информациологических ресурсов;
информациологических технологий;
информациологических технологий;
информациологических результатов;
информациологического эксперимента;
информациологических идей и методов/ интеллекта/;
информациологической интеллектуальной войны;
информациологического интеллектуального щита государства;
информациолого- энергетических технологий;
информациологических компьютериых технологий;
информациологических технических средств;
информациолого- правовой деятельности;
информациологических знаний в книгах;
единого мирового информационно- сотового простраиства.

Классификация информациологических систем контроля и управления по Юзвишину позволяет представить глобальио задачи анализа и синтеза с целью получения эффективиых, качественно новых и оптимальиых сложных систем.

СПИСОК ЛИТЕРАТУРЫ

1.Кузьмин И.В. Оценка эффективности и оптимизация АСКУ. – М.: - Советское радио,1971. –296 с.

2. Юзвишин И.И. Основ информациологии. – М.: Высшая школа, 2000. –517 с.

A. Tsakonas, C. Papadopoulos, G. Dounias (Greece, Chios)

CALCULATION OF THROUGHPUT FOR PRODUCTION LINES WITH BUFFERS USING COMPUTATIONAL INTELLIGENCE

Introduction

The domain of serial production lines lacks the existence of general formulas for acquiring useful measurements and line characteristics, such as throughput. Throughput is called the average number of jobs per hour that can flow through a production line. The obvious complexity of the domain, due to combinatorial explosion, depends on the number of workstations involved in the examined line, the capacity of buffers existing within the workstations, the variability in processing times, etc. The authors attempt to approximate this problem by applying modern genetic programming techniques [Koza 1992], [Koza 1994], [Angeline et. al 1996], in other words creative programming techniques that belong to the area of computational intelligence and learning. Genetic programming is an automated method for creating a working computer program from a high-level problem statement of the problem. The evolutionary search adopted, uses the Darwinian principle of survival of the fittest and is patterned after naturally occuring operations, including crossover (i.e. sexual recombination), mutation, gene duplication, gene deletion, etc. The objective of this work, is to obtain an analytical formula for throughput x, in terms of the abovementioned production line parameters (i.e. of 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. Through this paper, different formulas are given for each size of short production lines with respect to their line length, and then, an additional attempt is described and analyzed for unifying all the throughput formulas obtained during the initial approach. The formulas obtained are quite long but easily programmable in a single line of source code, and thus very useful for immediate use in real world applications.

The problem

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. If a station has completed its processing and the next buffer has space available, the processed part is passed on and the station starts processing a new part that is taken from its input buffer. If the buffer has no parts, the station remains empty, until a new part is placed in the buffer. This process causes the well-known phenomenon of blocking and starving in manufacturing.

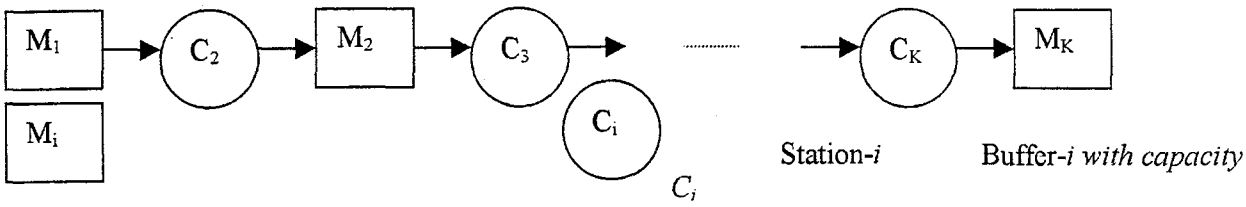

Figure 1: A serial production line (K-stations and K-1 buffers)

The whole system operates under the assumption that the first station is never starved and the last station is never blocked. The processing (service) times at each station are assumed to be independent random variables following the exponential distribution, with mean service rates, μ_i, $i = 1,2,....K$. Breakdowns in the line stations are not allowed in this model. Figure 1 represents a K-station line having *K-1* intermediate locations for buffers, whose capacity is denoted as $C_2, C_3, \ldots, C_K$. The basic performance measures in the analysis of production lines are the mean production rate or "throughput" and the average work-in-progress *(WIP)* or equivalently the average production time. The object of the present work is to identify the line's throughput for various intermediate buffer sizes (*i.e.*, $N = 0,1,2,3,4$), when mean processing time varies from *0.98* to *1.02* and the line is considered short, that is $K = 2, 3, \ldots, 10$. Note that the number of feasible allocations of N buffer slots among the *K-1* intermediate buffer locations, increases dramatically with N and K.

The standard method for approximating the mean throughput of a serial production line at any given conditions, is the decomposition method described, among others, by [Dallery & Frein, 1993]. The decomposition method gives the throughput for any K-station line with finite intermediate buffers of capacity C_i and exponentially distributed processing times $(M_i)^{-1}$. In decomposition methods, the queueing network is decomposed into a set of smaller subsystems. Several decomposition methods have been proposed in literature with respect to limitations such the type of service times, the way the subsystems are characterized and the way the unknown parameters are determined. All of these decomposition approaches characterize the subsystems, then derive a set of equations to determine the unknown parameters of each subsystem, and finally derivate an algorithmic procedure for solving this set of equations and thus for determining the unknown parameters of all subsystems. Then the performance of each subsystem can be calculated and an approximation of the performance of the original system can be provided. Decomposition algorithms and techniques offer adequate and reliable solutions in practice, but presuppose the existence of computer assist to the production engineer in order to be able to dynamically set up the production line in the optimal way. The provision of unified formulas for obtaining the line's throughput at any conditions instead of applying decomposition algorithmic approaches is useful and desirable.

Methodology

To identify approximate generalized formulas for calculating the throughput of short serial production lines with intermediate buffers the methodological steps given below were followed for each value of number of workstations K:

If K=2,3,4: (STEP A1): Identify exhaustively all the exact throughput values for this K, using the decomposition algorithm approach, **(STEP A2)**: Initiate an iterative application of a genetic programming scheme in the full data set acquired from Step A1, to obtain an approximate formula containing basic algebraic operation, **(STEP A3)**: Stop the genetic programming process, when all the parameters for the examined K have been used and the generated formula exceeds 99.9% accuracy (i.e. mean difference among decomposition and approximation values is lower than 0.01%)

If K=5,6,..,10: **(STEP B1):** Apply special sampling techniques for selecting the most representative set of data for the task of "learning" performed through genetic programming, **(STEP B2):** If complexity of the search space remains extremely high, reduce the representative values of the set of alternative mean processing time values, **(STEP B3):** Initiate an iterative application of a genetic programming scheme in

the reduced data set acquired from Steps B1 (and B2), in order to obtain an approximate formula containing basic algebraic operations, **(STEP B4):** Accept or reject the obtained formula with respect to its overall accuracy, not only in the sampling data but in the whole data set for the given K. If the formula obtained is rejected then go back to Step B1, else proceed with the next value of K.

If K>10, Complexity of such a serial production line becomes a difficult barrier to overcome with currently existing genetic programming approaches.

Genetic programming based approximation

Genetic and evolutionary algorithms are used in various domains where a direct search method (e.g. backpropagation in neural networks) cannot be applied due to the nature of the problem. The main disadvantage of a genetic algorithm from the point of developing complex functions is the lack of representation [Koza 1992]. With genetic algorithms, a coding scheme for a fixed-length string must be used, which is a selection constraining the "size" of a solution. When the type of the "near-optimal" solution is not known a-priori, the coding prevents the solver of the exploration of areas of the solution space, which may have better performance. Therefore, for more complex systems, an encoding in a hierarchical expression is required.

This need has led research in the development of genetic programming. When applying genetic programming, much of the theory of genetic algorithms is followed. The basic difference is the form of representation of the structures used, which resembles more than a variable-length tree, than a fixed string. In genetic programming, a population of random trees is initially generated, representing programs. Then, the genetic operations (crossover, mutation etc.) are performed on these trees. In order to create a population, a function set F and a terminal set T is primarily defined. Notice that, the terminal set may contain both, variables and constants. These functions must be able to pass information between each other. The term describing this need is called *closure achievement.*

In order to create a random tree, we usually select randomly from $T \cup F$, until all tree branches end in terminals. However, more advanced techniques on the creation of the initial application have been developed [Langdon 1996] [Koza 1992, [Koza 1994], and these are followed in the implementation of this work. There are generally, four types of operators in genetic programming: crossover, mutation, reproduction and inversion. Crossover and reproduction are considered [Koza 1994] the most important operations. However, under recent development it is suggested [Angeline et. al. 1996] that special types of mutation may offer better search in the solution space. 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 selections of the individuals for the crossover and other operations. Actually, as with genetic algorithms, the fitness function, works separately than the whole genetic procedure, enabling this way the independence of the algorithm. The genetic programming process followed in our work is given through five steps:

(1) Create a random population of programs using the symbolic expressions provided, **(2)** 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, **(3)** use reproduction techniques to copy existing programs into the new generation, **(4)** recombine genetically the new population with the crossover function from a randomly based chosen set of parents **(5)** repeat steps 2-4 until termination criterion has been achieved.

In order to be able to evaluate a program without error (to *achieve closure)* it is necessary the symbolic expressions operated by the genetic programming mechanism be compatible. Therefore in order to avoid division by zero during the evaluation process, "standard division" was substituted by "*protected division*". As an initial step in our experiments, to ensure *sufficiency*, the four basic functions were selected for the considered function set F={+,-,/,*}, intending in future work to enhance it with more non-linear functions such as *tanh*, square and cubic root, etc. While a lot of experimentation may be accomplished in this domain, the authors applied crossover 70 % of time, mutation 20% of time, and straight copy 10% of time. The crossover used is a *sub-tree crossover.*

Results

For the calculation of throughput by the decomposition method, the parameters set to create the genetic procedure train set like convergence criterion, first buffer;s capacity, etc., were carefully chosen.

It is worth to note here that the evolving of the derived data set is exponential. Therefore, as it is made clear that the data set derived for number of stations greater than four (4), limits the genetic process, while the train set becomes too large to be handled by present technology. This is the reason why

sampling techniques are proposed to handle the throughput approximation, for the case of five (5) or more workstations of a serial production line with intermediate buffers and exponential service times. Specifically, a solution in this problem could be the sampling by changing the calculation interval (e.g. 0.02 for station service instead of 0.01), losing this way precision on the resulting formula but making the computation possible. In all cases, the abovementioned ranges are considered, which are for station service rates from 0.98 to 1.02 and for buffer capacities from 0 to 4. It is easily observed that under this fixed sampling, the possibility of learning is reduced when the number of stations is increased, while the train data set contains even smaller proportions of the original set. However, the above formulas offer a sufficient function approximation, by providing accuracy higher than 99.99% for the training values used, and therefore may be considered as near-optimal solutions.

For two (2) workstations, i.e., K=2, after ~200,000,000 iterations, and having average square error of 10^{-7}, the following formula was obtained:

$$x_2 = \frac{85M_1}{122} + \frac{\left(\frac{C_2}{6-C_2} + \frac{M_1^2}{C_2} - 2C_2\right) - \frac{C_2}{M_1}}{\left(-\frac{127M_1}{C_2} - 3C_2\right)\left(\frac{M_1}{M_2} - M_2\right) - 34}$$

Special conditions apply on this formula, like, (a) If $C_2=6 \Rightarrow C_2-6=1$ (not a realistic possibility), (b) if $C_2=0 \Rightarrow M_1^2 / C_2 = 1$ and also $(127*M_1/C_2)=1$ (division by zero), (c) the complex algebraic quantity existing on the enumerator, when divided by $C_2=0$ is also substituted by the value[1] of 1. For three (3) workstations, i.e. K=3, after 7,788,000 iterations, and having sum sq. error of –0.378532 the following formula was obtained:

$$x_3=(((1-((22-C2)/51))-((C2(M3+C3))*(M2-88*C2)))+$$

$$+(2*C3/(C3*C3-((111/((M1/C3)+((C3/(2*C3*C3))+C3)))-87))))$$

The reader should have in mind that special conditions apply on this formula too as a result of the definition of the division by zero.

For four (4) workstations, i.e. K=4, after 117009 iterations and with a sum sq. error of approximately -17.0066, the formula obtained is

$$x_4=((((((M3/(-99*C2/(79*C3+126*C4+84))/C3)-C4)/C4)-$$

$$-(C3+C4-M3+43))/(2*C2+C3-M1-M2-75))$$

For two (2), three (3) or four (4) workstations, after 87342 iterations and with a sum sq. error of approximately -25.4467 (mean accuracy again exceeds 99.99%), the formula obtained is (K denotes number of stations):

$$x_{2,3,4}= (((C2+(2*C3)+58-(2*M3))/(((M1-C2)+(102+(M4/((60/$$

$$/((115*M1)*(M1+C2)-(M2/(M4-103))))-C3))))+(((C2/M2)/((((M3+$$

$$+16)/M1)/(((C2+2*C3-2*M3+58)*(C4/M4))-(C3+11-K)))--M1))/C2)))$$

$$+((((C4/M4)/18)*(M4-M1-47))/(M1/(M4-64)-44)))$$

Other accurate formulas applied only for limited length of short production lines (K=2, 3) or referring to equal buffering, have been given in past literature for the studied problem, from [Muth & Alkaff 1986], [Muth 1984, 1987], [Blumenfeld 1990], [Hunt 1956], [Martin 1993].

Discussion and further work

The problem addressed in this work, was a symbolic regression application in the throughput rate calculation of short exponential production lines with finite intermediate buffers. This domain consisted of a highly complex search space. The applied genetic programming technique, although it achieved to find near-optimal solutions, it didn't manage yet to reveal accurate generalized formulas. The training set

[1] Formula Trials for K=2: (a) If M1=0.98, M2=0.98, C2=4 then DECO gives x = 0.84 and our formula estimates throughput rate x = 0.8411701 , (b) If M1=1, M2=1, C2=1 then DECO gives x = 0.75 and our formula estimates throughput rate x = 0.7496624

included all possible combinations for a given parameter range. Thus, the algorithm was restricted to solve only problems with 4 or less stations, while the training set was increased exponentially. For more stations, sampling techniques were proposed. However, the recent acquired formulas reached more than 99% accuracy and they can be easily implemented in a software program. These formulas have not any significant resemblance with other respective formulas obtained by previous works. This result, may be due to the fact that, for a specified training set, a large number of non-linear approximation functions exist, which would satisfy even the highest desired accuracy. Moreover, the obtained formulas often need special translations and application treatment, while the division operator is actually a modified version of the common division, known in the literature as "protected division". The authors currently focus towards a decomposed symbolic regression approach, which could possibly reveal similarities in the throughput formulas of slightly different production lines, and thus offer significant conclusions for a generalized formula.

REFERENCES

1. [Dallery & Frein, 1993] Dallery Y. and Frein Y., "On decomposition Methods for Tandem Queueing Networks with Blocking", in Operations Research, 41:(2), pp. 386-399

2. [Angeline et al. 1996] Angeline P. J. and Kenneth Kinnear, Jr. E., "Advances in Genetic Programming", Vol.2, The MIT Press, 1996

3. [Koza 1994] Koza J. R., "Genetic Programming II - Automatic Discovery of Reusable Programs", The MIT Press

4. [Koza 1992] Koza J. R., "Genetic Programming - On the Programming of Computers by Means of Natural Selection", The MIT Press

5. [Langdon 1996] Langdon W. B., "Data Structures and Genetic Programming", in P. J. Angeline and K. E. Kinnear Jr., "Advances in Genetic Programming", Vol.2, pp.395-414, The MIT Press.

6. [Martin 1993] Martin G. E., "Predictive Formulae for Unpaced Line Efficiency", Int. Jour. of Prod. Research, Vol. 31, No. 8, pp. 1981-1990.

7. [Hunt 1956] Hunt G. C., "Sequential Arrays of Waiting Lines", Operations Research, 4, pp. 674-683.

8. [Muth & Alkaff 1986] Muth E. J. and Alkaff A., "The Throughput Rate of Three-Station Production Lines, a Unifying Solution", Research Report No. 86-12, Univ. of Florida, Dept. of Industrial and Systems Engineering, Gainsville, Fla, USA. (also appeared in the Int. Jour. of Prod. Research).

9. [Muth 1984] Muth E. J., "Stochastic Processes and Their Network Representations Associated with a Production Line Queueing Model", European Journal of Operational Research, 15, No.1, Jan.1984, pp. 63-83.

10. [Muth 1987] Muth E. J. "An Update on Analytical Models of Serial Transfer Lines", Research Report No. 87-15, Univ. of Florida, Dept. of Industrial and Systems Engineering, Gainsville, Fla, USA.

11. [Blumenfeld 1990] Blumenfeld D.E., "A Simple Formula for Estimating Throughput of Serial Production Lines with Variable Processing Times and Limited Buffer Capacity", in Int. J. of Prod. Res., 28, pp. 1163-1182.

УДК 681.3:62-52

В. Дубовий, О. Глонь (Україна, Вінниця)

ПЕРЕТВОРЕННЯ НЕЧІТКИХ ДАНИХ ДИНАМІЧНОЮ СИСТЕМОЮ

Часто на етапі проектування та функціонування автоматизованої системи управління, особливо якщо до її складу входять нечіткі контролери, на її вхід подаються нечіткі сигнали, що задаються функцією належності. Для проектування та дослідження таких технічних систем необхідно розв'язати задачу знаходження фуикції належності внхідного сигналу, якщо відомі функція належності вхідного сигналу та імпульсна перехідна функція динамічного перетворювача рис.1.