Proceedings of the 2008 Winter Simulation Conference
S. J. Mason, R. R. Hill, L. Mönch, O. Rose, T. Jefferson, J. W. Fowler eds.

A COMPARATIVE STUDY OF GENETIC ALGORITHM COMPONENTS IN SIMULATION-BASED OPTIMISATION

Birkan Can
Enterprise Research Center
University of Limerick
Limerick, IRELAND.

Andreas Beham
Research Center
Upper Austria University of Applied Sciences
Softwarepark 11, Hagenberg, AUSTRIA.

Cathal Heavey
Enterprise Research Center
University of Limerick
Limerick, IRELAND.

ABSTRACT

In this paper, we present a comparative study of different stochastic components of genetic algorithms for simulation-based optimisation of the buffer allocation problem. We explore the effects of elements such as operators, fitness assignment strategies and elitism. Three different recombination operators, incorporated with constraint handling mechanisms such as repair and penalty functions, are examined. Under the shed of the experiments, we incorporate problem specific knowledge to further enhance the practicality of GA in decision making for buffer allocation problem.

1 INTRODUCTION

The increasing body of literature in the field of simulation-based optimisation reveals the importance of moving beyond the use of simulation models as simply explorative tools for alternative system designs in an heuristic fashion (Belton et al. 2004). This area is now the focus of much research and development in both academia and practice.

The buffer allocation problem (BAP) is a combinatorial optimisation problem which can be observed in many manufacturing and supply chain systems (Dolgui et al. 2007). BAP is mainly concerned with improving the throughput rate in a production line by exploring alternative buffer allocation schemes.

BAP belongs to the class of NP-hard combinatorial problems such as knapsack and travelling-sales man problems (Huang et al. 2002). For such problems, there exists no algorithm to solve them in polynomial time. Turchyn (2007) identifies that complete methods which guarantee an optimal solution for NP-hard problems may require exponential computation time. In most of the real-world problems, the decision variable space can be large and solution evaluation durations may be long depending on the methodology. These issues essentially render complete methods computationally inefficient for complex problems. Simulation-based optimisation is a good example of such problems owing to the complexities such as uncertainties, size of the system modelled and number of the decision variables. These difficulties may require a sacrifice from solution quality and promote approximate optimisation methods, instead.

Simulation-based optimisation has been a fruitful domain considering the approximate optimisation techniques, such as stochastic approximation (L'Ecuyer and Yin 1998), random search (Andradttir 2006), meta-heuristics (Lutz et al. 1998, Ahmed 2007, Chang and Feng-sheng 1994). Detailed reviews of such techniques can be found in Tekin and Sabuncuoglu (2004) and Fu (2002); an empirical comparison of some search algorithms can be found in Lacksonen (2001).

Meta-heuristic algorithms are among these approximate techniques which can be used to solve complex problems. Simulated annealing (SA) imitates the annealing process in crystalline solids, whereas tabu search (TS) exploits the memory structure in living beings. Similarly, genetic algorithms (GAs) emulate the evolutionary process in nature.

GAs, a sub-branch of evolutionary algorithms (EAs), are developed to solve difficult optimisation problems. A key property of GAs is the existence of a population during the search procedure. This allows multiple point iterations through out the optimisation in contrast with other point-to-point meta-heuristics. With the additional flexibility of GA components, they have been applied in various problem domains. Oduguwa et al. (2005) overview the applications of evolutionary algorithms in various sectors of manufacturing industry. Similarly, Chaudhry and Luo (2005) provide an extensive review of applications of GAs in production and management operations. Finally, a technical review of genetic algorithms can be found in Srinivas and Patnaik (1994).

In this study, we will exploit GAs to perform the optimisation of BAP via different implementations of its components. First we try to identify the best performing GA operators on different sizes of BAPs. Subsequently, we introduce problem specific knowledge to the process to improve the algorithm performance. The remainder of this paper is organized as follows. In the following sec-

978-1-4244-2708-6/08/$25.00 ©2008 IEEE

tion, the literature review accompanied with the problem description is given. In section 3, we illustrate the operators implemented and present the results of the experiments performed. Section 3.5 presents results with exploitation of the best operators found and exemplifies a GA implementation, where practical operation of the algorithm is tried to be improved. Discussions and conclusion finalise the article.

2 LITERATURE REVIEW

2.1 Buffer Allocation Problem

Efficiency, e.g. utilisation of resources, has been as important a concept in manufacturing as flexibility and adaptability in manufacturing systems (Bordoloi et al. 1999). Buffer allocation is commonly observed as a utilisation of available storage space in many production systems. Similar problems can also appear in most supply chains operations, communication technologies and so on (Dolgui et al. 2007). Therefore, BAP has received a significant attention in research and practice, particularly in production and operations management. Moreover, its combinatorial nature and the existence of inherent stochastic elements, such as machine failures, processing times, in real-world manufacturing make the BAP more challenging, thus appealing.

Conway et al. (1988) identify roles of buffers in production systems. To illustrate, using intermediate buffers can help improving production rate and smooth operation of a manufacturing facility by eliminating disruptive effects of possible stochastic elements such as processing times and failures (Harris and Powell 1999). However, increase in capacity can lead to increase in work-in-process (WIP) inventory. This will essentially result in more space requirements for storage at intermediate buffers. Hence, when the gain over the improvement of production rate is no more desirable compared to the costs incurred owing to WIP and storage needs, buffer allocation problems arise.

In this study, we are interested in maximisation of expected throughput rate of a serial production line via buffer allocation. The problem setup is as shown in Figure 1.

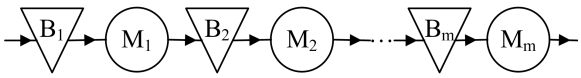

Figure 1: Serial production line.

The serial production line is composed of serial workstations consisted of a single machine (server), M_i, and a finite buffer, B_i of size q_i, as shown in Figure 1, for m-station serial line. Each reliable machine has exponential processing times with $\mu_i = 1$ in order to reflect the variabilities in the process, such as failures. Jobs arrive at the system at the first buffer and sequentially proceed through the line in first-come-first-served manner. As a result of having finite buffers and variability in processing times, this scheme of processing may cause starvation and blocking at intermediate stages. When a machine is free to process, it takes a job from its upstream buffer. If there is no job to process in its upstream buffer, then the machine is said to be starved. Conversely, blocking in production lines can be commonly observed in two different ways, communication and production blocking (Papadopoulos et al. 1993, pg. 91). In communication blocking, the machine does not start processing a job from its upstream buffer if there is no space in the downstream buffer for that job to be stored. Therefore, it is also known as blocking-before service. Alternatively, in production blocking, if the machine is free to process a job, it is taken from the upstream buffer. However, after completion of the processing, the job can not be passed on to the downstream buffer if there is no space available, hence the machine gets blocked-after-service. In our model, we assume that only production blocking occurs and the first machine is never starved and the last machine is never blocked. The DES model is verified and validated with different buffer allocation schemes accessible in Papadopoulos et al. (1989).

2.2 Optimisation of Production Lines

Throughput rate, work-in-process, total buffer allocation, customer service levels can be considered as popular performance measures in manufacturing, particularly in queuing systems. Such measures become crucial to drive with evaluative tools to assess the system performance against its configurations. Moreover, these evaluative tools can assist the decision making when integrated with optimisation routines in order to identify optimal (or better) alternative designs for system improvement.

The generative techniques generally applied in optimisation of production lines via buffer allocation can be loosely sub-categorised into two. Over the last decades, many strategies and heuristics that characterise the buffer allocation schemes on different type of problem settings have been introduced to literature (Hillier 2000, Chan and Ng 2002). Alternatively, optimisation routines, which consider the problem as a 'black-box', can be used in order to automatize and attain a more generally applicable allocation procedure (Lutz et al. 1998, Spinellis et al. 2000, Yamashita and Altiok 1998). Indeed, a third category can be branched from the combination of optimisation procedures and allocation strategies (Dolgui et al. 2007).

Optimisation in a 'black-box' fashion will essentially require incorporation of generative techniques to system evaluation tools. These tools can be regarded as a descriptive component, or a black-box, providing performance of a system against a prescribed set of decision variables by the generative method. Gershwin and Schor (2000) review the

evaluation models of production lines and a detailed study in performance analysis of manufacturing systems can be found in Altiok (1997). There have been several applications of heuristic decomposition techniques (Asadathorn and Chao 1999), exact numerical algorithms (Heavey et al. 1993, Papadopoulos et al. 1989). Even though analytical models make the evaluation of the systems very cost efficient in terms of evaluation time, they suffer from the fact that they are approximate models and unrealistic assumptions may also be necessary to make the problem more tractable (Altiparmak et al. 2007). Alternatively, discrete-event simulation (DES) can provide the sufficient descriptive environment to include desired level of details in the most complex systems. Moreover, reusability of the model components for different problem topologies as well as transparency can render DES more advantageous. Bulgak (2006) and Harris and Powell (1999) give examples of DES used as evaluative tools in optimisation of BAP.

The descriptiveness of DES brings up a disadvantage over its analytical counterparts because of the time required to evaluate candidate solutions provided by generative algorithms. This can majorly stem from the introduction of stochastic elements (e.g. processing times etc.) of the real-world problems. When such properties are incorporated into a simulation model, one needs to use specific output analysis techniques which eventually require multiple or longer simulation runs at steady state. Subsequently, expected average performance of a solution can be attained. Alexopoulos (2006) provides a comprehensive review of statistical techniques used in simulation output analysis. In our study, we have used batch-means analysis in steady-state simulation. We perform 30 batch collections to attain the expected throughput rate and use it as the performance for a corresponding solution.

Since the expected throughput rate is considered as the performance criterion, the problem can be formulated as in Equation 1.

$$\left.\begin{array}{ll} \text{maximise} & E[T(Q,\omega)] \\ \text{subject to} & \sum_{2}^{m} q_i \leq N, \\ & q_i \geq 0 \quad \forall i \in [2,m] \end{array}\right\} \tag{1}$$

The objective function in Equation 1, $E[T(Q,\omega)]$ corresponds to the expected throughput rate for a buffer allocation scheme, $Q = (q_2, q_3, ..., q_m)$. The parameter, ω represents the stochastic effects (noise) in the function. The constraint limits the buffer allocation to be less than an allowed size, N. In this setup, the optimisation technique should efficiently explore a large decision space as given by Equation 2 for a serial production line buffer allocation problem (Lutz et al. 1998).

$$C\begin{pmatrix} N+m-2 \\ m-2 \end{pmatrix} \tag{2}$$

The literature review indicates that the formulation given in Equation 1, i.e. maximisation of throughput rate given a maximum available buffer space, dominates the studies dealing with optimisation of BAP. Alternatively, a dual problem can also be formed considering the minimisation of total buffer allocation provided that a minimum throughput rate will be satisfied (Yamashita and Altiok 1998, Gershwin and Schor 2000). Spinellis et al. (2000) outline the main decision variables in optimisation of production lines as; buffer, server and work allocation. Buffer allocation problems share the most substantial majority of the studies owing to its effects on the efficiency of production lines as illustrated by Conway et al. (1988).

2.3 Optimisation via Genetic Algorithms

There has been a variety of optimisation techniques integrated with DES models of production lines for BAP. In an early simulation-based study, Bulgak et al. (1995) apply the naive genetic algorithm to maximise average production rate of asynchronous assembly systems via buffer allocation. Harris and Powell (1999) utilizes an adaptation of Spendley-Hext and Nelder-Mead simplex search algorithms to solve buffer allocation in production lines, where servers have different processing times, i.e. unbalanced. Lutz et al. (1998) propose a simulation-search procedure based on TS. Similarly, Spinellis et al. (2000) investigates SA and a basic GA, with decomposition algorithm as an evaluation tool. They conclude that SA converges to a solution faster than GA for shorter lines, where as GA outperforms SA in larger problems in terms of solution quality and convergence rate.

Genetic algorithms (GAs), a sub-branch of EAs, are developed to solve difficult optimisation problems. In general, there are some components which need to be implemented for EAs to work. Initially, a representation, similar to DNA in living beings, is required to define (encode) a candidate solution. When the BAP is considered, an allocation scheme, Q, can be encoded into an integer array which represents individuals. Therefore, each gene of an individual will be decoding q_i for the corresponding buffer location (See Equation 1). Generally, these individuals are randomly generated and they form a population of solutions, which distinguishes EAs from other point-to-point optimisation algorithms, such as SA and TS. Each individual possesses a quality value which corresponds to simulation performance in our context. This quality, i.e. fitness, determines its relative ability to survive and reproduce. Furthermore, there are stochastic operators in EAs such as recombination, mutation and selection which are sequentially applied

to individuals through the evolutionary cycle. The process can be illustrated as in the Algorithm 1.

input : Population size;
maximum generation, t_{max};
operator probabilities: P_c, P_m.
output: Return best individual found
begin
  $0 \longleftarrow t$;
  Generate population, $P(t)$;
  Evaluate individuals in P(t);
  while *Stopping criterion not satisfied* **do**
    Perform selection of parents from $P(t)$;
    Generate offspring via crossover with P_c;
    Apply mutation to offspring, with P_m;
    Insert offspring into $P(t+1)$;
    Repeat 6 to 9 $P(t+1)$ is full;
    Replace $P(t)$ with $P(t+1)$;
  end
end

Algorithm 1: Evolutionary algorithm

Briefly, selection operators provide individuals to the mating pool which are to undergo recombination. Recombination operators perform exploration of the search space by means of data exchange between individuals. This process resembles the reproduction in creatures and yields new individuals (offspring). Mutation, a variation process, provides genetic diversity and exploitation of a selected individual by causing slight changes. A key feature accompanied with the existence of stochastic elements in EA is that even a weak individual can find chance to reproduce or survive. This further enhances EAs in terms of avoiding local optima in complex search landscapes.

Considering the evolutionary cycle (Algorithm 1) and the operators briefly explained in the above paragraph, the very abstract description of EAs has lead to various type of algorithms, such as evolution strategies, evolutionary programming and genetic algorithms. In the following section, we will introduce designs of key components of GAs to solve BAP and provide the experimental results subsequently.

3 EXPERIMENTS AND RESULTS

3.1 Implementation of Genetic Algorithms

Recent advances in software for both DES and optimisation have enabled researchers attain flexible and extensive simulation optimisation tools. Especially with the available software environments, e.g. Microsoft .Net, algorithm development has become easier. Moreover, many tasks requiring graphical interfaces can now be performed with less programming required compared to former programming languages.

A review of optimisation domain reveals that the algorithm implementations in C++ dominates the field. Alternatively, we used a framework built in Visual C#, HeuristicLab 1.1, accessible via <www.heuristiclab.com>. HeuristicLab, which is easy to modify and extend, gives a framework which provides necessary interactions between operations throughout the evolutionary cycle, graphical interaction and easy control to the user. A plug-in housing the operators is coded in C# and integrated into the main framework (available on <www.heuristiclab.com>). The DES tool, eM-Plant 7.5 (<www.emplant.com>) is used to model the problem. The communication with simulation is established through the COM library of eM-Plant directly without an intermediate database to increase efficiency of the interaction between simulation and optimisation modules.

In all the experiments, an elitist-model GA is used. Elitism guarantees survival of the best solution by substitution of it to the next generation. Elsewhere (Rudolph 1996), elitist GAs have been shown to converge to global optimum. Recombination and mutation are stochastic operators in the sense that selected individuals will undergo these operations with a predefined probability.

We have tested a single mutation operator of type swap mutation. The individual to undergo mutation is divided into 3 random parts and the first and the third parts are swapped. This type of move provides exploration of alternative allocation schemes of a candidate, which may be considered as a local optimisation around a total allocation level. Primary recombination operators implemented are average (AXO), convex (CXO) and discrete (DXO) crossovers (Gen and Cheng 1997).

Because of the buffer constraint, CXO and DXO may yield infeasible solutions during the search. Such solutions are handled via different evaluation schemes including penalty and repair mechanisms during the fitness assignments. Conversely, AXO operates only in the feasible solution space. To illustrate, *CXORep* refers to a modified implementation of CXO where the infeasible solutions are repaired, whereas penalty function is applied the infeasible solutions generated by CXO and DXO during fitness assignment. The repair operation in *CXORep* is performed as shown in the Algorithm 2. Each time an offspring receives a gene (allocation to a buffer position) the feasibility is checked and allocation is allowed accordingly, otherwise the corresponding gene receives a zero.

Alternatively, the penalty approach can also be used to handle the constraints. The penalty mechanism preliminarily used in Experiments I is the reciprocal of the constraint violation, i.e. capacity divided by total allocation. In Experiment II, results for different penalty strategies are presented.

input : Receive parents; X^t and Y^t.
output: Return offspring, X^{t+1}.
serve, occupy $\leftarrow 0$;
Generate random numbers; $r \in [0,1]$, $\alpha \in [0,1]$;
if $r \leq p_c$ **then**
  for $i \leftarrow 0$ **to** l **do**
    serve = $\alpha * X_i^t + (1-\alpha) * Y_i^t$;
    occupy = occupy+serve;
    if *occupy* $\leq$ *capacity* **then**
      X_i^{t+1} =serve;
    else
      $X_i^{t+1} = 0$;
      occupy = occupy-serve;
    end
  end
end

Algorithm 2: CXORep crossover

Since EAs are stochastic in nature, the experiments need to be repeated several times to provide a confidence in the achievements of the algorithms. In our experiments, we perform 10 independent experiments for each configuration simulated and average the total number of evaluated solutions in tests when an optimal or sub-optimal solution in the vicinity of 1% of the best quality is reached. And this averaged evaluation count is considered as the performance criteria. The problems are labelled as XsYb, where X refers to station number and Y is the total available buffer space.

3.2 Experiment I – Identification of the Best Components

Three different selection operators in GAs; roulette wheel, linear ranking and tournament selection are experimented. More details on these mechanisms can be found in (Goldberg and Deb 1990). These operators are coupled with different recombination operators to solve a 9s8b problem. GA is started up with the settings; population size, 40, $P_c = 0.5$ and $P_m = 0.05$. The optimum solution for this problem settings is known, where each intermediate buffer receives '1-unit' storage capacity, and maximum average expected throughput rate is 0.5663.

Table 1: Experiments on randomly chosen selection & recombination couples.

Exp. No	Selection	Crossover	Count	E[T(Q)]
1	Roulette	AXO	217	0.5477
2	Linear Rank	CXORep	288	0.5313
3	Tournament	DXO	369	0.5663

The comparative results are given in Table 1. In the first and the second experiments, GA prematurely converged to non-optimal values in all tests whereas in the third experiment GA was 90% successful at locating the optimum as shown in Figure 2.

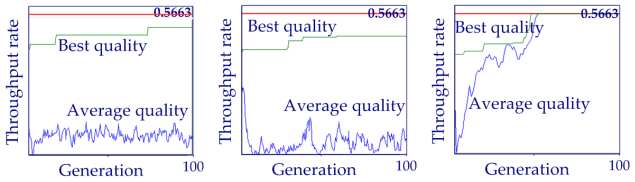

Figure 2: Typical single run charts representing the behaviour of GAs in Table 1: 1 (left), 2 (centre), 3 (right).

As tournament selection and DXO outperformed the other settings shown above, they are independently tested with different components, subsequently (See Tables 2 & 3). The findings imply that allowing the formation of infeasible solutions with a penalty strategy, particularly in DXO, outperforms when such solutions are repaired instead.

Table 2: Experiment results for Tournament Selection with different crossover operators. Results still indicate superiority of DXO.

Selection	Crossover	Evaluation Count
Tournament	AXO	474
Tournament	CXORep	542
Tournament	CXO	422
Tournament	DXO	369

Table 3: Results for DXO with other selection operations.

Selection	Crossover	Evaluation Count	E[T(Q)]
Roulette	DXO	964	0.5663
Linear Rank	DXO	415	0.5663
Tournament	DXO	369	0.5663

It is worth mentioning that there exists an optimum population size which is dependent on the size of the corresponding search space defined by the amount of decision variables and constraints of a problem (See Equation 2). Table 4 shows how the population size effects the performance of GA in terms of the total number of evaluated solutions during the search in the 9s8b problem. It has been empirically observed that below a certain population size, there is more probability for GA to prematurely converge to a non-optimum solution owing to loss of gene diversity. Conversely, when the population size is increased, the search space may be unnecessarily explored. The third column in Table 4 summarizes the percentage of convergence to optimum or near optimum solutions in 10 experiments.

Table 4: Effects of population sizes on the performance.

Population size	Evaluation Count	%Convergence
20	194	60
40	369	90
80	677	90

3.3 Experiment II – Penalty Strategies

After identification of successful components, we present results for different penalty strategies, subsequently. We compare three different penalty strategies on a 15s30b problem. The first strategy is already utilized in Section 3.2, with results given in Tables 1 to 3 for tournament selection and DXO in 9s8b problem. In the second strategy, the infeasible solutions are given death penalty by assigning zero fitness. Alternatively, an extension to the death penalty is made as a third strategy in which the infeasible individuals are penalised as in the first strategy if deviation form the total buffer capacity exceeds one; otherwise they undergo the death penalty. Such modification can be helpful to avoid the loss of genes in competitive candidates. Table 5 shows the results for these strategies for 9s8b and 15s30b problems, respectively.

Table 5: Average evaluation count results for 9s8b & 15s30b problems.

Strategy	9s8b	15s30b
1	369	3127
2	346	3339
3	381	4022

The results for the 9s8b problem have implied that death penalty approach has yielded better results, although more experiments would be needed to be able to say it is clearly better. The ultimate advantage of such a penalty is the avoidance from evaluating infeasible solutions. However, it can have negative effects on the population diversity for the proceeding iterations due to loss of genes in the individuals with infeasible allocations and can lead to premature convergence. To cope with the diversity loss, a larger population and a higher mutation rate can be used with the cost of an increase in the evaluation count. A larger initial population essentially has more chance to provide a richer gene pool and increased mutation rate will cause small variations on the individuals at the further stages of the evolution. Hence, the population size and mutation rate are increased to 60 and $P_m = 0.1$, respectively, for the 15s30b problem since it has a larger search space.

3.4 Experiment III – Investigating the Effect of Elitism

Elitism refers to preservation of a number of the best performing individuals from a generation to the following generation. Elitist GAs have been formerly shown to converge to global optimum in general search spaces (Rudolph 1996). Convergence behaviour in stochastic problems is further illustrated by Allen et al. (2002). In its naive sense, selecting one or more elite solutions throughout the iteration resembles selecting a subset of individuals from which system comparisons can be made. Since the simulation output is stochastic, one can say a solution is better than the other probabilistically relying on the confidence interval. While enlarging elitist subset can help to improve convergence rate by decreasing the number of evaluated solutions, it can amplify the selection pressure, hence may result in premature convergence. The Figure 3 shows how this is observed in the 15s30b problem.

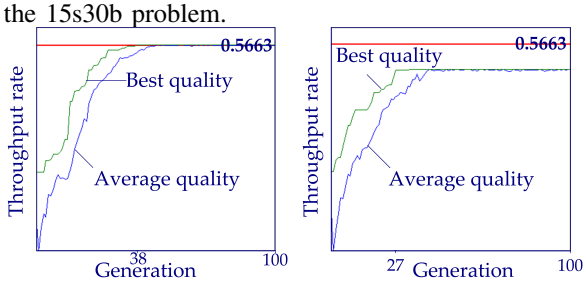

Figure 3: A typical single run chart representing the behaviour of convergence of GA with elitist selection 2 (left) and 4 (right).

Experimental results in Table 6 indicate that selection of an elitist subset improves the algorithm convergence rate dramatically when the size of the subset is not very large to result in premature convergence. Table 6 presents the evaluation count results for the 15s30b problem with different number of elitism employed. The crossover used is DXO with $P_c = 0.5$ and selection operator is tournament selection in all tests. The settings for the case where elitism is one, the population size is 60, and $P_m = 0.1$ for convergence performance over 10 experiments. The population size is 40 and $P_m = 0.05$ for the case where elitism is two or four. Results imply the improvements in terms of average number of evaluated solutions over 10 experiments where the deviation from the optimum throughput rate is less than 1%.

During the experiments shown in the table above, GA with 4 elitist solutions was outperforming in terms of evaluation count in the experiments. However, the setup where elitism is 2 was more preferable because of being 80% successful in locating the global optimum in 10 runs, whereas it was 2 out of 10 times for the Experiment 3.

Table 6: Elitism vs. Evaluation count tests for the 15s30b problem.

Exp. No	No.of Elites	Evaluation count	$E[T(Q)]$
1	1	2817	0.6352
2	2	1212	0.6352
3	4	940	0.6352

3.5 Sensitivity of Performance against Total Buffer Size

In the experiments up to this point, problem specific knowledge is not used in GA implementations in order to keep the solver's independence from the problem topology. Nonetheless, success of a certain operator, e.g. DXO, implies its suitability for the specific problem.

Although genetic algorithms have performed satisfactorily in Experiments I to III, locating optima can still be perceived as impractical in operational sense. While this weakness mainly stems from the long model execution times, it can be overcome by introduction of problem specific knowledge to the algorithm. Moreover, 'no free lunch' theorem implies the advantage of integrating domain knowledge to the algorithms (Wolpert and Macready 1997).

BAP literature offers many corollaries characterising successful allocation schemes (Hillier and So 1995, Hillier 2000, Chan and Ng 2002). Since we have studied a balanced asynchronous serial production line with finite buffers, it can be helpful to study integration of related corollaries within the algorithm. Hillier (2000) reflects on the characteristics of buffer allocation on balanced lines. During the simulation validation process, it is observed that some of these specifications did not hold for our model. Therefore, we evaluate some design points and try to identify a lower limit on total buffer allocation below which the performance is assumed to be no more appealing to a decision maker.

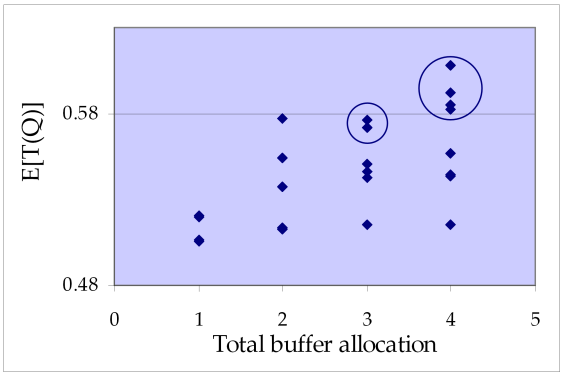

Figure 4: Alternative allocations for 5s4b problem. The possible promising regions are encircled.

In Figure 4, we consider a 5s4b problem and first try to determine a threshold allocation level for allocation vs. throughput rate which will be uniformly distributed through the line. This dictates GA to perform the global optimisation over an attractive region of the search space, rather than the whole domain. Owing to the concave increasing relationship between the throughput rate and total buffer allocation (Hillier et al. 1993), a lower limit is induced to the constraint this way in providing the initial population to GA. Table 7 shows the resultant evaluation costs and indicates operationality of such an approach in different size of problems. In these problems, the recombination operator, DXO with $P_c = 0.5$, is incorporated with the penalty mechanisms from Section 3.3. Similar to previous setups, tournament selection and Swap2Segments mutation, with $P_m = 0.05$, are preferred. Since a smaller portion of the decision space is to be explored, smaller population sizes are used depending on the size of the decision space of the problem. In all of the 10 runs of each experimental setup, GAs located global optimum or near-optimal solutions with less than 0.4% deviation from the maximum expected throughput rate.

Table 7: Results after incorporation of problem-specific knowledge to GA.

Problem	Pop. Size	Penalty type	Elitism	Count
9s8b	20	(2)	2	199
15s20b	20	(2)	2	242
15s25b	20	(2)	2	356
15s27b	30	(3)	2	423

4 DISCUSSION & CONCLUSION

The buffer allocation problem represents various type of problems in the manufacturing industry. Owing to its NP-hardness and high number of alternative solutions, solving BAP requires algorithms which can explore a large discrete solution space and discover the promising areas. We have utilised genetic algorithms to perform this task and further enhance it via knowledge-based modifications.

In this work, we performed an empirical comparison of different recombination mechanisms as well as constraint handling techniques. Our results have shown that allowing the formation of infeasible solutions and incorporation of linear penalty functions during the search exhibit a better performance. We have also presented how the size of the elitist subset influence the GA performance in solving BAP. This illustration can shed some light to our future GA implementations when the noise in the evaluation is considered. Finally, inclusion of domain-specific knowledge in the GA implementation has dramatically improved the algorithm performance. This way, GA is driven to exploit a

desirable region. It would be interesting to divide the overall search space to such sections and investigate the progress of the parallel GAs which can be helpful in a multi-objective study in approximating to the pareto-frontier.

Another observation made during the experiments is degeneracy of the population. In a population at each iteration, when the fitness values of different individuals are very close to each other despite their genetic diversity, the population is then called as degenerate population which tends to prematurely settle at a sub-optimum solution. The empirical tests have revealed the degeneracy in BAP. This also implies that BAP has a decision space with large number of local optima surrounding the global optimum with slight differences in performance value. This fact further complicates the problem since it becomes more difficult to reach the global maximum. Therefore, while using GAs for optimisation, one may also consider the methods to tackle the problem further when the population is degenerated. In our experiments, we have only tested a single mutation type, Swap2Segments. Since small genetic variations and diversity can be provided by mutation, a new operator suitable for the problem structure can be introduced to restrain from the degeneracy. Alternatively, performing a response surface study over the degenerate population may yield the landscape character of the promising areas of the search space.

In the future, we would like to exploit these outcomes in order to extend our work to multi-objective optimisation of simulation models of similar systems such as supply chains.

REFERENCES

Ahmed, M. A. 2007. A modification of the simulated annealing algorithm for discrete stochastic optimization. *Engineering Optimization* 39 (6): 701–714.

Alexopoulos, C. 2006. A comprehensive review of methods for simulation output analysis. In *Proceedings of the 2006 Winter Simulation Conference, Vols 1-5*, 168–178. New York: Ieee.

Allen, T., W. Ittiwattana, and M. Bernshteyn. 2002. *An elitist genetic algorithm incorporating sequential subset selection*. Ph. D. thesis, The Ohio State University.

Altiok, T. 1997. *Performance analysis of manufacturing systems*. New York: Springer-Verlag.

Altiparmak, F., B. Dengiz, and A. A. Bulgak. 2007. Buffer allocation and performance modeling in asynchronous assembly system operations: An artificial neural network metamodeling approach. *Applied Soft Computing* 7 (3): 946–956.

Andradttir, S. 2006. Simulation optimization with countably infinite feasible regions: efficiency and convergence. *Acm Transactions on Modeling and Computer Simulation* 16 (4): 357–374.

Asadathorn, N., and X. Chao. 1999. A decomposition approximation for assemblydisassembly queueing networkswith finite buffer and blocking. *Annals of Operations Research* 87 (0): 247–261.

Belton, V., J. Hodgkin, and G. Montibeller. 2004. From design to decision: An integrated approach linking discrete event simulation, evolutionary multi-objective optimisation and multicriteria decision analysis. In *17th International Conference on Multiple Criteria Decision Analysis*. Whistler, Canada.

Bordoloi, S. K., W. Cooper, and H. Matsuo. 1999. Flexibility, adaptability, and efficiency in manufacturing systems. *Production and Operations Management Society* 8 (2): 133–150.

Bulgak, A. A. 2006. Analysis and design of split and merge unpaced assembly systems by metamodelling and stochastic search. *International Journal of Production Research* 44:4067–4080.

Bulgak, A. A., P. D. Diwan, and B. Inozu. 1995. Buffer size optimization in asynchronous assembly systems using genetic algorithms. *Computers and Industrial Engineering* 28:309–322.

Chan, F. T. S., and E. Y. H. Ng. 2002. Comparative evaluations of buffer allocation strategies in a serial production line. *International Journal of Advanced Manufacturing Technology* 19 (11): 789–800.

Chang, L., and T. Feng-sheng. 1994. Buffer allocation via the genetic algorithm.

Chaudhry, S. S., and W. Luo. 2005. Application of genetic algorithms in production and operations management: a review. *International Journal of Production Research* 43:4083–4101.

Conway, R., W. Maxwell, J. O. McClain, and L. J. Thomas. 1988. The role of work-in progress inventory in serial production lines. *Operations Research* 36 (2): 229.

Dolgui, A., A. Eremeev, and V. Sigaev. 2007. Hbba: hybrid algorithm for buffer allocation in tandem production lines. *Journal of Intelligent Manufacturing* 18 (3): 411–420.

Fu, M. C. 2002. Optimization for discrete-event simulation: Theory and practice. In *System Simulation and Scientific Computing*, ed. Z. J. Chen, 50–50. Beijing: Int Acad Publ Beijing World Publ Corp.

Gen, M., and R. Cheng. 1997. *Genetic algorithms & engineering design*. Engineering Design and Automation. New York, NY: John Wiley & Sons, Inc.

Gershwin, S. B., and J. E. Schor. 2000. Efficient algorithms for buffer space allocation. *Annals of Operations Research* 93 (1): 117–144.

Goldberg, D., and K. Deb. 1990. A comparative analysis of selection schemes used in genetic algorithms. In *Foundations of Genetic Algorithms*, ed. G. Rawlings, 69–93. San Mateo, CA: Morgan Kaufmann Publishers.

Harris, J. H., and S. G. Powell. 1999. An algorithm for optimal buffer placement in reliable serial lines. *IIE Transactions* 31:287–302.

Heavey, C., H. T. Papadopoulos, and J. Browne. 1993. The throughput rate of multistation unreliable production lines. *European Journal of Operational Research* 68 (1): 69–89.

Hillier, F. S., and K. C. So. 1995. On the optimal design of tandem queueing systems with finite buffers. *Queueing Systems* 21 (3): 245–266.

Hillier, F. S., K. C. So, and R. W. Boling. 1993. Toward characterizing the optimal allocation of storage space in production line systems with variable processing times. *Management Science* 39 (1): 126–133.

Hillier, M. S. 2000. Characterizing the optimal allocation of storage space in production line systems with variable processing times. *IIE Transactions* 32 (1): 1–8.

Huang, M.-G., P.-L. Chang, and Y.-C. Chou. 2002. Buffer allocation in flow-shop-type production systems with general arrival and service patterns. *Computers & Operations Research* 29 (2): 103–121.

Lacksonen, T. 2001. Empirical comparison of search algorithms for discrete event simulation. *Computers and Industrial Engineering* 40 (1-2): 133–148.

L'Ecuyer, P., and G. Yin. 1998. Budget-dependent convergence rate of stochastic approximation. *SIAM J. on Optimization* 8 (1): 217–247.

Lutz, C. M., K. R. Davis, and M. H. Sun. 1998. Determining buffer location and size in production lines using tabu search. *European Journal of Operational Research* 106 (2-3): 301–316.

Oduguwa, V., A. Tiwari, and R. Roy. 2005. Evolutionary computing in manufacturing industry: an overview of recent applications. *Applied Soft Computing* 5 (3): 281–299.

Papadopoulos, H. T., C. Heavey, and J. Browne. 1993. *Queuing theory*. Springer.

Papadopoulos, H. T., C. Heavey, and M. E. J. O'Kelly. 1989. Throughput rate of multistation reliable production lines with inter station buffers. (i) exponential case. *Comput. Ind.* 13 (3): 229–244.

Rudolph, G. 1996. Convergence of evolutionary algorithms in general search spaces. In *Proceedings of IEEE International Conference on Evolutionary Computation*, 50–54. Nagoya, Japan: IEEE.

Spinellis, D., C. Papadopoulos, and J. M. Smith. 2000. Large production line optimization using simulated annealing. *International Journal of Production Research* 38 (3): 509–541.

Srinivas, M., and L. M. Patnaik. 1994. Genetic algorithms: a survey. *Computer* 27 (6): 17–26.

Tekin, E., and I. Sabuncuoglu. 2004. Simulation optimization: A comprehensive review on theory and applications. *IIE Transactions* 36 (11): 1067–1081.

Turchyn, O. 2007. Comparative analysis of metaheuristics solving combinatorial optimization problems. In *CADSM '07. 9th International Conference - The Experience of Designing and Applications of CAD Systems in Microelectronics.*, 276–277. Polyana, UKRAINE.

Wolpert, D., and W. Macready. 1997. No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation* 1 (1): 67–82.

Yamashita, H., and T. Altiok. 1998. Buffer capacity allocation for a desired throughput in production lines. *IIE Transactions* 30:883–892.

AUTHOR BIOGRAPHIES

BIRKAN CAN is a full time research postgraduate in Enterprise Research Centre at the Department of Manufacturing & Operations Engineering at University of Limerick. He received a BSc. in Metallurgical and Material Engineering from Middle East Technical University (METU) in 2005. After a conditional year in Production Management in Industrial Engineering (METU), he has transferred to the University of Limerick for PhD. His current research interests involve simulation-based multi-objective optimisation and metamodelling, particularly of manufacturing and supply chain systems. His email address is <Birkan.Can@ul.ie>.

ANDREAS BEHAM received his MSc in computer science in 2007 from Johannes Kepler University (JKU) Linz, Austria. His research interests include heuristic optimization methods and simulation-based as well as combinatorial optimization. Currently he is a research associate at the Research Center Hagenberg of the Upper Austria University of Applied Sciences (Campus Hagenberg). His email address is <Andreas.Beham@heuristiclab.com>.

CATHAL HEAVEY is a Senior Lecturer of Operations Management in the Department of Manufacturing & Operations Engineering at the University of Limerick. He is an Industrial Engineering graduate of the National University of Ireland (University College Galway) and holds a M. Eng.Sc. and Ph.D. from the same University. He has published in the areas of queuing and simulation modelling. His research interests includes, simulation modelling of discrete-event systems; modelling and analysis of supply chains and manufacturing systems; process modelling; component-based simulation and decision support systems. His email address is <Cathal.Heavey@ul.ie>.