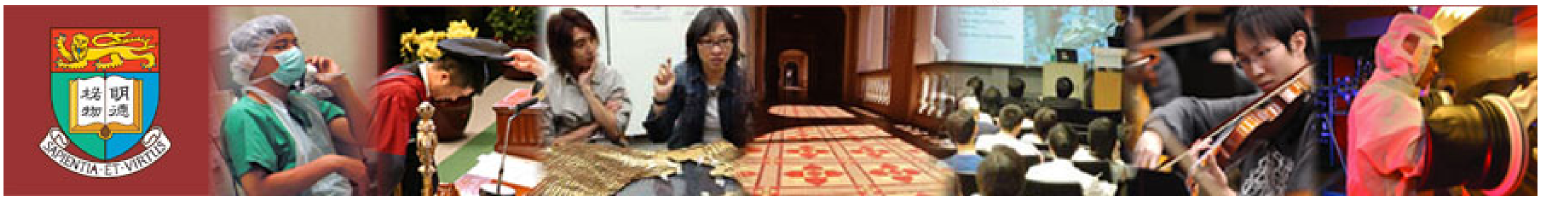

Preprints of the 18th IFAC World Congress
Milano (Italy) August 28 - September 2, 2011

An Optimization Framework for Modeling and Simulation of Dynamic Systems based on AIS

C.S.K. Leung* and H.Y.K. Lau**

** Department of Industrial & Manufacturing Systems Engineering, the University of Hong Kong, Hong Kong, China, (e-mail: chris06@hku.hk)*
*** Department of Industrial & Manufacturing Systems Engineering, the University of Hong Kong, Hong Kong, China, (e-mail: hyklau@hku.hk)*

Abstract: Modeling and simulation can be used in many contexts for gaining insights into the functioning, performance, and operation, of complex systems. However, this method alone often produces feasible solutions under certain operating conditions of a system in which such solutions may not be optimal. This is inevitably inadequate in circumstances where optimality is required. In this respect, an approach to effectively evaluate and optimize system performance is to couple the simulation model with operations research techniques. In this paper, an optimization framework consisting of a simulation model and an immunity-inspired algorithm is proposed for optimizing the key parameters in the domain of automatic material handling.

Keywords: simulation, optimization, artificial intelligence, artificial immune systems.

1. INTRODUCTION

In recent years, increasing competitive market factors, e.g. more rigid government regulations, increasing number of competitors, shorter product life cycle and more demanding customers, have created great pressure on all supply chain parties, especially manufacturers and distributors, to improve the efficiency and effectiveness of their production and distribution systems. For these reasons, analyzing and evaluating such systems, especially for complex automated material handling systems (AMHSs), are essential for improving and optimizing their operation to meet these challenges. In the past, these systems were investigated mainly by analytical methods such as linear programming. However, with the advancement of manufacturing technologies, these systems are increasingly more complex. These complex systems that are inherently stochastic in nature, with complex relationships between system components, existence of uncertainties and real world dynamics, make analytical methods hardly applicable. To meet the challenges, these systems can be studied more effectively and efficiently by computer-based modeling and simulation approaches. Unlike a mathematical model, simulation can handle uncertain structure and stochastic parameters of a system to reflect the dynamics and to allow the performance of comprehensive analyses. In addition, simulation is a cost-effective means for new system or process design as alternative solutions can be evaluated for correctness and feasibility before any actual implementation.

While it is well acknowledged that modeling and simulation techniques together with state-of-the-art simulation tools provide an effective means to analyze and visualize the performance of complex engineering systems, the decisions taken based on the results generated by simulation studies often depend on the quality of the simulation model and the experience of the analyst. This is inadequate from an optimization viewpoint. In order to improve the optimality of the process of simulation, a means to direct the undertaking of simulation study would be academically interesting and of great practical value. In this respect, this paper reports the development of an optimization framework for modeling and simulation of dynamic systems based on an emerging artificial intelligence method know as Artificial Immune Systems (AIS).

AIS is a comparatively new bio-inspired computation paradigm, which captures the ideas from biological immune system for modeling system behaviors and deriving solution methods to solve a wide array of problems. Such an engineering analogue of human immune system has drawn substantial attention recently due to its promising problem solving capability and its deep inspiration to the engineering sciences. AIS embodies a powerful and diverse set of features including autonomy, spatially distributed nature, dynamically changing coverage, specificity, diversity, immune learning, and memory, as well as the important immunological principles and theories, namely: negative selection principle, clonal selection principle, immune network theory, and danger theory. By making use of these immunological ideas, a number of algorithms have been developed to perform different tasks e.g. autonomous vehicles control (Lau et al., 2007), mobile robot navigation (Luh and Liu, 2008), distributed intrusion detection systems (Beltran, 2002), etc. There are many applications involving optimization e.g. job shop scheduling, travelling salesman problems, routing problems, etc. The AIS research community has considered optimization a promising application area for immunity-inspired algorithms, bringing about some novel algorithms e.g. CLONALG algorithm (de Castro and Von Zuben, 2000), the B-Cell algorithm (Timmis et al., 2004), and Opt-aiNet (de Castro and Timmis, 2002). Inspired by the appealing properties of AIS, an immunity-inspired optimization algorithm that captures the immunological ideas including clonal selection, affinity maturation, and immune suppression is developed. The algorithm works in cooperation with a simulation model to solve optimization problems, e.g. to optimize the overall processing time of machines in factories and other operations where AMHSs play a crucial role.

2. A FRAMEWORK FOR SIMULATION-BASED OPTIMIZATION

In this section, the framework (Fig. 1) for carrying out simulation optimization is described. The main concept behind this proposed approach is to use the evolutionary nature of AIS to direct the search for favorable solutions that are evaluated by the simulation model.

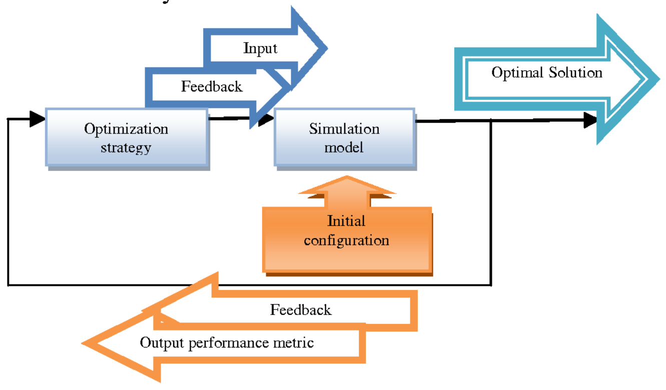

Fig. 1. A framework for simulation optimization

The framework consists of two critical components, namely 1) the simulation model and 2) the immunity-inspired optimization algorithm. The problem to be optimized is represented by a discrete-event simulation model, which is defined separately from the optimizer, taking inputs produced from the optimizer and then generates an output performance metric as feedback about the quality of solutions obtained from the optimizer. Based on the input values generated by the optimizer, the simulation is executed automatically with various parameter settings. It is worth emphasizing that the variability of the simulation outputs must be minimized by running multiple replications of the simulation model with the same input values but different random number streams. The optimization strategy is a search algorithm. The quality of the simulation results previously generated directs the algorithm to search for a new set of input values, hence producing feedback about the progress of the search for the optimal solution with respect to certain criteria, which in turn derives new inputs to the model. This iterative process repeats until a pre-specified termination criterion is met. In this optimization framework, the optimizer is implemented with VBA, whereas the simulation model is developed using the Flexsim simulation tool.

2.1 Simulation Model

2.1.1 Problem Formulation and Objective Setting

A single objective simulation-based optimization process is regarded as the search for an optimum of a simulation process, i.e., to find an optimal set of input decision variables that optimizes an output variable of the simulation model. To formulate a mathematical model for the simulation-based optimization problem, assume $X = \{x_j: j = 1, 2, \ldots, m\} \in \Omega$ being an m-dimensional vector of decision variables in real number representing the input variables for a simulation mode and $\Omega \in [l_j, u_j]$ is the feasible region of the decision variables x_j with the lower bound l_j and the upper bound u_j. The objective function $f(X)$ is an output random variable for the simulation model as it is assumed that $f(X)$ is not available directly but can be estimated by performing a sufficient number of simulation runs with input variables X. The optimization problem of interest is formulated as follows:

$$\min_{X \in \Omega} f(X) = E[L(X, \omega)] \quad (1)$$

Subject to

$$l_j \le x_j \le u_j \text{ (for } j = 1, 2, \ldots, m) \quad (2)$$

$$\sum_{j=1}^{m} (a_{jk}\, x_j) \le c_k \text{ (for } k = 1, 2, \ldots, n) \quad (3)$$

where the objective function $f(X)$ is the expected value of the random output variable $L(X, \omega)$ that is obtained from the simulation model, ω is a sample path, L is a sample performance measure, and Equations (2) and (3) are range constraints and linear constraints.

2.1.2 Simulation Strategy and Simulation Tool

After defining the objective, the next step is to consider the simulation strategy and simulation tool that fit the problem domain. In general, there are four types of discrete-event simulation strategies including process-interaction method, event-scheduling method, activity scanning method, and three-phrase method (Banks, 1998). With the problems concerned, the events to be simulated are not complex and the relationships between them are relatively straightforward, the event-scheduling method, thus, is used as the simulation strategy for this study. In this study, the discrete-event, object-oriented simulation tool - Flexsim is employed for implementing the simulation model. The motivation for adopting a commercial simulation tool for the study is as follows: First, Flexsim adopts the event-scheduling method as its simulation engine. Second, it can model systems on either discrete or continuous basis. Another benefit is that Flexsim is able to interface with common database and spreadsheet applications for manipulating data.

2.2 Optimization strategy and algorithm formulation

It is widely known that the essence of the clonal selection principle for an optimization problem is to generate a varied, enlarged population of antibodies around their parents based on their antigenic affinity (fitness). In this way, the search space is expanded from a low dimensional space to a high dimensional one. At the end of each generation, the population will return to the original size with elitist antibodies with better affinity. From generation to generation, the original population evolves in a way that high performers are likely to survive and reproduce new and hopefully better children, while low performers will be eliminated from the population. As a result, an optimal solution can be obtained after a sufficient number of iterations are performed. In this section, an innovative optimizer called suppression-controlled clonal selection algorithm (SCCSA) is developed. The mapping between the biological immune system and the proposed artificial one is given in Table 1.

Table 1. Mapping between the biological immune system and SCCSA

Biological Immune System	SCCSA
Antigen (Ag)	Objective function (simulation model)
Antibody (Ab)	Candidate solution (a set of decision variables)
Ag-Ab affinity	Fitness value of each candidate solution
Ab-Ab affinity	Distance to other candidate solutions
Immune (Ab-Ab) suppression	Mechanism to control the number of nearby candidate solutions with regards to Ab-Ab affinity

SCCSA has taken several key immunological features into consideration: clonal selection and expansion, affinity maturation, metadynamics, and immune suppression. In SCCSA, two types of entities are considered – an antigen (Ag) and an antibody (Ab). While the antigen represents a problem to be solved or optimized, each antibody in a population corresponds to a candidate solution to the problem. Therefore, the evaluation of the simulation model for a given candidate solution represents the antigenic Ag-Ab affinity. In addition, Ab-Ab affinity, which is a measure for maintaining the population diversity, mimicking the immune suppression undergone by the interacting B-cells based on immune network theory, is also considered. The proposed SCCSA is, in principle, a search algorithm in the solution space initiated by the antigenic affinity, which comprises four main immune operators: cloning operator, hypermutation operator, suppression operator, and selection operator. The detail of the proposed algorithm is discussed below and the block diagram showing the computational steps for the proposed SCCSA is presented in Fig. 2.

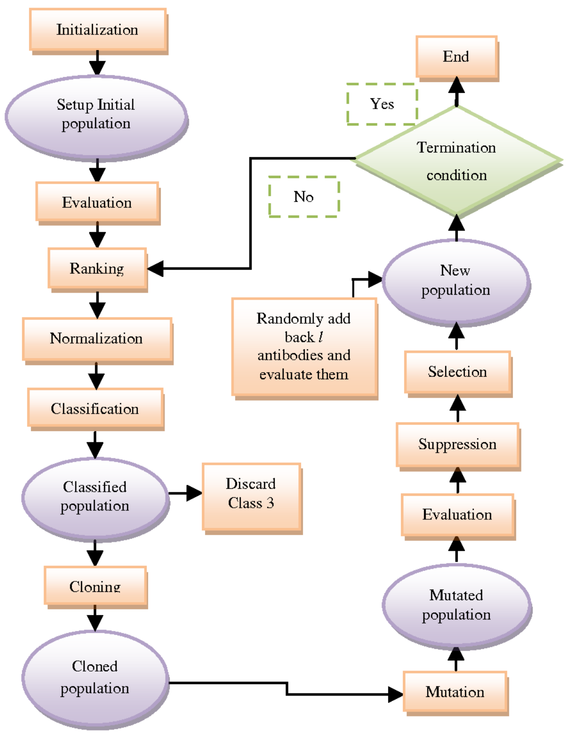

Fig. 2. Computation model of SCCSA

Step 1: Initialization

Assume t is the iteration counter of the algorithm so that $\mathbf{Ab}(t) = \{ab_i: i = 1, 2, \ldots, N\}$ is a set of candidate solutions of t^{th} iteration in the current fitness landscape, where N is the population size, and $ab_i = \{x_j: j = 1, 2, \ldots, m\}$ is a candidate solution to the fitness function. Each candidate solution x_j is a string of length m with the lower bound l_j and the upper bound u_j. At $t = 0$, the initial population of candidate solutions is randomly generated using the formula:

$$x_j = l_j + \beta \times (u_j - l_j) \tag{4}$$

where β is a random number uniformly distributed in the range [0, 1].

Step 2: Simulation Evaluation of Initial Population

The fitness of each antibody $ab_i \in \mathbf{Ab}(t)$ is evaluated using simulation. In this way, a fitness vector $\vec{f}$ storing all the parent antibodies' fitness f_i ($i = 1, 2, \ldots, N$) is determined.

Step 3: Ranking

All antibodies in the current population $\mathbf{Ab}(t)$ is first ranked in ascending order in accordance with their fitness value.

Step 4: Fitness Normalization

The elements in the fitness vector $\vec{f}$ at this step are normalized within the interval [0, 1] as follows:

$$F_i = \frac{f_H - f_i}{f_H - f_L} \tag{5}$$

where F_i is the i^{th} antibody's normalized fitness, f_i is the i^{th} antibody's fitness, f_H is the highest fitness value among the current antibody population, and f_L is the lowest one.

Step 5: Classification

To better increase the efficiency, the population is divided into three classes - $\mathbf{Ab}^{(c1)}(t)$, $\mathbf{Ab}^{(c2)}(t)$, and $\mathbf{Ab}^{(c3)}(t)$ with different sizes for handling different tasks as defined below:

Class 1 - $\mathbf{Ab}^{(c1)}(t)$: $F_i \geq \bar{F} + 0.5\sigma$

Class 2 - $\mathbf{Ab}^{(c2)}(t)$: $\bar{F} - 0.5\sigma < F_i < \bar{F} + 0.5\sigma$

Class 3 - $\mathbf{Ab}^{(c3)}(t)$: $F_i \leq \bar{F} - 0.5\sigma$

where $\bar{F}$ is the mean of all antibodies' fitness and σ is the standard deviation. After the classification is done, different jobs will be undertaken by them. Class 1 containing high performers will advance towards the local optimal solutions with a lower mutation rate. Class 2 comprising medium performers will concentrate on exploring more promising areas for the global optimum by performing a more dramatic mutation. Class 3 antibodies possessing the lowest fitness will do nothing and will be substituted by randomly generated antibodies for maintaining the population diversity.

Step 6: Cloning

The cloning operator helps enlarge the population by generating a number of copies of Class 1 and Class 2 subpopulations, which are denoted as $\mathbf{C}^{(c1)}(t)$ and $\mathbf{C}^{(c2)}(t)$ respectively. Hence the size of the population now is $N + N_c$ and N_c is obtained by:

$$\begin{cases} N_c = \sum c_i \\ c_i = \text{round}(max\_clone \times F_i) \end{cases} \tag{6}$$

where N_c is the total number of copies produced for the two classes of antibodies, c_i is the clone size for $ab_i \in \mathbf{Ab}^{(c1)}(t) \cup \mathbf{Ab}^{(c2)}(t)$, max_clone is the maximum clone size of each antibody, and round() is an operator for rounding its argument to the closest integer.

Step 7: Mutation

The hypermutation operator induces multi-point mutations to the pool of clones. The clones are mutated as follows:

$$\text{Class 1} \quad \begin{cases} \alpha = e^{-\rho \times F} \\ \mathbf{C}'^{(c1)}(t) = \mathbf{C}^{(c1)}(t) + \alpha \times R \times \omega_1 \end{cases} \tag{7}$$

$$\text{Class 2} \quad \begin{cases} \alpha = e^{-\rho \times F} \\ \mathbf{C}'^{(c2)}(t) = \mathbf{C}^{(c2)}(t) + \alpha \times R \times \omega_2 \end{cases} \tag{8}$$

where α represents the mutation rate that is inversely proportional to the normalized fitness F_i, ρ is an exponential coefficient controlling the decay of α, $R \in [-1, 1]$ is a m-dimensional random vector obtained with uniform distribution, and ω_1 and ω_2 are the mutation step factors for the Class 1 antibodies and the Class 2 antibodies respectively. In order to allow the better performers in Class 1 to take a smaller mutation step to locate local optima while diverting the direction for poorer performers in Class 2 by taking a larger mutation step in search for global optimum in a bigger search space, ω_2 is always set to be larger than ω_1 ($\omega_1 < \omega_2$).

Step 8: Simulation Evaluation of Mature Clones

Class 1 and Class 2 subpopulations are combined to form a total clone population $\mathbf{C}'(t)$ and then the fitness of each mature clone $C_i' \in \mathbf{C}'(t)$ ($i = 1, 2, \ldots, N_c$) will be evaluated using simulation. In this way, a fitness vector $\vec{f}'$ storing all the child's fitness f_i' ($i = 1, 2, \ldots, N_c$) is determined.

Step 9: Suppression

A suppression operator is introduced and works on each clone to avoid antibody redundancy and maintain the population diversity based on the idea of immune network theory such that B-cells are stimulated and suppressed by not only non-self antigens but the interacting B-cells. To achieve this, the affinity (similarity) among the newly generated antibodies is determined. The affinity between two antibodies is defined as the Euclidean distance between them:

$$\mathrm{d}(ab_i, ab_k) = \sqrt{\sum_{j=1}^{m}[ab_i(x_j) - ab_k(x_j)]^2} \le \varepsilon \tag{9}$$

where $\mathrm{d}(ab_i, ab_k)$ is the Euclidean distance between the two antibodies, m is the length of each antibody, and ε is a positive threshold value. In this step, if the distance between two clones is smaller than the threshold, then the clone with lower fitness is suppressed and eliminated from the population. This procedure is repeated until all clones are compared in terms of both affinity and fitness. Eventually, a surviving clone population $\mathbf{C}''(t)$ is formed and then enters into the selection process.

Step 10: Selection

An evolutionary selection operator is used to select only the improved children in the surviving clone population $\mathbf{C}''(t)$ with better fitness to replace some of the less fit parents. The low-fitness children in Class 3 are replaced by l randomly generated antibodies to enhance the population diversity. Finally by combining the updated Class 1 and Class 2 subpopulations and the replaced Class 3 subpopulation, a new population $\mathbf{Ab}(t+1)$ containing N high performers based on the simulation results (antigenic affinities) for the next generation t+1 is formed.

Step 11: Termination

To control the termination of the optimization process, the function Termination_Condition() is introduced. It returns True if no significant changes (change within an acceptable range, η) on the average fitness of both Class 1 and Class 2 subpopulations over successive iterations, *term_max*. The optimization process will also terminate if the maximum number of iterations *Tmax* is performed. If these conditions are not satisfied Steps 3 to 10 are repeated until one of the predetermined termination conditions is met. The pseudo-code of SCCSA is given in Table 2.

Table 2. Pseudo-code of SCCSA

```
1.   procedure SCCSA ($N$, $T_{max}$, $c$, $\alpha$, $\omega$, $\varepsilon$)
2.   $t \leftarrow 0$;
3.   Ab($t$) ← Generate_Initial_Population ($N$);
4.   Simulation_Evaluation (Ab($t$));
5.       while ( not Termination_Condition () ) do
6.           ($\mathbf{Ab}^{(c1)}(t)$, $\mathbf{Ab}^{(c2)}(t)$, $\mathbf{Ab}^{(c3)}(t)$) ← Classification (Ab($t$));
7.           ($\mathbf{C}^{(c1)}(t)$, $\mathbf{C}^{(c2)}(t)$) ← Cloning ($\mathbf{Ab}^{(c1)}(t)$, $\mathbf{Ab}^{(c2)}(t)$, $c$);
8.           ($\mathbf{C}'^{(c1)}(t)$, $\mathbf{C}'^{(c2)}(t)$) ← Hypermutation ($\mathbf{C}^{(c1)}(t)$, $\mathbf{C}^{(c2)}(t)$, $\alpha$, $\omega$);
9.           $\mathbf{C}'(t)$ ← Combination ($\mathbf{C}'^{(c1)}(t)$, $\mathbf{C}'^{(c2)}(t)$);
10.          Simulation_Evaluation ($\mathbf{C}'(t)$);
11.          $\mathbf{C}''(t)$ ← Supprssion ($\mathbf{C}'(t)$, $\varepsilon$);
12.          Ab($t$+1) ← Selection (Ab($t$), $\mathbf{C}''(t)$);
13.          $t \leftarrow t + 1$;
14.      end while
15.  end procedure
```

3. CASE STUDY

3.1 Scenario Description

In this section, a case study is performed based on the operation of an integrated automated material handling systems (AMHS) installed in Flexible Automation Lab at the University of Hong Kong. The system consists of a flexible conveyor system (FCS) and an automated storage and retrieval system (AS/RS) working collaboratively. The objective of the study is to minimize the system cycle time, i.e., the time between taking out pallets from AS/RS compartments and placing them back to the compartments after all the manufacturing processes are completed.

3.1.1 The Basics of the AMHS

The FCS composes of a number of interconnected 2-meter long modular chain conveyor units that can be flexibly reconfigured as depicted in Fig. 3.

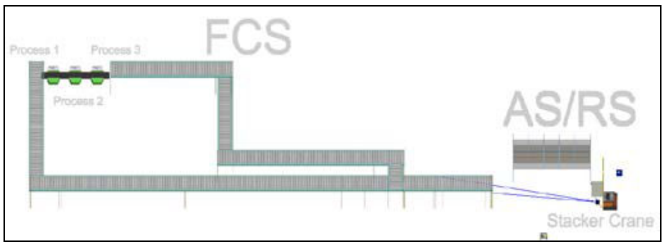

Fig. 3. The layout of the AMHS

Each conveyor module has a programmable logic controller to control the movement of the items and to communicate with the central computer. For the AS/RS, it is connected to the conveyor by the stacker crane, consisting of the single-deep 5-cloumn rack with 20 compartments for the storage of intelligent pallets and items.

The logic of the loading and unloading of the crane is dependent on the type of pick-up order received. There are 2 types of pick-up orders: one is initiated by the AS/RS and another one is initiated by the FCS. When the crane receives a pick-up order initiated by the AS/RS, it transports the pallet containing a piece of raw material from the corresponding compartment to the conveyor. For the pick-up orders initiated by the conveyor, the crane moves the pallet with a processed item from the conveyor back to its original compartment in the AS/RS. If two different types of orders are received at the same time, the working sequence is based on the current position of the crane. That is to say, if the crane parks in front of the conveyor, the pick-up task initiated from the conveyor will be handled first; if the crane parks at one of the column positions of the rack, the pick-up order initiated by the AS/RS will be performed first.

3.1.2 The Operation of the System

The operation of the system is implemented with the Flexsim simulation tool. These operation steps are performed sequentially in the simulation process. The operation is as follows:

1. Initially, all the pallets are stored in the compartments of the AS/RS and the stacker crane waits at the starting position.
2. After the system is turned on, each pallet is moved out from the compartment and, in turn, placed on the conveyor by the stacker crane.
3. After the pallet arrives at the conveyor, it is transported via different sections of the conveyor system where it undergoes different manufacturing processing activities. These manufacturing processes are modeled as stochastic processes where indeterminism exists.
4. When all the processes are completed, the pallet is sent back to its original compartment in the AS/RS and the cycle time of the whole process is measured.

3.1.3 Assumptions

Since the system being studied is a laboratory setup for experimental purposes, a number of real-world factors are ignored. Thus, the following assumptions are made:

1. The total number of products is 6.
2. The system only processes one type of product.
3. The demand created from succeeding processes or end customers is not considered.
4. The arrival rate of products generated from preceding processes or suppliers is not considered.
5. The processing time of each processing activity is a random variable that follows a normal distribution.
6. No machinery maintenance and mechanical breakdown are considered so that rework and yield are not considered.
7. All products are processed in the same sequence.

3.1.4 Initial Model Settings

The configuration and system parameters of the actual system were implemented in Flexsim and the initial model settings (Table 3) are as follows:

Table 3. Initial model settings

Item	**Value**
Conveyor speed	14.9 cm/sec
Crane speed	5.5 cm/sec
Forks speed	4.7 cm/sec
Crane acceleration	3 cm/sec ^{2}
Crane deceleration	3 cm/sec ^{2}
Capacity of crane	1 pallet
Spacing of conveyor	1 pallet
Processing time of Process 1	Normal distribution with a mean of 6 sec and a standard deviation of 4 sec
Processing time of Process 2	Normal distribution with a mean of 7 sec and a standard deviation of 3 sec
Processing time of Process 3	Normal distribution with a mean of 8 sec and a standard deviation of 4 sec

3.2 Performance Evaluation

3.2.1 Sensitivity Analysis of Key Parameters and Optimization Problem Formulation

As the results of SCCSA may be sensitive to certain initial parameters including the number of replications for each simulation run, initial population size, maximum number of clones to be produced by the parents, mutation factors, suppression threshold, and termination factors, they are tested through sensitivity analysis to observe the impact of individual parameters on the performance.

From the results of the sensitivity analysis, we can see that the maximum speed and forks speed (forks speed of moving up and down) of the stacker crane are the most critical factors affecting the system's performance in terms of cycle time. Based on these results, we can conclude that the crane's speed is a determining factor of the whole system and optimization of these two parameters can improve the overall system performance. Therefore, a set of decision variables or an antibody *ab* is defined as follows: x_1 is taken to be the maximum speed and x_2 being the forks speed, and the optimization problem is represented by:

$$\min_{ab\in\Omega} f(ab) = E[cycle\ time] \tag{10}$$

Subject to

$$1 \le x_j \le 50 \text{ (for } j = 1, 2) \tag{11}$$

where the objective function $f(ab)$ is the expected value of the random output variable *cycle time* that is obtained from the simulation model, and Eq. (11) represents the physical constraints.

3.2.2 Experimental Results and Analysis

To evaluate the performance of SCCSA, two experiments were performed. The first one is to make a comparison between the results of the simulation model without the use of an optimizer and the results of coupling simulation and optimization in order to investigate the optimization algorithm's effectiveness. The second experiment was conducted to benchmark SCCSA against CLONALG and Opt-aiNet with respect to its convergence. Each algorithm was run for 10 times to obtain the average performance of each algorithm on the problem.

Based on the results from the sensitivity analysis, the parameters of SCCSA were set as: Initial population size, $N = 5$; Maximum number of clone, *max_clone* = 3; Maximum iteration, *Tmax* = 100; Number of replication per each fitness evolution, *replication* = 10; Exponential distribution coefficient, $p = 0.05$; Mutation step factors, $\omega_1 = 0.5$ and $\omega_2 = 3.5$; Suppression threshold, $\varepsilon = 2$; Acceptable Range, $\eta = 0.1$; Number of successive iterations that no significant change is found, *term_max* = 5, and the settings of CLONALG and Opt-aiNet were set with similar values.

3.2.2.1 Simulation without Optimization vs. Simulation with Optimization

The results obtained from the first experiment are shown in Table 4 in which simulation results were obtained by performing the algorithms for 100 generations.

Table 4. Comparison between the results of simulation alone and those of combining optimization and simulation

	Cycle Time
Simulation without optimization	914.46 sec
Optimization via SCCSA	399.46 sec
Difference (%)	515.00 (56.32%)
Optimization via CLONALG	399.49 sec
Difference (%)	514.97 (56.31%)
Optimization via Opt-aiNet	399.47 sec
Difference (%)	514.99 (56.32%)

The table reveals that the cycle time of the whole process is reduced by more than 56% by employing optimization algorithms (SCCSA, CLONALG or Opt-aiNet). This indicates that the use of an optimizer can result in significant improvement. The results indicated that the speed of the crane should be increased for better performance.

3.2.2.2 Performance Comparison among SCCSA, CLONALG and Opt-aiNet

In order to assess the convergence characteristics of SCCSA, the rate of arriving at optimal solutions by SCCSA is compared with that of CLONALG and Opt-aiNet. All the algorithms are applied to solve the above-mentioned problem. The results are shown in Fig. 4.

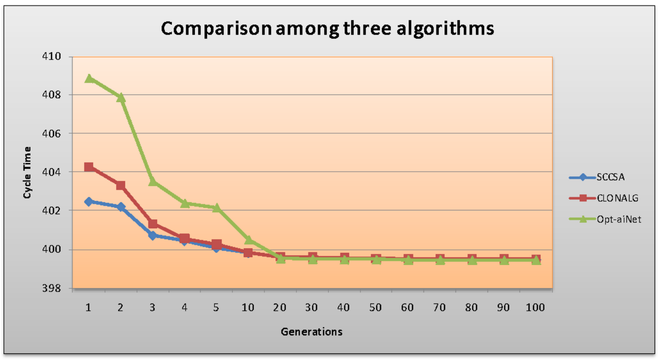

Fig. 4. Comparison of the rate of convergence of SCCSA, CLONALG and Opt-aiNet

It can be seen that SCCSA converges faster than CLONALG and Opt-aiNet as near optimal solution can be found with a much smaller number of generations taken by SCCSA. Therefore, SCCSA is the best performer. The performance of SCCSA is largely attributed to the suppression operator and classification operator. The suppression operator helps to reduce antibody redundancy, and hence significantly minimizes the number of unnecessary simulation evaluations. The classification operator works closely with the multi-point mutation operator to improve the convergence and enhance the search performance by facilitating local and global search simultaneously. As a result, SCCSA takes much fewer numbers of generations to reach the optimum. This implies less computational resource and cost are needed, thus indicating that SCCSA outperforms the other two algorithms.

4. CONCLUSIONS

The results of the case study show that the proposed immune-inspired SCCSA framework can produce a set of optimal system parameters for minimizing the cycle time of the system in an efficient way. When SCCSA is compared with other two AIS-based optimization algorithms, namely, CLONALG and Opt-aiNet, SCCSA shows the best performance on the AMHS problem.

Moreover, the framework also serves as a decision support tool for AMHS operation for helping to find optimal system settings e.g. speed of machines, number of operators or machines, and any other decision variables of interest. Although the system under study is a laboratory scale system, the framework is scalable to tackle other real world problems of larger complexities. In the future, the framework will be extended to tackle nonlinear and multiobjective problems.

REFERENCES

Banks, J. (1998). *Handbook of Simulation - Principles, Methodology, Advances, Applications, and Practice*, John Wiley & Sons.

Beltran, F. N. (2002). A change detection software agent based on immune mixed selection. *Evolutionary Computation*, 1, 693–698.

de Castro, L. N. and Timmis, J. (2002). An artificial immune network for multimodal function optimization. *Proceedings of the 2002 Congress on Evolutionary Computation*, 1, 699-704.

de Castro, L. N. and Von Zuben, F. J. (2000). The Clonal Selection Algorithm with Engineering Applications. *Workshop Proceedings of Genetic and Evolutionary Computation Conference*, Las Vegas, 36-37.

Lau, H. Y. K., Wong, V. W. K. and Lee, I. S. K. (2007). Immunity-based autonomous guided vehicles control. *Appl. Soft Comput.*, 7, 41-57.

Luh, G.-C. and Liu, W.-W. (2008). An immunological approach to mobile robot reactive navigation. *Appl. Soft Comput.*, 8, 30-45.

Timmis, J., Edmonds, C. and Kelsey, J. (2004). Assessing the performance of two immune inspired algorithms and a hybrid genetic algorithm for optmisation. *Genetic and Evolutionary Computation Conference*, 308–317. Springer.