A COMPARATIVE STUDY ON A SELECTION OF SEARCH DESIGN ALGORITHMS FOR SOLVING THE BUFFER ALLOCATION PROBLEM IN SERIAL PRODUCTION LINES

C.T. Papadopoulos(1, *), M.E.J. O’Kelly(2), M.I. Vidalis(3), C. Petridis(1), N. Pistofidis(1), M. Nikita(4)

(1) Department of Economics, Aristotle University of Thessaloniki, Greece. E-mails: hpap@econ.auth.gr, costas.e.petridis@gmail.com, neoklis_pistofidis@yahoo.gr
(2) Waterford Institute of Technology, Waterford, Ireland. E-mail: mejokelly@gmail.com
(3) Department of Business Administration, University of the Aegean, Greece. E-mail: mvid@ath.forthnet.gr
(4) Department of Applied Informatics, University of Macedonia, Greece. E-mail: nikita_ma@yahoo.gr

Abstract:

In the design of production lines the classical approach to the buffer allocation problem (BAP) is to use a search algorithm in association with an evaluative algorithm to obtain the mathematical optimum of the specified objective function. In practice, a choice has often to be made between which search algorithm to use for the efficient solution of the BAP. This paper gives results of a carefully selected set of experiments on short (K= number of stations = 3, 4, ..., 11 stations), medium length (K=12, 13, ..., 30 stations) and longer lines (K=40, 50, ...,100 stations) with small N (N=total number of buffer slots = K/2 if K is even; = (K-1)/2 if K is odd), medium N (N=K+1) and large N (N=2K) to evaluate the effectiveness of the following search algorithms: simulated annealing, genetic, Tabu search, myopic and complete enumeration (where possible). All the experiments were run on a readily available desktop PC with the following specifications: Windows XP Professional Version 2002 Service Pack 3, Pentium® Dual-Core CPU E5300@2.60 GHz, 2,00GB RAM. The measures of performance used are computer time and closeness to the maximum throughput achieved.

Keywords: Production lines, Buffer allocation problem, Search algorithms, Comparative efficiency.

1. The Design of Production Lines

By design of production lines is meant the specification of some of the parameters (structure of the production system) to achieve a specific objective. The design of production lines is confined mainly to the following issues: (1) Work-load at each station: There are well-known design guidelines which result in increased throughput of the line. These guidelines will specify the mean production rates at each of the work-stations. Such design problems are referred to as work-load allocation problems, WAP. (2) Determination of the number of machines at each work-station. The use of parallel systems will affect the throughput of the line. The associated design problem is referred to as the server allocation problem, SAP. (3) Specification of the sizes of the buffers, known as the buffer allocation problem, BAP.

The rest of the paper is organized as follows: In the next Sub-section the buffer allocation problem is defined. Section 2 describes the numerical experiments performed in this study and the results obtained. Finally, Section 3 outlines the ranking of the search algorithms used in this study and the main findings of the research and recommends some areas for further research.
1.1 The buffer allocation problem (BAP)

The formulation of the buffer allocation problems depends on the objective function chosen. These objective functions may be concerned with maximizing throughput, minimizing average work-in-process, or minimizing the total number of buffer slots subject in each case to appropriate constraints. In detail:

Problem BAP-A: (the dual problem) Suppose there are K machines and K-1 storage areas with N total integer (>=0) buffer slots to be allocated. A possible solution is a vector of the form: \( n = (N_1, N_2, \ldots, N_{K-1}) \).

The throughput of each solution is symbolized by \( X(n) = X(N_1, N_2, \ldots, N_{K-1}) \). The objective is to maximize the throughput of the production line subject to the constraint that the total number of buffer slots is N (Note: All buffer slots, \( N_i \), \( i = 1, 2, \ldots, K-1 \), allocated to the K-1 buffers must be integer (>=0)). The problem may be stated as follows:

\[
\max X(n) = \max X(N_1, \ldots, N_{K-1}) \\
\text{s.t.} \\
\sum_{i=1}^{K-1} N_i = N \\
N_i \geq 0 \quad \forall i = 1, \ldots, K-1
\]

Problem BAP-B: (the primal problem) The second problem is to find the minimum total number of buffer slots to be allocated among the K-1 buffers given that a pre-specified minimum throughput, \( X_0 \) is reached (Note: All buffer slots, \( N_i \), \( i = 1, 2, \ldots, K-1 \), allocated to the K-1 buffers must be integer (>=0)). The problem may be stated mathematically as follows:

\[
\min \ N = \sum_{i=1}^{K-1} N_i \\
\text{s.t.} \\
X(n) = X(N_1, \ldots, N_{K-1}) \geq X_0 \\
N_i \geq 0 \quad \forall i = 1, \ldots, K-1
\]

These two problems, the dual and the primal, are not independent and often the results obtained for the dual problem can be used to solve the primal problem.

A third problem is also used in practice which seeks to minimize the average work-in-process given that the throughput of the line exceeds a pre-specified minimum throughput, \( X_0 \). Due to space limitation, the mathematical formulation of this problem is omitted. In this study, the dual problem only was investigated.

The method of solution of the BAP follows a process, not unique to the BAP, which consists of a loop process leading to an optimal solution after a finite number of iterations. The analyst initiates the process by specifying an initial configuration of the system by giving values to the decision variables, in this case, the buffer allocation. The evaluative method determines the value of the performance objective for the system as specified. The optimization or generative method (search algorithm) takes over and presents to the evaluative method a sequence of candidate systems with new values of the decision variables. The evaluative method calculates for each system presented the value of the objective function. Evaluative methods, which estimate the objective functions of the system, are based on aggregation approaches (Lim, Meerkov and Top, 1990*) (Note: Due to lack of space all references with the * after the year of publication are not given in the references list of this paper. Instead, the reader is referred to the book by Papadopoulos et al., 2009 for further details), decomposition methods (Gershwin, 1987* and Diamantidis et al., 2007*), approximate methods, Markovian exact methods (Papadopoulos et al., 1989*) and simulation. In this study a Markovian method (where applicable, for small lines with up to K=8 stations) and a decomposition method were used as evaluative techniques to calculate the throughput of the production lines. Optimization methods, on the other
hand, which lead to the optimal or near optimal values of the decision variables and work on results of the
evaluative methods, are very varied. In this study the following five optimization methods were used: (a)
Genetic algorithms (GA), (Grefenstette, 1986*, and Papadopoulos and Karagiannis, 2001*) (b) Simulated
annealing (SA), (Spinellis and Papadopoulos, 2000*) (c) Myopic algorithms (MA), (Nikita, 2010) (d) Tabu
search algorithms (TS) (Glover and Laguna, 1998* and Pistofidis, 2010) and (e) Complete enumeration
(CE), where applicable. Due to lack of space, these algorithms are not described here. The interested reader
is referred to the fifth chapter of the book by Papadopoulos et al. (2009) where the algorithms GA, SA and
TS are described and to the Master thesis by Nikita (2010) where the myopic algorithm (MA) is fully
developed.

As usual, when comparing the performance of algorithms or any computational procedure the reader should
be mindful that there may be issues relating to the relative effectiveness of the developers in translating the
relevant flow diagrams into code and to the appropriateness of the computer system used. However, the
authors know that the codes used have been found to be very robust in obtaining solutions over a range of
serial production lines and the computer system used is readily available to designers.

2. Numerical Experiments and Results

Three (3) sets of numerical experiments (see Table 1) were carried out, involving certain categories in the
number of stations and the total amount of buffer slots to be allocated. These categories are classified
according to the number of stations in the line.

<table>
<thead>
<tr>
<th>#1</th>
<th>Small lines (K=3, 4, ..., 11)</th>
</tr>
</thead>
<tbody>
<tr>
<td>#2</td>
<td>Medium lines (K=12, 13, ..., 30)</td>
</tr>
<tr>
<td>#3</td>
<td>Large lines (K=40, 50, ..., 100)</td>
</tr>
</tbody>
</table>

Thus we consider small lines with K stations (K=3, 4, ..., 11), medium lines with K stations (K=12, 13, ..., 30)
and large lines with K stations (K=40, 50, ..., 100). In all cases, reliable, exponential and balanced lines were
considered, i.e., with equal mean service rates: \( \mu_i = \mu = 1 \). All the experiments were run on a readily
available desktop PC with the following specifications: Windows XP Professional Version 2002 Service
Pack 3, Pentium® Dual-Core CPU E5300@2.60 GHz, 2.00GB RAM. The measures of performance used
were computer time and closeness to the maximum throughput achieved. These experiments were carried out
using as evaluative algorithms: (1) the Markovian algorithm for up to K=8 stations and (2) the
decomposition algorithm for all stations (from K=3 up to 100 stations) and as search algorithms the five
algorithms: GA, SA, MA, TS and CE (the latter only for small lines with up to K=10 stations). In general, it
has been observed that Complete Enumeration (CE) provides better results when combined with the
Markovian evaluative algorithm than when it is used with the decomposition algorithm (DECO). For the
cases: K=3, 4, ..., 8, where a comparison can be made among the two evaluative algorithms, the Markovian
algorithm gives higher throughput than the decomposition algorithm.

In each of the sets of experiments, the total number of buffer slots, N, were allocated among the K-1 buffers
of the K-station production line as follows:

- Small number of total buffer slots: \( N < \frac{K}{2} \) if \( K \) is even; \( \frac{(K-1)}{2} \) if \( K \) is odd;
- Medium number of total buffer slots: \( N = \frac{K}{2} + 1 \);
- Large number of total buffer slots: \( N = 2K \).

Thus a total of 105 different lines were analyzed. The throughputs and the CPU times of each line in the
three sets of experiments were obtained using the five search algorithms: GA, SA, MA, TS and CE (the latter
where applicable) and are given in Figure 1.
Figure 1: Throughput of Set of Experiments #1 (A), Set of Experiments #2 (B), Set of Experiments #3 (C), and CPU Time of Set of Experiments #1 (D), Set of Experiments #2 (E), Set of Experiments #3 (F).
3. Ranking of the Search Algorithms, Findings of the Study and Further Research

In this work, a comparison among the search algorithms: GA, SA, MA, TS and CE (the latter where applicable) was carried out, based on the maximum throughput and the minimum CPU time of execution. The methodology followed was to rank each algorithm against all others for each case in the three sets of experiments given in Table 1. The following two ranking procedures were used:

A simple ranking scheme, viz., 1, 2 up to 4 or 5 depending on the number of search algorithms being ranked and with a rank of 1 being considered the best value. In effect there are a total of either 10 or 15 weights to be allocated over the 4 or 5 algorithms, respectively. If ties occur, the total of the appropriate weights is distributed evenly between the tied algorithms, e.g., if three algorithms tied for third place in the ranking of 5 algorithms, each of the tied algorithms would receive a rank (weight) of 4 and in this case no algorithm would receive a rank of either 3 or 5. When all of the algorithms used in each line in a particular set of experiments are ranked, the average rank of each algorithm over the set is determined and the rank of these averages are reported as given in Table 2 and Table 3 using the simple ranking scheme just described, with the lowest average rank being considered best and given a rank of 1.

A % absolute deviation ranking scheme, whereby the percentage absolute deviation of the value obtained by each algorithm from the best value obtained by any algorithm for each line in a particular set of experiments is determined, i.e., % absolute deviation is equal to 100(X - Xbest)/Xbest where X is the value achieved by the algorithm being ranked and Xbest is the best value achieved by any algorithm in respect of a particular production line. The symbol indicates absolute value to allow for the two cases, viz., ranking of throughput where Xbest is a maximum and the ranking of CPU time where Xbest is a minimum. The average value plus three times the standard deviation of the % absolute deviations associated with each algorithm over a particular set of experiments is determined and finally these statistics are ranked again using the simple ranking scheme and are given in Table 2 and Table 3.

<table>
<thead>
<tr>
<th>Table 2 Ranking of all algorithms of Set of Experiments #1, #2, #3 based on the maximum throughput.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Set of Experiments #1</td>
</tr>
<tr>
<td>----------------------</td>
</tr>
<tr>
<td>Average Ranking</td>
</tr>
<tr>
<td>GA</td>
</tr>
<tr>
<td>SA</td>
</tr>
<tr>
<td>MA</td>
</tr>
<tr>
<td>TS</td>
</tr>
<tr>
<td>CE</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Table 3 Ranking of all algorithms of Set of Experiments #1, #2, #3 based on the CPU time.(Cases where CPU time was zero were excluded from the calculation of (μ+3σ)% absolute deviation from minimum)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Set of Experiments #1</td>
</tr>
<tr>
<td>----------------------</td>
</tr>
<tr>
<td>Average Ranking</td>
</tr>
<tr>
<td>GA</td>
</tr>
<tr>
<td>SA</td>
</tr>
<tr>
<td>MA</td>
</tr>
<tr>
<td>TS</td>
</tr>
<tr>
<td>CE</td>
</tr>
</tbody>
</table>
CE crashed for medium and large lines. In the third set of experiments (K=40, 50,...,100) apart from CE, the following algorithms: SA, MA and TS failed to converge in some cases.

The first set of experiments (for lines with K=3, 4,...,11) showed that CE is the algorithm that provides the maximum throughput as was expected. SA gives the second higher throughput and in many cases the same throughput as CE at a CPU time greater than that required by GA. SA is the slowest algorithm while MA is the fastest algorithm for set #1 of experiments.

In general, for the set #2 (for lines with K=12, 13,...,30) and the set #3 (for lines with K=40, 50,...,100) the following observations may be made:
(i) In set #2 of experiments, SA provides a near optimal solution (SA was ranked first regarding the throughput), however, it is the slowest algorithm. In set #3 of experiments, SA fails to converge in some lines. In those cases, MA provides the highest throughput.
(ii) GA was ranked in the last place regarding throughput and third in regard to the CPU time.
(iii) MA provides the best throughput after SA and overall is the fastest of all the search algorithms examined. Both GA and MA proved to be reliable as they provided a solution in the cases where SA and TS failed and in all cases examined in this study.
(iv) In set #2 of experiments, TS was ranked in the third place out of the four algorithms examined, but is one of the fastest algorithms as it was ranked in the second place regarding CPU time. In more detail: TS provides the lowest throughput for the case N<<K (N=K/2 if K is even; = (K-1)/2 if K is odd). For the same cases, MA is slower than GA. In the case N=K+1, TS provides a higher value of throughput than GA and for the case N=2K, TS gives a better solution than MA and GA. In the latter two cases MA improves its convergence speed. In set #3 of experiments, TS also fails to converge in some cases.

An area for further research might be the investigation of other efficient search algorithms such as the gradient algorithm (Gershwin and Schor, 2000*) and its use in conjunction with smart segmentation techniques as the one employed by Shi and Gershwin (2011).

References


