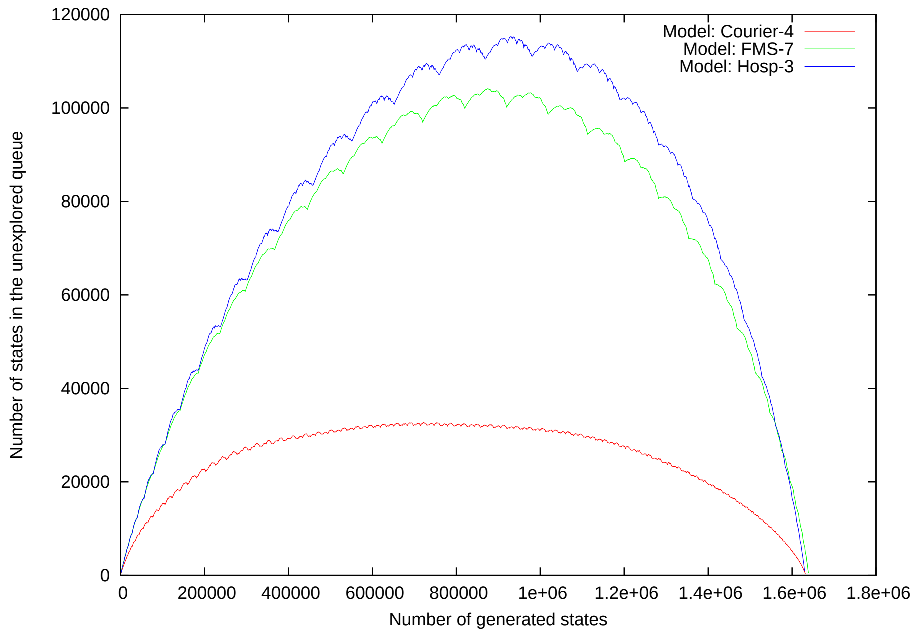


Figure 2: The number of unexplored states during the state-space generation process for three similarly-sized GSPN models.

2 Breadth-First Search

In DNAmaca [7], the breadth-first search algorithm shown in Fig. 1 is employed to explore the model's state-space. This starts from an initial state s_0 and uses a FIFO queue (F) and a list of explored states (E) to generate the state graph (A). The functions `insert()` and `remove()` add a state to and extract a state from F respectively, while `succ(s)` returns the set of successor states of s. The function `id(s)` returns a unique sequence number for state s. DNAmaca uses a probabilistic hash-based scheme [7, 8, 9] to store E in a memory-efficient and easily-searched manner. Breadth-first search is favoured over depth-first search as it allows the continuous time Markov chain's (CTMC's) generator matrix Q to be created and written to disk row-by-row without the need to maintain more than one row in memory.

A guide to the progress of the state-space generation process can be gained during the execution of the BFS algorithm by examining the number of states in F. Fig. 2 shows how the length of F changes over the course of state-space generation for three of the models described in Section 4.

3 Least-Squares Fitting Method

We observe that the shape of the graph in Fig. 2 can be approximated by a curve with equation $y = ax^2 + bx$, for some values of a and b. In order to find these values we use the GNU Scientific Library (GSL) [4] to perform multiparameter fitting using least squares. This fits a model of p parameters to n observations – in our case, there are two parameters (a and b) and the observations are the

Model Name	k	Tangible States
courier2	2	84 600
courier3	3	419 400
courier4	4	1 632 600
courier5	5	5 358 600
fms5	5	152 712
fms6	6	537 768
fms7	7	1 639 440
fms8	8	4 459 455
fms9	9	11 058 190

Table 1: Number of tangible states in the Courier and FMS models in terms of the sliding window size/ the number of unprocessed parts (k).

Model Name	Patients (P)	Nurses (N)	Doctors (D)	Ambulances (A)	Tangible States
hosp1	7	2	2	1	54 228
hosp2	10	2	2	1	561 704
hosp3	11	4	2	2	1 630 905
hosp4	13	4	2	2	5 728 971

Table 2: Number of tangible states in the hospital model in terms of the number of patients (P), nurses (N), doctors (D) and ambulances (A).

current number of unexplored states in F during the BFS process. As the total number of states can be very large, if we were to take observations at each iteration of the BFS process we would potentially have several million and so, to keep the problem size small, we only observe the number of unexplored states once for every 10 000 states generated.

The first step in the fitting process is to express the problem in matrix-vector form $y = Xc$ where y is the vector of n observations, X is an n-by-p matrix of the predictor variables and c is the vector of p unknown best-fit parameters. As we are fitting a polynomial of degree 2 we define $X_{ij} = x_{ij}$ for $0 \leq i \leq (n-1)$ and $0 \leq j \leq (p-1)$. We then employ the `gsl_multifit_linear()` routine (which implements the modified Golub-Reinsch singular valued decomposition algorithm [5] with column scaling) to find the values of a and b which yield the best fit to the observations.

4 Results

To demonstrate the power of our approach, we present results using three Generalised Stochastic Petri Net (GSPN) models. GSPNs are attractive as they typically feature a small number of parameters (tokens) which can be easily varied to produce CTMCs of differing sizes. The GSPN in Fig. 3 models the ISO Application, Session and Transport layers of the Courier sliding-window

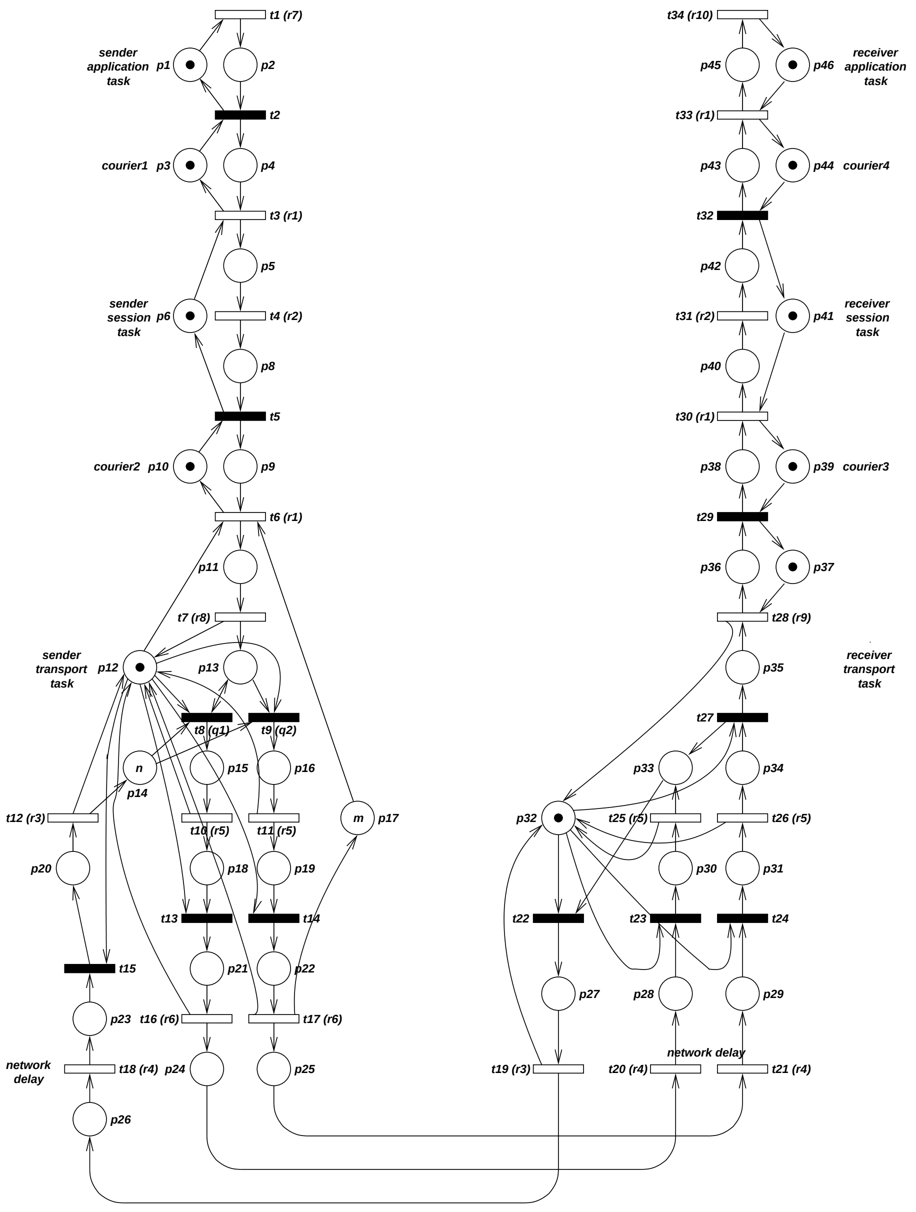


Figure 3: GSPN model of the Courier communications protocol [13].

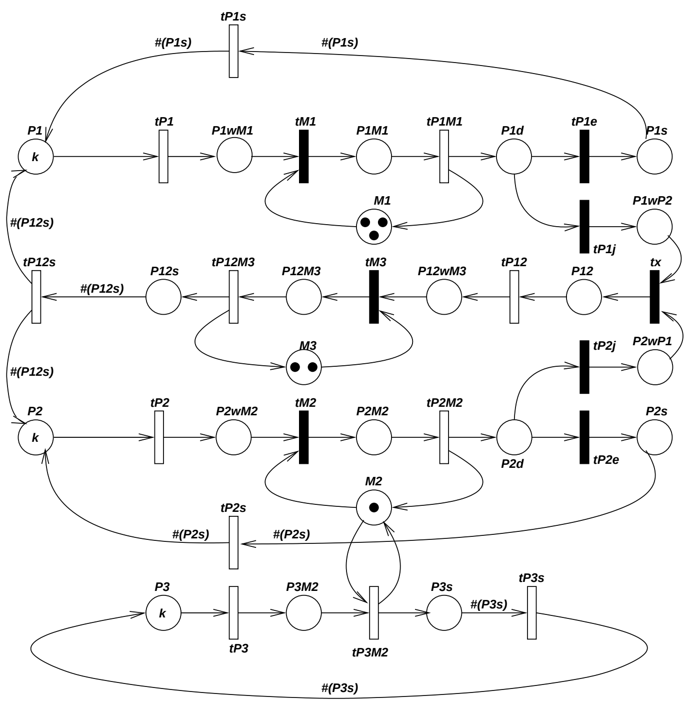


Figure 4: GSPN model of a Flexible Manufacturing System [1].

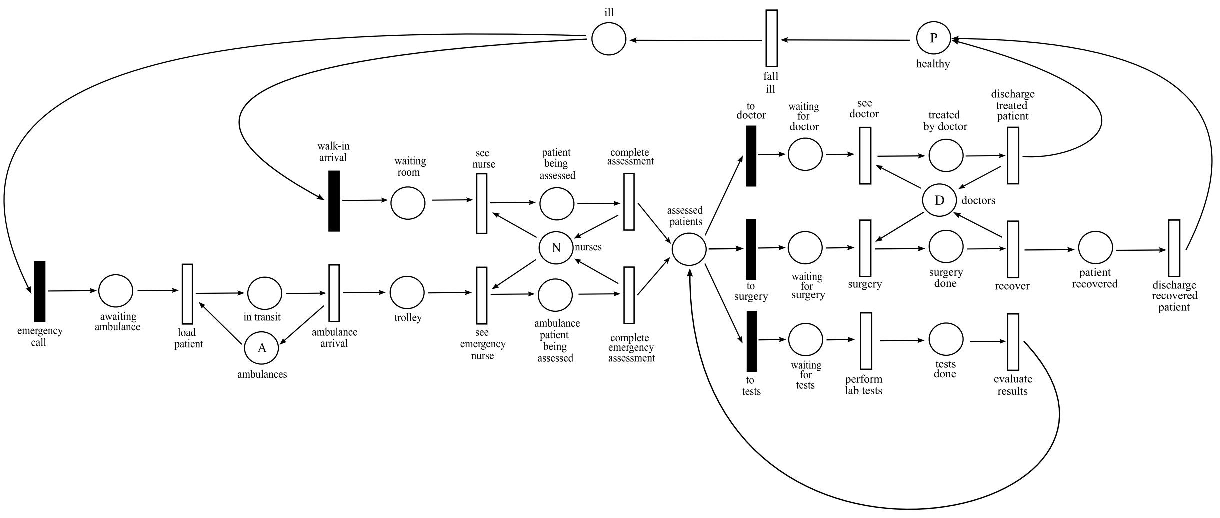


Figure 5: GSPN model of patient flow in a hospital environment.

Model Name	Tangible States	25% Estimate	25% % Error	50% Estimate	50% % Error	75% Estimate	75% % Error
courier2	84 600	47 440	43.9%	72 617	14.2%	83 720	1.0%
courier3	419 400	180 880	56.9%	317 845	24.2%	404 244	3.6%
courier4	1 632 600	707 434	56.7%	1 153 623	29.3%	1 514 634	7.2%
courier5	5 358 600	2 458,700	54.1%	3 769 918	29.6%	4 824 976	10.0%
fms5	152 712	116 370	23.8%	149 793	1.9%	163 166	6.8%
fms6	537 768	375 743	30.1%	517 008	3.9%	570 206	6.0%
fms7	1 639 440	1 113 737	32.1%	1 548 048	5.6%	1 721 391	5.0%
fms8	4 459 455	2 982 118	33.1%	4 181 830	6.2%	4 671 613	4.8%
fms9	11 058 190	7 435 833	32.8%	10 304 305	6.8%	11 532 975	4.3%
hosp1	54 228	53 630	1.1%	61 825	14.0%	63 015	16.2%
hosp2	561 704	414 694	26.2%	582 222	3.7%	633 916	12.9%
hosp3	1 630 905	1 193 169	26.8%	1 666 353	2.2%	1 809 587	11.0%
hosp4	5 728 971	4 053 685	29.2%	5 784 496	1.0%	6 303 895	10.0%
		Average error	34.4%		11.0%		7.6%

Table 3: Difference between the number of states predicted by least-squares fitting and the actual number generated. Results are presented at three points in the state generation process for each of the three GSPN models.

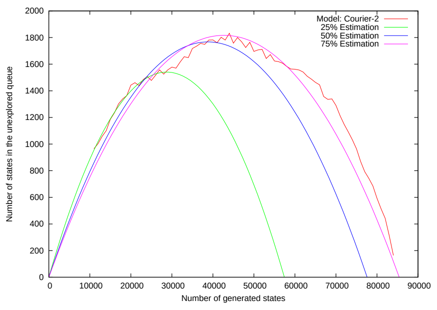
communication protocol [13]. Data flows from a sender (p_1 to p_{26}) to a receiver (p_{27} to p_{46}) via a network. The sender's transport layer fragments outgoing data packets; this is modelled as two paths between p_{13} and p_{35}. The transport layer is characterised by two important parameters: the sliding window size n (p_{14}) and the transport space m (p_{17}). In our investigations we will be varying n to produce varying sizes of state spaces.

Fig. 4 shows a 22-place GSPN model of a flexible manufacturing system [1]. The model describes an assembly line with three types of machines ($M1$, $M2$ and $M3$) which assemble four types of parts ($P1$, $P2$, $P3$ and $P12$). Initially, there are k unprocessed parts of each type $P1$, $P2$ and $P3$ in the system. There are no parts of type $P12$ at start-up since these are assembled from processed parts of type $P1$ and $P2$ by the machines of type $M3$. When parts of any type are finished, they are stored for shipping on places $P1s$, $P2s$, $P3s$ and $P12s$.

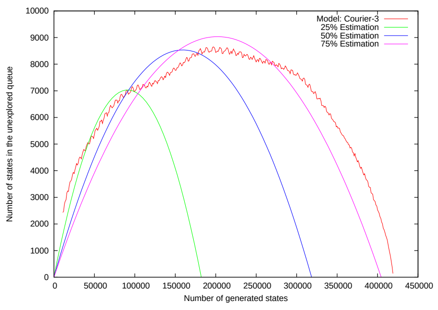
Fig. 5 shows a GSPN model of a hospital's Accident and Emergency department. The key parameters in this model are the numbers of patients (P), nurses (N), doctors (D) and ambulances (A).

Table 1 shows the number of states in the underlying CTMCs for the Courier and FMS models in terms of the parameters in the GSPN models. Note that the first column gives a short name for each of the configurations by which we will refer to it for the remainder of this paper. Likewise, Table 2 contains the number of tangible states in the hospital model for various values of P, N, D and A, along with an associated short name in the first column.

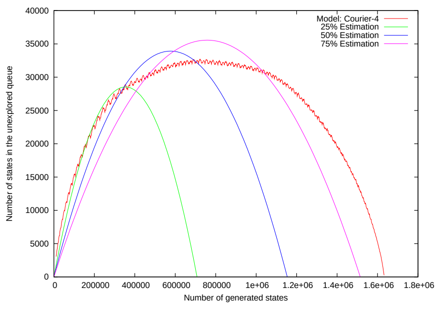
The results of the estimation process using least-squares fitting are shown graphically in Figs. 6, 7 and 8 for each of the three GSPN models. The estimation was performed at three points in the state generation process where 25%. 50% and 75% of the total number of states had been generated. The number of states predicted at each of these points for the three models, along with the



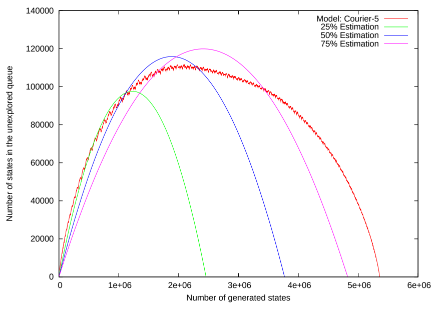
(a) *courier2*



(b) *courier3*

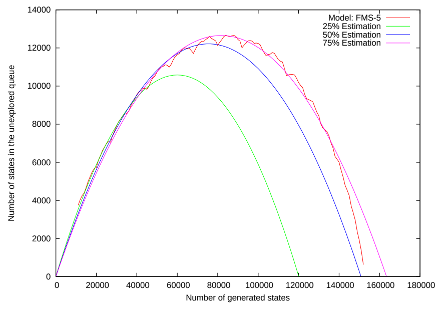


(c) *courier4*

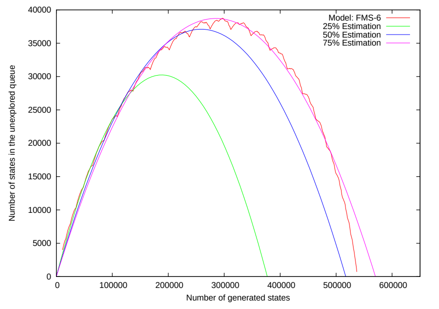


(d) *courier5*

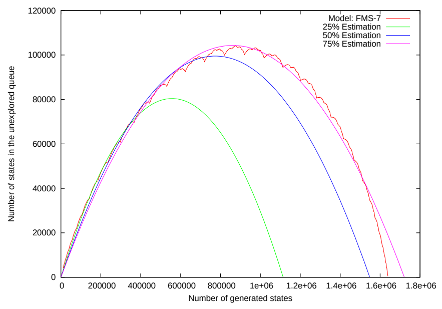
Figure 6: Least-squares fitting estimation of the number of states in the Courier model at 25%, 50% and 75% of the state generation process.



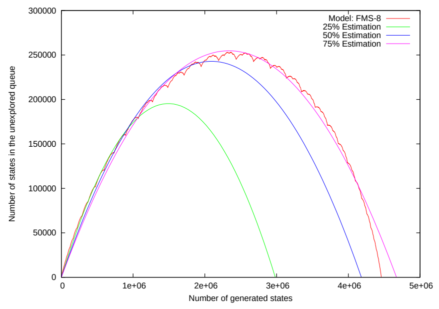
(a) *fms5*



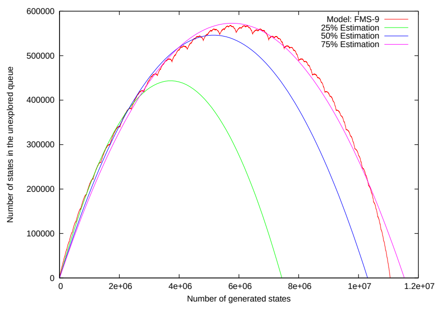
(b) *fms6*



(c) *fms7*

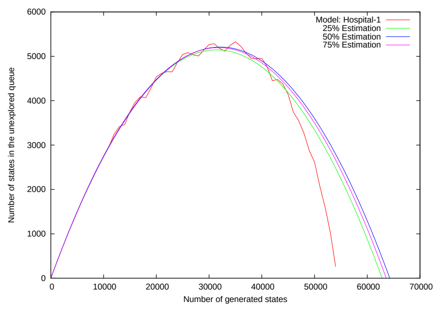


(d) *fms8*

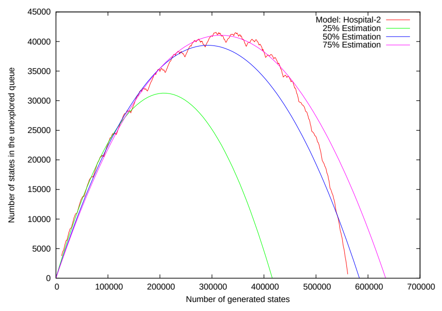


(e) *fms9*

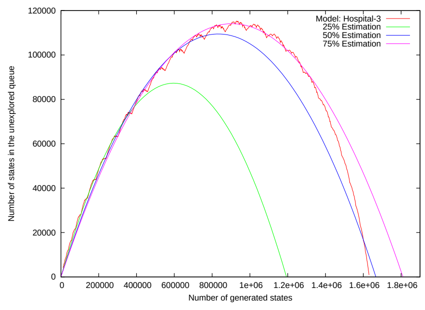
Figure 7: Least-squares fitting estimation of the number of states in the FMS model at 25%, 50% and 75% of the state generation process.



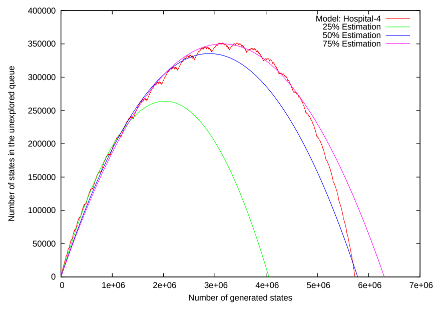
(a) *hosp1*



(b) *hosp2*



(c) *hosp3*



(d) *hosp4*

Figure 8: Least-squares fitting estimation of the number of states in the Hospital model at 25%, 50% and 75% of the state generation process.

percentage error compared with the actual final amount, is given in Table 3.

We observe that early on in the state generation process the estimation of the total number of states varies greatly from the actual number in all but the smallest Hospital model (*hosp1*), with an average percentage error of 34.4%. By the half-way stage, however, the estimate is usually much more accurate (all within 7% in the case of the FMS model) and the average error falls to 11.0%. When three-quarters of the state space has been generated the estimation further improves (with an average overall error of 7.6%), although the accuracy does decrease compared with the half-way point estimate in the Hospital model. Nevertheless, we believe that the ability to estimate to within approximately 10% of the actual total number of states by the half-way point in the generation process demonstrates the applicability of our technique.

5 Conclusion

We have demonstrated how least-squares fitting can be used to accurately estimate the total number of states in the underlying CTMCs of high-level models during the state generation process. Results from our experiments suggest that estimates produced in this way do provide a good guide to the likely eventual number of states. On average, an error of 11.0% in the predicted total was observed at the half-way point in the state generation process.

With the accuracy of our technique demonstrated we will now investigate incorporating it into DNAmaca and derived tools such as HYDRA [3]. In particular, it would be interesting to employ the dynamic process management features of the MPI-2 parallel programming library [6], in conjunction with our estimation method and a parallel state-space generator [8], to automatically spawn extra processors when analysing large models.

References

[1] G. Ciardo and K.S. Trivedi. A decomposition approach for stochastic reward net models. *Performance Evaluation*, 18(1):37–59, 1993.

[2] D.D. Deavours and W.H. Sanders. "On-the-fly" solution techniques for stochastic Petri nets and extensions. *IEEE Transactions on Software Engineering*, 24(10):889–902, 1998.

[3] N.J. Dingle. *Parallel Computation of Response Time Densities and Quantiles in Large Markov and Semi-Markov Models.* PhD thesis, Imperial College, London, United Kingdom, 2004.

[4] M. Galassi, J. Davies, J. Theiler, B. Gough, G. Jungman, M. Booth, and F. Rossi. *GNU Scientific Library: Reference Manual.* Network Theory Ltd., February 2003.

[5] G.H. Golub and C.F. van Loan. *Matrix Computations.* John Hopkins University Press, 3rd edition, 1996.

[6] W. Gropp, E. Lusk, and R. Thakur. *Using MPI-2: Advanced features of the Message-Passing Interface.* MIT Press, Cambridge, Massachussetts, 1999.

[7] W.J. Knottenbelt. Generalised Markovian analysis of timed transition systems. Master's thesis, University of Cape Town, Cape Town, South Africa, July 1996.

[8] W.J. Knottenbelt. *Parallel Performance Analysis of Large Markov Models.* PhD thesis, Imperial College, London, United Kingdom, February 2000.

[9] W.J. Knottenbelt, P.G. Harrison, M.A. Mestern, and P.S. Kritzinger. A probabilistic dynamic technique for the distributed generation of very large state spaces. *Performance Evaluation*, 39(1–4):127–148, February 2000.

[10] R. Pelánek and P. Šimeček. Estimating state space parameters. In *PDMC'08, Proc. 7th Intl. Workshop on Parallel and Distributed Methods in Verification*, Budapest, Hungary, March 2008. Elsevier.

[11] E. Tronci, G. Della Penna, B. Intrigila, and M.V. Zilli. A probabilistic approach to automatic verification of concurrent systems. In *Proc. 8th IEEE Asia-Pacific Software Engineering Conference (APSEC 2001).*

[12] J.F. Watson III and A.A. Desrochers. State-space size estimation of Petri nets: A bottom-up perspective. *IEEE Transactions on Robotics and Automation*, 10(4):555–561, August 1994.

[13] C.M. Woodside and Y. Li. Performance Petri net analysis of communication protocol software by delay-equivalent aggregation. In *Proceedings of the 4th International Workshop on Petri nets and Performance Models (PNPM'91)*, pages 64–73, Melbourne, Australia, 2–5 December 1991. IEEE Computer Society Press.