Acquisition of accurate or approximate throughput formulas for serial production lines through genetic programming

Konstantinos Boulas
Management and Decision Engineering Laboratory Dept. of Financial and Management Engineering, University of the Aegean 41 Kountouriotou Street, 82100 Chios, Greece

Georgios Dounias
Management and Decision Engineering Laboratory Dept. of Financial and Management Engineering, University of the Aegean 41 Kountouriotou Street, 82100 Chios, Greece

Chrissoleon Papadopoulos
Dept. of Economics, Aristotle University of Thessaloniki, 54124 Thessaloniki, Greece

Athanasios Tsakonas
Management and Decision Engineering Laboratory Dept. of Financial and Management Engineering, University of the Aegean 41 Kountouriotou Street, 82100 Chios, Greece

Abstract

Genetic Programming (GP) has been used in a variety of fields to solve complicated problems. This paper shows that GP can be applied in the domain of serial production systems for acquiring useful measurements and line characteristics as throughput. Extensive experimentation has been performed in order to set up the genetic programming implementation, and to deal with problems like code bloat or over fitting. Further work is needed, but so far, results are encouraging.

KEYWORDS

Serial production lines, genetic programming, symbolic regression

1. INTRODUCTION

A basic sub-category of production systems is that of serial production lines. A K-station production line with K-1 intermediate buffers is a system in which, each part enters the system from the first station, passes in order from all the stations and the intermediate buffer locations and exits the line from the last station. The description and the assumptions of the model are given in (Papadopoulos, et al., 2002). The basic performance measure in the analysis of production lines is the mean production rate or throughput. Another useful parameter of the system is utilization which is defined as the ratio of mean arrival rate to mean service rate. The domain of serial production lines is complicated due to combinatorial explosion, depending on the number of workstations involved in the examined line, the capacity of buffers existing within the workstations etc (Dallery & Gershwin, 1992). The number of feasible allocations of B buffer slots among the K-1 intermediate buffer locations increases dramatically with B and K and is given by the formula (1) :

$$\binom{B-K+2}{K-2}=\frac{(B+1)(B+2)\ldots(B+K-2)}{(K-2)!} \tag{1}$$

Consequently general formulas for acquiring useful measurements and line characteristics, such as throughput, either can be found for only very short lines with simple characteristics (Hunt, 1956) or do not exist at all. For instance a system with 3 stages under the assumptions in (Hunt, 1956) has 8 states. In order to obtain an exact value of production rate involving Markov analysis the result of maximum utilization is a ratio of a numerator polynomial 8^{th} degree containing 22 terms. The denominator polynomial has a degree 7 and contains 24 terms respectively. A system of 6 stages has 144 states and for 10 stages the number of states increases to 6765 (Muth, 1984). These numbers represent an indication of the difficulties one should overcome in order to find an exact solution for more complex models.

The determination of unknown parameters is usually realized through decomposition algorithms and techniques. These methods presuppose the existence of computer assistance to the production engineer in order to be able to dynamically setup the production line in the optimal way. The overall design of a production line's problem is complicated and difficult. As a result, a few approximation methods (Blumenfeld, 1990), (Martin, 1993), (Muth, 1987), (Li, et al., 2015) and algorithms mainly related to computational intelligence (Papadopoulos, et al., 2002) have been used in the past, in an attempt to acquire general formulas.

This work attempts to approach the problem by applying genetic programming techniques (Koza, 1992), (Koza, 1994), (Angeline & Kinnear, 1996), (Poli, et al., 2008) in order to extract useful general formulas. The basic methodological steps are described as follows: The problem is treated by increasing gradually the operating parameters i.e. the number of the stages. A combination of servers which follow the exponential distribution is produced and with the appropriate method an accurate training data set for a dependent characteristic is obtained. So far, we have examined four different algorithms to form our training data sets for different line settings.. Using the implementation as presented in (Papadopoulos, et al., 2009) we concluded that our training set will be formed firstly by the available till now exact formulas, secondly by applying the decomposition method beyond the exact formulas. With the available data set we apply a genetic programming algorithm in order to acquire a formula for the given system. The aim is to find an accurate formula for the characteristic of the system using a simple function set with terminals, parameters of the system, such as mean service rate. Some accurate formulas from Hunt's work (Hunt, 1956) based on Markovian analysis are confirmed for short lines with simple characteristics and now experiments are extended in order to acquire more complex formulas. At the same time the influence of genetic programming setup parameters is investigated in order to improve the effectiveness of the proposed approach.

The rest of the paper is organized as follows: Section 2 briefly describes the basic methodological steps of the current approach. Then a reference is made to genetic programming and some results are given. In Section 3 concludes the paper with the discussion of results and future research.

2. METHODOLOGY

As already stated above, the objective of this work, is to obtain accurate or approximate formulas that characterize the examined system in terms of the production line parameters (i.e., the number of stations, size of buffers, mean processing time), assuming there are sufficient jobs at the beginning of the line to ensure that the first station is never starved of jobs and that the last station is never blocked. Hunt's formulas are used in order to generate the complete training data set that will be later used as input to the genetic programming scheme for the acquisition of the accurate or approximate formulas.

2.1 Major Steps

To identify accurate or approximate generalized formulas for calculating the throughput of short (i.e. K=2,3) serial production lines with or no intermediate buffers, the methodological steps, given below, were followed for each value of number of workstations K:

If K=2, 3:

1. STEP 1: Identify the exact throughput or utilization values for this K, using the Hunt's accurate formulas.
2. STEP A2: Initiate an iterative application of a genetic programming scheme in the full data set acquired from Step 1, to obtain an approximate formula containing basic algebraic operations.
3. STEP 3: Stop the genetic programming process, when an accurate formula has been obtained for the training data set or another parameter of genetic programming has reached the limits set (i.e. maximum number of generation, maximum length of solution).
4. STEP 4: If the genetic programming execution has identified a solution, then transform this solution into a useful infix notation.

2.2 Genetic Programming Implementation

Genetic programming techniques are applied in this paper, to approximate the calculation of throughput or utilization in short serial exponential production lines. Due to the complexity of the solution space of the problem, an encoding in a hierarchical expression is required like the tree representation of a solution. In genetic programming, a population of random trees is initially generated, representing programs, i.e. candidate solutions of the given problem. Then, the genetic operations i.e. crossover and mutation are performed on these trees. We use the standard flattened (linear) representation for trees in a prefix notation. The number of generations in an execution varies from 50 to 1000 and the population ranges from 2000 to 40.000 or more, up to 200.000. The initialization method is the growth method or the half and half ramped

method; the selection is performed by using tournament selection. Crossover happens at about 90% at each generation and the used point mutation is set at about 2%. A generation is concluded when for the size of the examined population all crossover and mutation operations have been performed. In each generation we obtain statistics and a formula with the best fitness of the population which is transformed in infix representation. If the formula satisfies the accuracy criterion then is treated further i.e. is simplified in order to improve the readability and to be easily manageable.

In order to create a population, a function set F and a terminal set T, are primarily defined. The terminal set contains variables as the mean service rate, μ_i or the buffer size, B_i for each stage of the process. The functions must be able to "pass" information between each other. The term describing this need is called closure achievement. The function set includes addition, subtraction, multiplication and protected division. In order to create a random tree, we select randomly from Function set, until all tree branches end in terminals. A generation is considered "concluded", when the number of crossover / mutation operations that have been performed is equal to the number of individuals which compose the population. The selection of each candidate individual for the participation of genetic operation is tournament selection. Sub tree crossover is used. The selection of crossover points is uniform, so every node is chosen equally likely. Point mutation is used. The candidate nodes for mutation in the tree are randomly chosen. Grow or the half and half ramped initialization method is used to create the initial population. After considering the initialization of a random population and the operators selection, the next step is to determine a fitness function, which will be used for the evaluation of the candidate solutions, and therefore for the selection of the individuals for the crossover and other operations. The fitness function is the sum of the absolute differences between the actual program output calculated with an iterative procedure which exploits the prefix notation of tree representation and the desired output for each fitness case. For each generation, statistics are calculated and collected for the evaluation of the overall procedure.

The implementation of the genetic programming system creates LISP-like symbolic expressions. The structure of the final solution is not predetermined and is created by performing the genetic programming operators. Therefore, a problem is searched through all the candidate combinations of symbolic expressions, which are defined initially by the solver. The genetic programming process followed in our work can be divided into the following five steps:

STEP G1: Create a random population of programs using the symbolic expressions provided.
STEP G2: Evaluate each program assigning a fitness value according to a pre-specified fitness function, which actually corresponds to the ability of the program to solve the problem.
STEP G3: Use reproduction techniques to copy existing programs into the new generation.
STEP G4: Recombine genetically the new population with the crossover or mutation operation from a randomly based chosen set of parents after tournament selection.
STEP G5: Repeat steps 2 - 4 until a termination criterion has been achieved.

To evaluate a program without error (i.e. to achieve closure) it is imperative that the symbolic expressions operated by the genetic programming mechanism are compatible. Therefore, in order to avoid division by zero during the evaluation process, the standard division is substituted by the following expression (called protected division): double div(double x, double y) {if (y == 0) return x; else return (x/y);}

As an initial step in our experiments, to ensure sufficiency, the four basic functions were selected for the considered function set F={+,-,*,%}. As during the training phase the algorithm allocates a large proportion of computer memory, the authors adopted a steady-state genetic process. In steady state genetic programming, the parents to be recombined are selected from the current population using some criteria, but a child is also chosen from the same population. The recombined genetic program is evaluated and it takes the position of the selected child. A generation is considered as completed when the number of created children is equal to the size of the population. In some experiments at some point the average program size starts growing at a rapid pace with no accompanied by any corresponding increase in fitness. This phenomenon is the well known code-bloat and in order to restrict this, some empirical techniques have been implemented. A type of parsimony pressure method is adopted. According to this, the fitness measure is defined to be: fitness = error + size of program/ weight. The value of weight is the result of a trial and error method (Lai, 2003).

3. RESULTS

We consider the case of a line with two or three stations and exponentially distributed processing times. For the case of two identical stations with finite number of buffers at the front of second stage the genetic programming algorithm gave the exact solution as this is described in (Hunt, 1956), (Blumenfeld,1990).

For the case of two stages with $\mu_1 \neq \mu_2$ we present the stages of our approximation. We aim at validating equation 19 of (Hunt, 1956) i.e. the maximum possible utilization. After performing a GP execution, the tree with value of fitness equal to zero is shown in Figure 1. This tree has been obtained from the third (and last) generation of the GP execution. The prefix notation for this tree is: $\rho_{max}=(/(*/\mu_2\mu_1)\mu_1)(+\mu_2(*\mu_1(/\mu_1(+\mu_2\mu_1)))))$. We write again in infix notation: $\rho_{max}=((\mu_2/\mu_1)*\mu_1)/(\mu_2+(\mu_1*(\mu_1/(\mu_2+\mu_1))))$ which actually is the equation 19 of Hunt (i.e. the exact formula):

$$\rho_{\max} = \mu_2(\mu_1+\mu_2)/(\mu_1^2+\mu_1\mu_2+\mu_2^2) \tag{2}$$

For the case of 3 stages with $\mu_1 \neq \mu_3 \neq \mu_3$ and without intermediate buffers, the maximum utilization is given from equations 21 and 22 and 23 of (Hunt, 1956). So far the genetic programming implementation has approximated the exact solution. An approximation is presented in the following equation (3) and the results from this are presented in Figure 2. This formula has been obtained from the 117th generation of a GP execution:

$$\rho_{K=3} = \frac{\mu_2}{\left(\mu_2+\mu_2\left(\frac{\mu_3}{\mu_1+\frac{\mu_1^2\mu_2^2}{4\mu_1^2\mu_3^2-2\mu_2\mu_1^2\mu_3+2\mu_2\mu_1^2+2\mu_2\mu_1\mu_3+\mu_2\mu_3}}+\frac{\mu_1^5\mu_2^3\mu_3}{\mu_1^4\mu_2^5+\mu_1^4\mu_2^4\mu_3+\mu_1^4\mu_2^4\mu_3^2+\mu_1^2\mu_2\mu_3^3+\mu_1\mu_2^2\mu_3^3+\mu_1\mu_3^5+3\mu_2\mu_3^4}\right)\right)} \tag{3}$$

For the same case the throughput is figured out with DECO-2 algorithm (Papadopoulos, et al., 2009). An approximate formula is presented below and results from this formula are shown in Figure 3. This formula (4) has been obtained from the 998th generation of a GP execution:

$$\text{throughput}_{K=3} = \frac{2}{2\mu_2+\mu_3+\frac{\mu_1}{\mu_2+\frac{\mu_2}{\mu_1\left(\mu_3+\mu_1^2\mu_2-\frac{\mu_1^2\mu_2^4\mu_3}{(2\mu_2\mu_1^2+\mu_3)(3\mu_1+\mu_3)}\right)}}} \tag{4}$$

Figure 1: The tree for the case of two stages with $\mu_1 \neq \mu_2$. We are looking for the ρ_{max}.

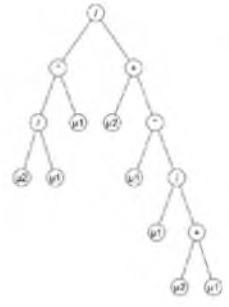


Figure 2: Maximum Utilization for three stages with $\mu_1 \neq \mu_2 \neq \mu_3$

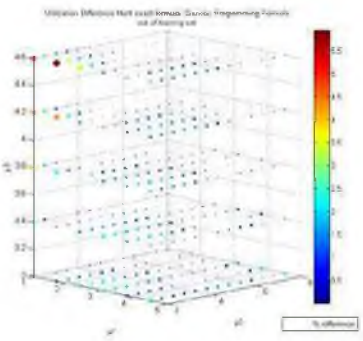
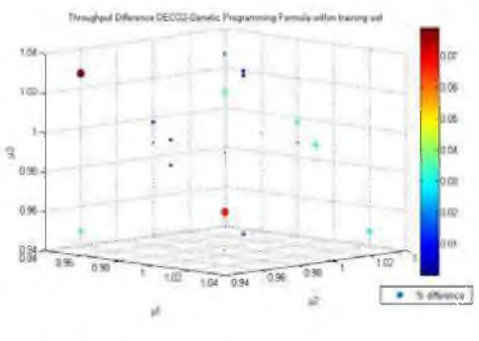


Figure 3: Throughput for three stages with $\mu_1 \neq \mu_2 \neq \mu_3$



3. CONCLUSIONS

In this work we have studied the behavior of a genetic programming implementation for extracting useful accurate or approximate formulas for serial production systems. During extensive experimentation we have

observed that GP-programs are growing in size rapidly which means consuming of computational resources, need for additional processing time and difficulties in fitness improvement. In this sense an effort has been given in order to restrict the code bloat phenomenon. The introduction of the parsimony factor leads to smaller and more manageable programs, while it increases comprehensibility and it reduces the time of an execution. In all GP-experiments the search process evolves rather slowly. The existence of accurate formulas is very important. The fact that we can induce an accurate formula allows the investigation of a very large area like the one represented in Figure 2. We are experimenting with methods to prevent code bloat, to improve the search of solution space, to implement different genetic programming operations, to test other function sets or GP setup. Our aim far ahead is to increase the value of fitness and to obtain solutions for systems with many stages. Note that this is one of the few times that in practice GP-approaches verify existing accurate formulas at any real world application domain. Moreover, it is the first time that this verification concerns accurate (exact) formulas in production lines. This happens for short (two stages) serial production lines. For three stages highly accurate approximate formulas have been discovered so far, performing at an acceptable level of accuracy also out of the range of the training set (out of sample performance), with the use of a rather limited training set. The whole formula discovery process seems to be depended on the quality and special characteristics of the initialization process (a high quality solution obtained during the initialization of the GP-tree, leads quickly to more accurate or even exact solutions). In this direction we plan for the future the design and application of GP-ensemble models, able to perform several parallel GP-executions, in order to ensure the best possible initialization of the evolution process.

ACKNOWLEDGEMENT

This research has been co-financed by the European Union (European Social Fund – ESF) and Greek national funds through the Operational Program "Education and Lifelong Learning" of the National Strategic Reference Framework (NSRF) - Research Funding Program: Thales. Investing in knowledge society through the European Social Fund

REFERENCES

Angeline, P. J. & Kinnear, K. E., 1996. Advances in genetic programming. s.l.:MIT Press Cambridge.

Blumenfeld, D. E., 1990. A simple formula for estimating throughput of serial production lines with variable processing times and limited buffer capacity. 28(6), pp. 1163-1182.

Dallery, Y. & Gershwin, S. B., 1992. Manufacturing flow line systems: a review of models and analytical results. Queueing systems, 12(1-2), pp. 3-94.

Hunt, G. C., 1956. Sequential arrays of waiting lines. Operations Research, 4(6), pp. 674-683.

Koza, J., 1994. Genetic programming II- Automatic Discovery of Reusable Programs. s.l.:Massachusetts Institute of Technology.

Koza, J. R., 1992. Genetic programming: on the programming of computers by means of natural selection. s.l.:MIT press.

Lai, T., 2003. Discovery of understandable math formulas using genetic programming. Genetic Algorithms and Genetic Programming at Stanford, pp. 118-127.

Li, L., Qian, Y., Du, K. & Yang, Y., 2015. Analysis of approximately balanced production lines. International Journal of Production Research, Issue ahead-of-print, pp. 1-18.

Martin, G., 1993. Predictive formulae for unpaced line efficiency. The International Journal Of Production Research, 31(8), pp. 1981-1990.

Muth, E. J., 1984. Stochastic processes and their network representations associated with a production line queuing model. European Journal of Operational Research, 15(1), pp. 63-83.

Muth, E. J., 1987. An update on analytical models of serial transfer lines. In: Research Report No. 87-15. s.l.:Department of Industrial and Systems Engineering, University of Florida Gainesville FL.

Papadopoulos, C. T., O'Kelly, M. E., Vidalis, M. J. & Spinellis, D., 2009. Analysis and design of discrete part production lines. New York: Springer.

Papadopoulos, C., Tsakonas, A. & Dounias, G., 2002. Combined Use Of Genetic Programming And Decomposition Techniques For The Induction Of Generalized Approximate Throughput Formulas In Short Exponential Production Lines With Buffers. s.l., s.n.

Poli, R., Langdon, W. B. & McPhee, N. F., 2008. A Field Guide to Genetic Programming. s.l.:[SL]: Lulu Press (lulu. com), 2008..