PLAT – Production Lines Analysis Tool

P. Fernandes (1), M.E.J. O’Kelly (2), C.T. Papadopoulos* (3), and A. Sales (1)

(1) Pontifical Catholic University of Rio Grande do Sul, Av. Ipiranga, 6681 – Prédio 32
90619-900 – Porto Alegre, RS, Brazil. Phone: +55 51 3320 3611. Fax: +55 51 3320 3621.
Emails: paulo.fernandes@pucrs.br, afonso.sales@pucrs.br

(2) Waterford Institute of Technology, Cork Road, Waterford, Ireland.
Phone: +353 51 302 000. Fax: +353 51 378 292. Email: mejokelly@gmail.com

(3) Aristotle University of Thessaloniki, Thessaloniki, GR-541 24 Greece.
Phone: +30 2310 99 74 66. Fax: +30 2310 99 64 52. Email: hpap@econ.auth.gr

Abstract

This paper presents a software tool, called PLAT (Production Lines Analysis Tool), which computes
exact stochastic solutions of models for reliable production lines. The user interface accepts the fol-
lowing parameters of serial production lines: the number of reliable stations \((K)\), the service rate of
each station \((\mu_i)\) and the buffer slots in front of each station \((B_i)\). In the case of a transient solution,
the analyst inputs the initial state of the specified production line and the amount of time over which
the transient solution is required. The solution method implemented consists of the generation of an
equivalent Stochastic Automata Network (SAN) model, its conversion to a Kronecker representation,
and the determination of the exact numerical solution through pre-existent SAN solvers (PEPS2007
and GTAEXPRESS). Transient and stationary probabilistic distributions of the system are developed to
give various performance measures such as throughput, server utilization, average number of jobs and
sojourn times.

Keywords: Automatic Performance Analysis, Markov Processes, Structured Markovian Models, Tensor
(Kronecker) Algebra, Stochastic Automata Networks, Serial Production Lines

1 Introduction

There is a great interest in the determination of the exact solution of production lines because many
approaches to the solution fail to deliver precise values for throughput, station busy rates, buffer occupancy
and sojourn times. Moreover, even the determination of approximate solutions is challenging to the practi-
tioner as the processes involved are complex [1]. Fortunately, some recent results [2] offer exact solutions
for a class of production lines using structured Markovian representation based on Kronecker formulations.
This approach is quite effective in developing both stationary and transient exact analysis of production lines
of reliable exponential stations with limited intermediate buffers with a blocking behaviour. The require-
ments in respect to CPU time and computer memory to achieve these exact solutions are reasonable, and
this development represents a good alternative (i) to fast and inexact analytical solutions and (ii) to solution
approaches based on traditional simulation methods, which are generally time consuming.

This paper describes an automatic tool, called PLAT, to generate the exact solution of production lines
composed of reliable stations, with exponential servers, finite intermediate buffers and blocking behaviours.

*Corresponding author (hpap@econ.auth.gr). The order of authors is merely alphabetical. Paulo Fernandes is funded by CNPq-
Brazil (PQ 307284/2010-7). Afonso Sales receives grants from CAPES-Brazil (PNPD 02388/09-0).
Once the production line is defined, the PLAT tool generates the equivalent structured Markovian model and solves the system by using specialized software packages, in order to produce stationary or transient solutions as required.

The input and output of the PLAT software tool is customised by using production line terminology. Because of this, the user does not need to fully understand the modelling issues involved in the solution. The production line specification consists of the number of stations, service rates, buffer sizes, and, in the case of transient solution, the initial state and amount of time to consider. The solution is expressed in throughput of the line station, buffer occupation, server utilization and sojourn time at each station and precedent buffer.

In this paper we do not present further details on how the models are translated into a structured representation, or how these structured models are actually solved. Such information is given in [2].

The remainder of this paper is organized as follows. Section 2 describes the production lines that can be modelled by PLAT. Section 3 briefly presents the main modules of the tool and Section 4 illustrates the use of these modules through a small example. Finally, the conclusion discusses the features planned to be added to PLAT.

2 The production lines modelled in PLAT

In terms of classical queuing theory, production lines would be described as finite buffer tandem queuing systems where the workstations are the servers, storage facilities are the buffers or the waiting lines and the jobs are the customers. In Figure 1, which depicts a K-station production line, $M_i$ ($i = 1, 2, \ldots, K$) represents station $i$ and $B_i$ denotes the capacity of the buffer located in front of station $M_i$. Jobs enter station 1 from buffer $B_1$ of unrestricted capacity according to a Poisson distribution with mean arrival rate equal to $\lambda$. Each job enters the line at station 1, passes through all stations in order and leaves the $K$-th station (last) in finished form. All jobs at each station are processed according to a First-In-First-Out (FIFO) queuing discipline. In each station there is a single server.

![Figure 1: A K-station production line with intermediate buffers.](image)

PLAT determines, in an automated fashion, the exact stationary and transient solutions of production lines with single server stations. The assumptions of these production line models are as follows: (i) the service times at the single server station are exponentially distributed random variables with mean rates equal to $\mu_i$ ($i = 1, 2, \ldots, K$). In general, the service rates need not be identical (i.e., $\mu_i \neq \mu_j$ for $i \neq j$); (ii) all buffers between successive stations have finite capacities not necessarily of the same size; (iii) blocking of a station occurs as long as the downstream buffer is full and a unit which has been serviced at the station cannot exit from the station; (iv) stations are assumed to be perfectly reliable, i.e., they do not fail; (v) the general rule that deliberate idleness at a station is not allowed applies; (vi) a basic assumption is that the first station is never starved and the last station is never blocked. Although the arrival process is assumed to be Poisson, it is a necessary assumption of the model that the first station is never starved. This assumption characterizes the saturated line of the saturation model. The fact that the last station is never blocked relates to the storage capacity for final products. Much of the earlier research to determine the exact steady state solution of such systems concentrated on the so called “state explosion” issue in that as the number of stations and buffer sizes increase there is a very dramatic increase in the number of distinct states of the production line leading to computational difficulties. Generally, in the past, the system under consideration was modelled as a two-dimensional stochastic process $N(t) = [N1(t), N2(t)]$. See
Papadopoulos et al. [3] for the detailed construction of the underlying Markovian model. The states of the $K$-station line are described by the vector: $(n_2, s_2, \ldots, n_K, s_K)$ where, $n_i$ ($i = 2, \ldots, K$) denotes the status of buffer $i$ and $s_i$ denotes the status of station $i$. If $s_i = 0$, station $i$ is idle, if $s_i = 1$, station $i$ is busy, and if $s_i = 2$, station $i$ is busy and blocking preceding station. $n_1$ and $s_1$ are not included in the state vector as it is assumed that $n_1 \geq 1$ and $s_1 = 1$. The solution approach for solving the steady state solution of the Markovian model of this $K$-station production line consists of the following steps: (i) derivation of the states of the system. Two formulas are given in Papadopoulos et al. [3] for obtaining the number of states for production lines with equal and unequal buffer capacities, respectively; (ii) ordering of the states. This process facilitates the determination of the structure of the transition matrix; (iii) generation of the transition matrix (or the conservation matrix). The generating steps of a recursive algorithm are given in Papadopoulos et al. [3]; (iv) the solution of the resulting system of linear equations. An appropriate method for the exact solution of this system of equations is the Successive Over Relaxation (SOR) iterative method. The associated solution algorithm was coded in C++ by Heavey and is available with appropriate instructions at the website http://purl.oclc.org/NET/prodline associated with the text by Papadopoulos et al. [4] with the abbreviated name MARKOV.

The Markovian model and associated SOR iterative method to determine the steady state solution of the production lines described above become unwieldy if the number of states in the system exceeds 300,000 or so. There is no published available approach to determine the exact transient solution to production lines of such state space dimensions. Because of the difficulties associated with the determination of the exact steady state solutions of large production lines considerable research effort was devoted to the approximate steady state solution of these lines by such approaches as decomposition and aggregation methods (Gershwin [1] and Lim, Meerkov and Top [5]).

Essentially, what PLAT performs is the translation of the above production line into a SAN model and determines the exact steady state and transient solutions using specially developed software. PLAT is capable of obtaining exact steady state and transient solutions of systems with up to 100 million states. The user need only give the specifications of the production line in a specified format for the steady state solution and the same specifications together with the time horizon and initial conditions if a transient solution is required. The process of determining the steady state and transient solutions thereafter is completely automatic if the analyst so desires. In effect, the user need not necessarily understand the modelling, mathematical or computing processes involved. However, there is a facility in PLAT whereby the analyst can choose which software tool to use in developing the solution.

3 Tool modules

PLAT is divided into three (3) modules: (i) a module that generates an equivalent SAN model to the production line specification provided by the user; (ii) a module that determines the exact numerical solution of the model, i.e., a module that translates the SAN into a Kronecker form and computes its stationary or transient probability distribution; and finally, (iii) a module that computes the performance measures for the stations and buffers of the production line.

3.1 Model generation

In this first module, the user must specify the number of reliable stations ($K$) in the serial production line, as well as the buffer size in front of each station ($B_i$) and the service rate of each server ($\mu_i$), where $i$ indicates the index of the $i$-th station. This module automatically generates the equivalent SAN model to
the production line as specified, i.e., it performs the translation of the production line into a SAN model, using the input parameters provided by the user. See [2] the detailed construction of a SAN model from the specification of a production line is given. Using this SAN model, the PSS (Product State Space) and the RSS (Reachable State Space) of the model are computed and an appropriate solution method is recommended to the analyst depending on the characteristics of the production line. The exact performance measures of the production line are computed.

3.2 Solution methods

PLAT provides two probabilistic distributions, stationary and transient, of the modelled system, where the exact numerical solution is computed through pre-existent SAN solvers: PEPS2007 [6] or GTAEXPRESS [7]. PEPS2007 is a software tool package for modelling and solving SAN models. It uses a Kronecker descriptor to represent the transitions of the model in a memory efficient way. The SAN model is solved using this descriptor and applying vector-descriptor product (VDP) methods, such as Shuffle [8] or Split [9]. On the other hand, GTAEXPRESS stores the transitions of the model in a Harwell-Boeing Format (HBF) sparse matrix [10], which is computed from the reachable states of the model stored into a MDD (Multi-valued Decision Diagrams) structure [11]. From this HBF matrix, the solution of the model is computed applying the classical vector-matrix product (VMP) method (for models with up to few millions of reachable states). This solution method is provided by GTAEXPRESS as SAN Lite-Solver. For larger models (i.e., models where the RSS is the order of billions of states), which cannot be solved using GTAEXPRESS (SAN Lite-Solver), or even PEPS2007 (Shuffle or Split), an approximate solution can be obtained by the Perfect Sampling method [12]. PLAT chooses the appropriated pre-existent software tool to compute the probability distribution for the model states.

3.3 Performance measures

Given the probability distribution obtained by either PEPS2007 or GTAEXPRESS, PLAT integrates the probabilities computing the performance measures for each station and its precedent buffer. Specifically, it computes for each pair, consisting of a station and its precedent buffer: throughput of the station; average number of jobs in the buffer (buffer occupation); station server utilization; and sojourn time in the buffer and the station.

4 Example of usage

In this section an example of the use of the PLAT tool is presented. Assume an eight-station production line \((K = 8)\), where all stations have a single server, buffer sizes \(B_2 = 1, B_3 = 2, B_4 = 3, \ldots, B_8 = 7\), and the service rate of each server \(\mu_i = i\) (where \(i = 1, 2, \ldots, 8\)).

Figure 2 presents some screen shots for this example using PLAT. Initially, the user must specify the production line, which is to be analyzed - Figure 2 (a). Then PLAT computes both the PSS and the RSS of the model and recommends which method of solution to use, based on the characteristics of the model - Figure 2 (b). Note that this model has a RSS around 500 thousand states, which is easily stored into the main memory of an ordinary computer.

---

1. PSS is the cartesian product of the local states of all automata of the SAN model. And, from an initial state, only a subset of PSS could be reachable, defining the RSS (Reachable State Space) of the model.
A machine with a Intel Xeon E5520 processor, which runs at 2.27 GHz, is used and PLAT computes the performance measures of all stations - Figure 2 (c) - applying GTAEXPRESS (SAN Lite-Solver) solution with around 60 KBytes memory to store the HBF matrix and about 95 seconds to solve the model. The reader interested in using PLAT to obtain performance measures of a serial production line can apply it at the address [http://marfim.lad.pucrs.br:16000/plat/](http://marfim.lad.pucrs.br:16000/plat/).

### 5 Conclusion

The software tool PLAT offers an easy-to-use procedure to determine the exact solution of a class of production lines. Such a solution procedure represents a good alternative to approximate analytical and simulation solutions of such lines since it provides within a reasonable time exact stationary predictions. But PLAT can also provide transient solutions which is, to the authors’ best knowledge, a new development in the analysis of production lines.

The current limitations of PLAT include the classes of production lines (reliable stations, exponential single server, limited buffer and only blocking behaviour). However, such limitations will be removed in future versions of PLAT. The inclusion of production lines with stations with multiple servers is probably the easiest extension to add, since it can be achieved with load-depandent transition rates. The inclusion of such transition rates in the model allows also for the possibility of analysing lines with stations at which the service time can be accelerated or slowed according to buffer occupancy. The representation of non-exponential servers is a further feasible development by the use of phase-type distributions, but such an approach usually increases the memory and time demands for solutions. The inclusion of unreliable stations is also feasible, but this would add further complexity. Finally, the consideration of behaviours other than blocking is likely to be simple to incorporate, and it in certain circumstances may actually represent a reduction in model complexity.
References


