Order Pickings in an AS/RS with Multiple I/O Stations using an Artificial Immune System with Aging Antibodies

K.L. Mak and Peggy S.K. Lau

***Abstract*— This paper proposes an age artificial immune system (AAIS), for optimal order pickings in an Automated Storage and Retrieval System (AS/RS) with multiple input/ output stations. A mathematical model is presented to describe the characteristics of the AS/RS. It is optimized with the proposed algorithm, which is based on the clonal selection principle and the aging concept. Unlike conventional algorithms for artificial immune systems, the proposed algorithm consists of antibodies whose abilities to be cloned and to survive depend on their ages, and adopts a mutation scheme based on randomized rankings. To further improve the performance of AAIS, a crossover operator is also included in the algorithm to form the AAIS_CX algorithm. The performance of both algorithms is tested with the problems of optimal order pickings in an AS/RS with multiple input/output stations. Comparison of the results obtained by using AAIS_CX, AAIS, the techniques of nearest neighbor heuristics, genetic algorithms and ant colony systems clearly shows that AAIS_CX is superior to the other algorithms. Suggestions for future work are also included.**

***Index Terms*— Artificial immune system, AS/RS, clonal selection, order picking.**

I. Introduction

Many computational intelligence methodologies are inspired from natural phenomena, such as evolution and biological processes in human bodies. Artificial immune system, which has received much attention in recent years, is inspired from the immune system in human bodies. The technique is capable of learning and using memory to enhance the utilization of the available information, and is also well known for its easy adaptation to a changing environment. It has been applied to solve problems in a wide range of areas, such as pattern recognition, vehicle routing, and job shop scheduling.

Mak, Lau and Wang [1] have introduced the concept of aging in genetic algorithms for the design of virtual cellular manufacturing systems. They have assumed that the survival and birth rates of the chromosomes are age dependent, and that the chromosomes are discarded when their ages have exceeded a certain value. It has been shown that the aging concept is effective in preventing the search process from premature convergence. In this paper, an age artificial immune system (AAIS) based on the clonal selection principle and the aging concept is proposed. Unlike conventional algorithms for artificial immune systems, the proposed algorithm consists of antibodies whose abilities to be cloned and to survive depend on their ages, and adopts a mutation scheme based on randomized rankings. To further improve the proposed algorithm, a crossover operator is also included to form the AAIS_CX algorithm. Both algorithms are used to solve an order picking problem for an AS/RS with multiple input/output (I/O) stations. The results are compared with those of other metaheuristics like genetic algorithm (GA) and ant colony system (ACS).

This paper is organized as follows. Section 2 gives the literature review on the related topics. Section 3 describes the proposed algorithms and explains how they differ from conventional ones. Numerical experiments of optimizing the ones counting problem and the order picking sequence for an AS/RS with multiple I/O stations are presented in sections 4 and 5. The conclusion and suggestions for further research are given in Section 6.

II. Literature review

Artificial immune system (AIS) is inspired from the human immune system. It forms an identification mechanism which is capable of identifying and combating dysfunction antigens from one’s cells and infectious microorganisms. Antigen presenting cells (APC) are present in the immune system to ingest and digest harmful antigens found. They fragment the antigens into antigenic peptides which form major histocompatibility complex (MHC) molecules. The molecules are then recognized by T cells. T cells are activated to divide and secrete chemical signals to mobilize other components of the immune system, e.g. B-cells, to combat the antigens. B-cells have receptor molecules of a single specificity on the surface. They then divide and differentiate into plasma cells which secrete antibodies, and the antibodies can neutralize antigens. In order to response to different antigens, a wide diversity of B-cells is needed and achieved by frequent mutation and editing of genes [2]. Besides, there are different receptors on the surface of an antibody. They are responsible for binding and destroying antigens. Each antibody can only carry one kind

Manuscript received February 28, 2007.

K. L. Mak is Professor at the Dept. of Industrial and Manufacturing Systems Engineering (IMSE), the University of Hong Kong. (phone: 852-28592586; e-mail: makkl@hkucc.hku.hk).

Peggy S. K. Lau is a PhD student at the Dept. of IMSE, HKU (e-mail: peggylau@graduate.hku.hk).

of receptors. Receptors are hypermutated when B cells secret antibodies, resulting in different kinds of antibodies. It is suggested in [3] that occasionally B lymphocytes are found that have undergone receptor editing. The immune system practices molecular selection of receptors in addition to clonal deletion to avoid self-reactive cells.

By mimicking human's immune system, AIS can be applied to cases with no prior knowledge [3]. According to [3], there are 3 main categories of AIS. They are clonal selection principle based AIS, GA-aided AIS and immune networks. Clonal selection principle describes how the antibodies with higher affinity are selected, cloned and mutated, such that the population of antibodies can recognize and combat the present antigens better. CLONAGE [4] is one of the most popular algorithms based on this principle, which does not include any crossover operation among antibodies. GA-aided AIS makes use of the crossover operator in GA to form new antibodies on top of the clonal selection principle. Algorithms based on immune networks are inspired by the network principle among cells. It is suggested that B-cells are stimulated by antigens, and suppressed by other similar B-cells at the same time. This improves the diversity of cells and makes the algorithms more adaptable to a changing environment.

The proposed AAIS is based on the clonal selection principle. In AAIS, mutation is the only process which changes the antibodies, thus determining the efficiency of the algorithm. However, too much mutation may lead to loss of good antibodies. The difficulty can be overcome in some cases by including tailored made mutation schemes in the algorithms. For example, two tailor-made mutation processes, division processing with Simulated Annealing and escape processing were introduced in [5] for creating and integrating sub-tours among solutions to solve n-TSP problems. It was shown that the resulting algorithm is better than GA in terms of solution quality and computational time. Another clonal selection principle based algorithm called opt-IA was proposed in [6], which is a modification of CLONAGE [4]. The pseudo code of opt-IA is detailed in figure 2. In such an algorithm, B-cells are selected without duplication from the cloned population to form B-cells of the next generation. The remaining slots in the new population are filled up by randomly generated new B-cells. Unlike AAIS, the aging concept is not applied to the cloning and survival of antibodies in opt-IA. A B-cell is simply erased from the population at a particular age under the static strategy, or is erased with a probability governed by the equation, $P_{die}(\tau_B)=\left(1-e^{-\ln 2/\tau_B}\right)$ where τ_B is the age of an antibody. To test the performance of opt-IA, the algorithm is applied with different hypermutation operators to solve trap functions and a protein structure prediction problem [6]. It is shown that opt-IA performs better than CLONAGE. Although opt-IA is parameter sensitive, the performance of the algorithm can be improved by simultaneously using different mutation schemes. In [7], the performances of CLONAGE and opt-IA are tested and compared by solving the ones counting problem, trap functions, numerical functions, and the protein structure prediction problem. However, no test is conducted for common NP hard problems, like traveling salesman problems and vehicle routing problems.

In AAIS, the age concept is introduced to AIS. It was firstly applied in Age Genetic Algorithm (AGA) in Mak, Lau and Wang [1]. The authors have introduced a more comprehensive age concept to enhance diversity in the population. The survival and birth rates of individuals in a population depend on their ages. Age-group *l+1* of the new population is generated from age-group *l* according to the survival rate of the individuals. Individuals are selected as parents from different age groups according to their birth rates to give birth to new individuals. From the results reported in [1], AGA performs better than conventional GA. Compared with opt-IA, the age concept introduced in [1] is more comprehensive and brings larger effect on the search process.

Based on the clonal selection principal and the aging concept described above, the proposed AAIS is used to solve an order picking problem for an automated storage/ retrieval system (AS/RS) with multiple input/ output (I/O) stations. Since the introduction of AS/RS 50 years ago, different models of the system have been widely used in different industries. AS/RS do not only minimize human efforts in handling materials, but also increase~~s~~ the capability of handling heavy cargoes and allows computerized control to achieve optimal efficiency. Its advantages have been reported in many studies [8]. The order picking problem of an AS/RS has also been widely studied. Han [9] has shown that using dual command cycle in order pickings, the throughput of an AS/RS can be increased by 10-15%. Kanet [10] has detailed the cost related to the operations of an AS/RS and uses integer programming to determine the optimal operation sequence for retrieval of orders. Chetty and Reddy [11] have proposed a GA to solve the retrieval order sequencing problem for an AS/RS and have compared the algorithm with heuristics rules such as FCFS and NNB. The same problem has also been studied by Yin and Ran [12] using multiple pass GA. Lee and Schaefer [13] have presented both static and dynamic approaches to solve the order picking problem for an AS/RS with single I/O station. In the static approach, the optimal retrieval order sequence for a block of orders is determined. Once the orders have been completely processed, another block of orders is selected and its optimal retrieval order sequence determined. Berg and Gademann [14] have also applied a static block sequence approach to solve the order picking problem for an AS/RS with the I/O station located at an arbitrary position. They have modeled the problem as a transportation problem. Ghamai and Wang [8] have proposed a genetic algorithm to sequence retrieval orders for an AS/RS with multiple stock locations and shown that their algorithm performs much better than enumeration in terms of computational speed. However, the problem of optimizing both storage orders and retrieval orders simultaneously for an AS/RS with multiple I/O stations has received very little attention, although its solution has a profound effect on the operation of the system. On the storage racks in an AS/RS, some of the racks are connected with conveyors or other transport systems to transport the cargoes into or out of the racks of AS/RS, and these racks become input/output stations to the system. It is common that the cargoes which go into the

AR/RS line up on the conveyor at the input stations at a first-come-first-serve basis, and the cargoes which go out of the system are moved away from the system with the conveyors once they are out of the AS/RS. When there is more than one input/output station, the order picking sequence is constrained by the position of cargoes at particular station. A graphic representation of an AS/RS with one input and one output stations are shown in figure 1.

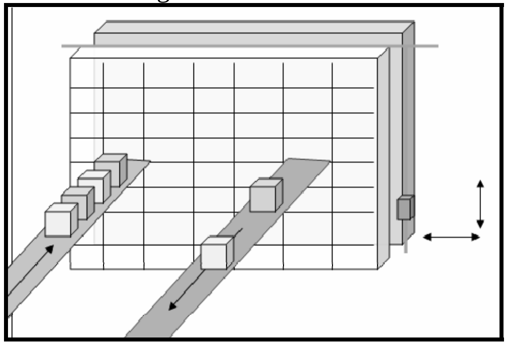


Fig. 1 Graphic representation of AS/RS with one input and one output stations.

```
opt-IA (l, d, dup, $\tau_B$, c, h, hm)

t:=0
$P^{(t)}$ := Initial_Pop()
Evaluate ($P^{(0)}$)
While ( not Termination_Condition())do
    $P^{(clo)}$ := Cloning ($P^{(t)}$, dup)
    If (H is TRUE) then
        $P^{(hyp)}$ := Hypermutation ($P^{(clo)}$, c,l)
        Evaluate($P^{(hyp)}$)
    If (M is TRUE) then
        $P^{(marco)}$:= Hypermacro($P^{(clo)}$)
        Evaluate($P^{(marco)}$)
    Aging ($P^{(t)}$, $P^{(hyp)}$, $P^{(marco)}$, $\tau_B$ )
    $P^{(t+1)}$:= ( $\mu+\lambda$ )-Selection($P^{(t)}$, $P^{(hyp)}$, $P^{(marco)}$)
    t:=t+1
end_while
```

Fig. 2 Pseudo-code of opt-IA [6]

III. Algorithms

A. AAIS

In the proposed AAIS, antibodies are assigned with an attribute called age. The algorithm differs from opt-IA in that the number of antibodies selected to enter the next generation and the number of clones produced from each selected antibody depend on the ages of antibodies. Both the clonal rate and the survival rate of an antibody increase initially and decline gradually as its age. Table 1 shows an example which indicates that the antibody reaches its golden age at age =2, and is eliminated from the body at age = 4.

Table 1 Clonal and survival rates

Age	0	1	2	3	4
Clonal Rate	0.5	0.8	0.9	0.6	0.3
Survival Rate	0.4	0.6	0.6	0.3	0

The following notations are used to facilitate the presentation:

t	iteration index (t= 0,1,2…)
n	population size
N	number of orders to be sequenced
n_f	number of antibodies survived to iteration $t+1$
n_r	number of new antibodies randomly generated for iteration $t+1$
cr_i	clonal rate at age i
sr_i	survival rate at age i
m	parameter in determining $MN(t, a)$
$Ab(t)$	the set of antibodies at iteration t
$Ab(t,a)$	antibody a at iteration t
$CAb(t)$	the set of cloned antibodies at iteration t
$CAb(t,a)$	cloned antibody a at iteration t
$CAb(t,best)$	the best cloned antibody at iteration t
$age(Ab(t, a))$	age of $Ab(t, a)$ at iteration t
$age(CAb(t,a))$	age of $CAb(t, a)$ at iteration t
$RAb(t,a)$	rank of the source of clones, $Ab(t, a)$ among $Ab(t)$
$Aff(Ab(t,a))$	affinity of $Ab(t, a)$ at iteration t
$Aff(CAb(t,a))$	affinity of $CAb(t, a)$ at iteration t
$CN(t,a)$	number of clones created from selected $Ab(t, a)$at iteration t
TCN	total number of clones created in each iteration
$MN(t,a)$	number of mutation carried out for $CAb(t, a)$ at iteration t
$Pb(CAb(t,a))$	probability of $CAb(t, a)$ to be survived at iteration t

The basic procedures of the proposed age artificial immune system are outlined below:

Step 1: Set t =0, generate n antibodies randomly, and assign $age(Ab(t,a))$ = 0 for $a = 1,2,3\ldots n$

Step 2:
a) Clone all the antibodies in $Ab(t)$ to $CAb(t,a)$. The number of clones created from $Ab(t,a)$ is determined by:

$$CN(t,a) = round\left(\frac{cr_{age(CAb(t,a))} \times Aff_{CAb(t,a)}}{\sum_{i=1}^{n_c} cr_{age(CAb(t,i))} \times Aff_{CAb(t,i)}} \times (TCN - n_c) \right)$$

b) Assign: $age(CAb(t,a))$ = 0 for $a = 1, 2, 3..\ (TCN-n_c)$
c) For $x = 1, 2,\ldots n_c$; $a= (TCN-n_c+1)\ldots .(TCN-n_c)$, copy $Ab(t,x)$ to the $CAb(t,a)$, set $age(CAb(t,a)) = age(Ab(t,x))$.

Step 3: Mutate the cloned antibodies by interchanging sections of the antibodies. The mutation scheme of interchanging two orders in the solution is applied here. The number of times of mutation performed is determined by:
$MN(t,a) = round(RAb(t,a) \times \text{rand}(0,1) \times m)$

and no mutation is performed on $CAb(t,a)$, for $a = (TCN-n_c+1)\ldots(TCN-n_c)$

Step 4: Set $t = t+1$; generate the population by:

a) Copy n_r randomly created antibodies to $Ab(t,x)$
b) Assign $age(Ab(t,x)) = 0$ for $x = 1,2,\ldots n_r$.
c) Select n_f antibodies from $CAb(t-1)$ to $Ab(t)$ and assign them as $Ab(t,x)$ for $x = n_r+1, \ldots, n_r+n_f$ with the following probability

$$Pb(CAb(t-1,a)) = \frac{sr_{age(CAb(t-1,a))}}{\sum_{i=1}^{TCN} sr_{age(CAb(t-1,i))}} \quad for\ a = 1,2,\ldots,TCN$$

d) Assign $age(Ab(t,a)) = age(Ab(t-1,a))+1$ for $a = 1,2\ldots TCN$
e) Copy $CAb(t, best)$ to $Ab(t, n)$

Step 5: Check the pre-specified stopping condition. If it is satisfied, terminate the search process, and return the overall best solution as the final solution. Otherwise, go to step 2.

In human bodies, it is unreasonable to assume that the mutation for antibodies with the same affinity is always the same. So, a mutation scheme is proposed based on randomized ranking in step 3. The number of mutation operations performed is defined as $round(RAb(t,a)*\text{rand}(0,1)*m)$. A random factor is added to introduce variations in the mutation for the antibodies of same affinity. Meanwhile, the ranking among the selected antibodies is still used as a guideline to direct the search process to more promising areas, as more clones are produced from better antibodies. Different degree of mutation performed on duplicated antibodies allows different degree of exploitation and exploration from the same solution. This prevents the search process from being trapped in a local optimum easily. Besides, it is important to maintain the antibody representing the overall best solution so far. If an antibody is found to be better than the overall best antibody, it replaces the overall best antibody to become the new overall best antibody. This enables the search process to converge to the global optimal solution regardless of the initial population distribution.

B. AAIS_CX

To further improve the performance of AAIS, a crossover operator is incorporated into the basic algorithm of AAIS to form the AAIS_CX. The procedures of AAIS_CX are outlined as follows:

Steps 1 – 3 are the same as steps 1–3 of AAIS.

Step 4: Select a pool of candidate antibodies by:

a) Copy n_r randomly created antibodies to the pool, assign their ages = 0.
b) Select n_f antibodies from all the $CAb(t-1, a)$ with the following probability:

$$Pb(CAb(t-1,a)) = \frac{sr_{age(CAb(t-1,a))}}{\sum_{i=1}^{TCN} sr_{age(CAb(t-1,i))}} \quad for\ a = 1,2,\ldots,TCN$$

c) Copy $CAb(t, best)$ to the candidate pool

Step 5: Select antibodies from the whole population and assign them to the parent pool by using the 2-antibodies-competition process: select two antibodies randomly, and assign the one with higher affinity to the parent pool.

Step 6: Create children antibodies with crossover operations. The crossover operations used can be any conventional crossover operation, such as EAX and single point crossover.

In this paper, a heuristics based crossover operator is proposed to solve the order picking problem. Its procedures can be illustrated in the following example and in figure 3. A random number r ($0 < r < N$) is selected. The orders located in the sequence position r of parent 1 or parent 2 are candidates to be the first order to be picked in the child antibody. Between these two orders, the one which is feasible and closer to the origin should be selected. Assuming that order A is the 1st order of the child antibody. The orders immediately following order A in parents 1 and 2 are compared. The one which is feasible and closer to order A is selected as the next picking order in the child antibody. If both orders are infeasible, a feasible order is then selected randomly. The process continues until the whole child antibody is formed.
time for job A- job D = 10 units
time to job A- job B = 12 units

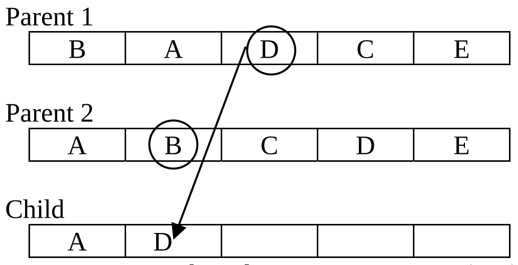


Fig. 3 Heuristics based crossover operator (HX)

Step 7: Build $Ab(t+1)$ by using the following steps:
Select antibodies from the candidate pool in accordance with their survival rate. An antibody can survive to the iteration $t+1$ if the random number generated is smaller than its survival rate corresponding to its age,

$$rand(0,1) <= sr_{age(CAb(t-1,\ a))}.$$

Otherwise, it is replaced by the children antibodies.

Step 8: Check the pre-specified stopping condition. If it is satisfied, terminate the search process, and return the overall best solution as the final solution. Otherwise, go to step 2.

Figures 4 and 5 show the flow of AAIS and AAIS_CX respectively. In general, the age concept is applied in the following areas:

1. Cloning: Under the conventional clonal selection principal, a highly affiliated antibody could produce a relatively large number of clones, which causes an imbalance between exploitation and exploration of the search space, resulting in the trapping of the search process into a local optimum. When

the aging concept is applied to the cloning process, antibodies which have reached a certain age are discarded even though they possess high affinity, thus allowing less affiliated antibodies to be cloned. This results in a better balance between exploitation and exploration of the search space.

2. Survival: The survival probability of an antibody depends on its age. The abandonment of old antibodies with high affinity will provide room for new antibodies to enter the next generation during the search process. This improves diversity and prevents premature convergence.

IV. ONES COUNTING PROBLEM

In order to show the convergence of AAIS and its effectiveness, it is tested on the ones counting problem. Ones counting problem is one of the most common toy problems used to test if an algorithm can converge to the optimum solution from a randomly created initial population [7]. The ones counting problem is simply aimed at maximizing the number of 1s in a bit string x_i of length l:

$$f(x) = \sum_{i=1}^{l} x_i$$

l is set as 100 here. The ones counting problem is a classical test to assess if an evolutionary algorithm is able to reach an optimal solution starting from a randomly initialized population.

Antibodies are coded in binary numbers with a length of 100. In AAIS, the antibodies are subjected to cloning, mutation and selection. The performance of algorithms on this problem is measured by the “Success rate (SR)” SR is calculated as no of times reaching optimum/ total times of running. The SR and the average numbers of evaluated solutions to reach the best value obtained (AES) achieved by AAIS are compared with the best results of opt-IA reported in [7] in table 2. It is also reported in [7] that opt-IA is better than CLONAGE in terms of AES which is an indicator of the convergence rate of the algorithm, however, not in terms of SR. CLONAGE achieves SR of 0 to 100 over different parameters. It is obvious that AAIS can perform better than both, as it can reach SR of 100 at a similar convergence rate. It improves opt-IA by helping the searching escape from local optimum. A sensitivity analysis is also conducted on two important parameters, the total number of cloned antibodies (TCN) and the age limit. The results are shown in table 3. AAIS can perform better than that of the opt-IA in [7] under nearly all the parameter settings. It shows that AAIS is more effective than opt-IA on this problem.

Table 2 SR and AES of opt-IA and AAIS on ones counting problem

Algorithm	Avg SR (best SR)	AES
opt-IA	**95 (97.5)**	66520
AAIS	**100 (100)**	83100

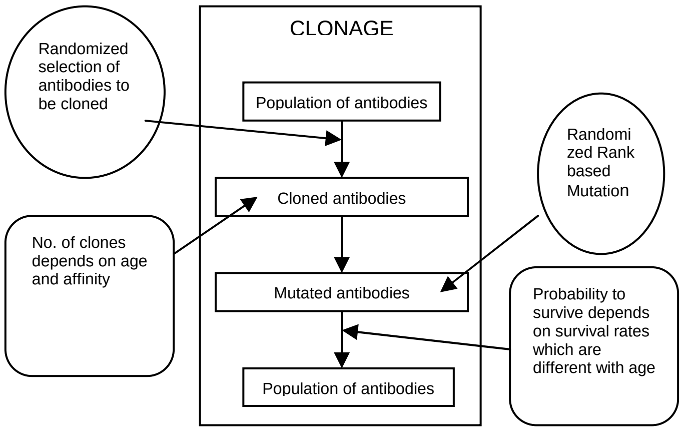
Table 3 Sensitivity analysis of AAIS on the ones counting problem

		AAIS						opt-IA[7]		
TCN		50		100		150		50	100	150
	Age limit	SR	AES	SR	AES	SR	AES	SR		
	6	**75**(99.75)	69600	**67**(99.67)	61230	**75**(99.75)	71820	40	30	30
	8	**60**(99.6)	71616	**80**(99.8)	85600	**80**(99.8)	75296	55	55	50
	10	**100**	73200	**100**	83100	**50**(99.5)	79325	80	70	60

V. SYSTEM ANALYSIS

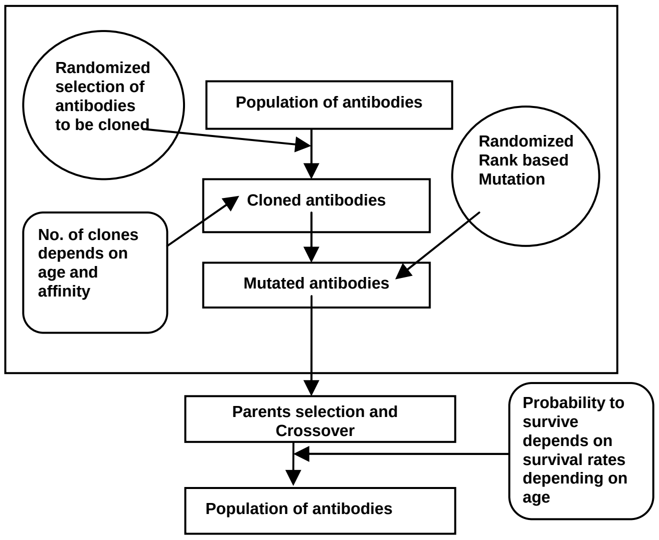
This study seeks to determine the optimal order picking sequence for an AS/RS with multiple I/O stations at an air cargo terminal. An optimal order picking sequence is essential to enhance the operational efficiency of the following three processes: (1) inbound cargoes are unloaded from planes and stored in the AS/RS before they are retrieved for order breaking service or picking up by customers, (2) outbound cargoes either arrive at the terminal in containers or as bulk cargoes, which are then packed together and stored in the AS/RS until they are retrieved to be loaded on planes, (3) some cargoes are reshuffled to better utilize the warehouse space. At each input station, storage orders are handled in a first-come-first-serve basis, and retrieval orders are taken from the rack to a particular output station, while reshuffle orders are moved from one rack to another.

All orders can simply be considered as of the same type but with different starting locations and destinations. It is assumed that the stacker crane is originally located at the origin, (0, 0), i.e. the bottom-left corner of the racks. Hence, the distance traveled in serving the first order is calculated as the sum of the distance traveled from the origin to the starting location of the first order and the distance traveled from the starting location to its destination. The distance traveled in serving any other order is calculated as the sum of the distance traveled between the destination of the preceding order to the starting location of the current order and the distance from the starting location to the destination of the current order. It is also assumed that the stacker crane will return to the origin after serving the last order. Therefore, the extra distance traveled after the last order is calculated as the distance traveled from the destination of the last order to the origin. The time to transfer containers between the crane and the racks is assumed to be negligible. Indeed, this problem can be formulated as a constrained traveling salesman problem (TSP).



CLONAGE
Randomized selection of antibodies to be cloned
Population of antibodies
Cloned antibodies
Randomized Rank based Mutation
No. of clones depends on age and affinity
Mutated antibodies
Probability to survive depends on survival rates which are different with age
Population of antibodies

Fig. 4 Structure and features of AAIS



Randomized selection of antibodies to be cloned
Population of antibodies
Randomized Rank based Mutation
Cloned antibodies
No. of clones depends on age and affinity
Mutated antibodies
Parents selection and Crossover
Probability to survive depends on survival rates depending on age
Population of antibodies

Fig. 5 Structure and features of AAIS_CX

A. Mathematical Model

The following notations are used in the development of the mathematical model.

$t_{\beta k}$	traveling time for the order β_k
t_{last}	traveling time of the extra distance traveled after all orders are finished
β_k	an order to handled by the stacker crane. $\{\beta_1, \beta_2, .. \beta_N\}$ is therefore a sequence of N orders to be handled.
N	number of orders
$W_{g\beta_k}$	horizontal distance traveled to the starting location of order β_k
$H_{g\beta_k}$	vertical distance traveled to the starting location of order β_k
$W_{p\beta_k}$	horizontal distance traveled to the destination of order β_k
$H_{p\beta_k}$	vertical distance traveled to the destination of order β_k
W_{last}	horizontal distance traveled after finishing the last order
H_{last}	vertical distance traveled after finishing the last order
$S_{w\beta_k}$	column index of the starting location of order β_k
$S_{h\beta_k}$	row index of the starting location of order β_k
$D_{w\beta_k}$	column index of the destination of order β_k
$D_{h\beta_k}$	row index of the destination of order β_k
h	height of a rack
w	width of a rack
SP_h	horizontal speed
SP_v	vertical speed
IP_x	The set of storage order from the input station x
$Ind_{x\beta_k}$	Index of storage order β_k in the input station x, i.e. if order β_k is the 1st order at input station x, then $Ind_{x\beta_k}$ will be 1

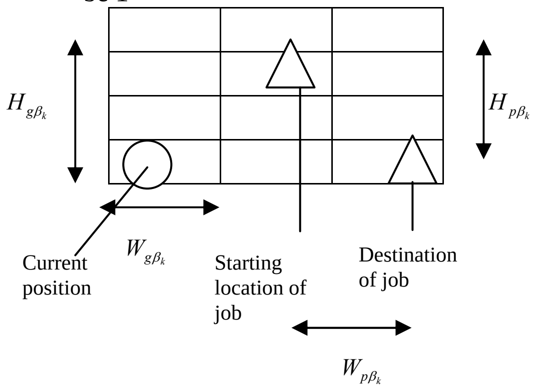


Fig. 6 Illustrations of $W_{g\beta_k}$, $H_{g\beta_k}$, $W_{p\beta_k}$ and $H_{p\beta_k}$

The objective of the model is to minimize the time needed to handle all orders at the AS/RS, including storage, retrieval and reshuffle orders:

$$\text{Minimize } \{ \sum_{k=1}^{n} t_{\beta_k} + t_{last} \} \quad (1)$$

where

$$t_{\beta_k} = \max\left\{\frac{W_{\beta_k g}}{SP_h}, \frac{H_{\beta_k g}}{SP_v}\right\} + \max\left\{\frac{W_{\beta_k p}}{SP_h}, \frac{H_{\beta_k p}}{SP_v}\right\} \quad (2)$$

$$W_{g\beta_k} = w \times \left|S_{w\beta_k} - D_{w\beta_{k-1}}\right| \quad \forall k \neq 1 \quad (3)$$

$$H_{g\beta_k} = h \times \left|S_{h\beta_k} - D_{h\beta_{k-1}}\right| \quad \forall k \neq 1 \quad (4)$$

$$W_{g\beta_k} = w \times \left|S_{w\beta_k} - 0\right| \quad \forall k = 1 \quad (5)$$

$$H_{g\beta_k} = h \times \left|S_{h\beta_k} - 0\right| \quad \forall k = 1 \quad (6)$$

$$W_{p\beta_k} = w \times \left|D_{w\beta_k} - S_{w\beta_k}\right| \quad \forall k \quad (7)$$

$$H_{p\beta_k} = h \times \left|D_{h\beta_k} - S_{h\beta_k}\right| \quad \forall k \quad (8)$$

$$t_{last} = \max\left\{\frac{W_{last}}{SP_h}, \frac{H_{last}}{SP_w}\right\} \quad (9)$$

$$W_{last} = w \times \left|D_{w\beta_k} - 0\right| \quad \forall k = n \quad (10)$$

$$H_{last} = h \times \left|D_{h\beta_k} - 0\right| \quad \forall k = n \quad (11)$$

Subject to

$$b > a \;\; if \;\; Ind_{x\beta_a} < Ind_{x\beta_b} \quad \forall \beta_a, \beta_b \in IP_x, \forall x = 1, 2, 3 \quad (12)$$

Equation (1) shows that the objective function represents the total traveling time of the stacker crane, which consists of two parts: (1) the total amount of time required to handle orders 1 to N, and (2) the amount of time required to travel back to the origin after order N has been completed. Hence, minimizing this objective function is equivalent to maximizing the throughput of the stacker crane which is defined as 3600/ (*objective value* /*N*). Equation (2) calculates the time required to handle one order. It is the summation of the time needed to travel the distance from the destination of the preceding order to the starting location of the current order and the distance from the starting location of the current order to its destination. As the stacker crane can move horizontally and vertically simultaneously, the time is measured as the maximum of the times needed to complete the horizontal movement and the vertical movement, respectively. Equations (3) and (4) calculate the horizontal (vertical) distance traveled by multiplying the width (height) of a rack with the differences of the column (row) indexes for the 2nd to the N^{th} orders. Equations (5) and (6) calculate the horizontal (vertical) distance for the 1st order when the column (row) indexes of the origin are zero. Equations (7) and (8) calculate the horizontal and vertical distances from the starting location of an order to its destination. Equation (9) calculates the time for the stacker crane to travel back to the origin after all orders are finished, which is the maximum of the time given by equations (10) and (11), of completing the horizontal movement and the vertical movement, respectively.

Constraint (12) states that storage orders β_a and β_b, waiting at the same input station with order β_a precedes order β_b, should be handled in a first-come-first-serve manner. As shown in figure 7, it is because the stacker crane cannot access order β_b before order β_a has left the station. Hence, if $Ind_{x\beta_a} < Ind_{x\beta_b}$, order β_a precedes order β_b, its position in the entire order sequence must be larger than that of order β_a, and if $Ind_{x\beta_a} > 1$, order β_a cannot occupy the first position of the entire order sequence.

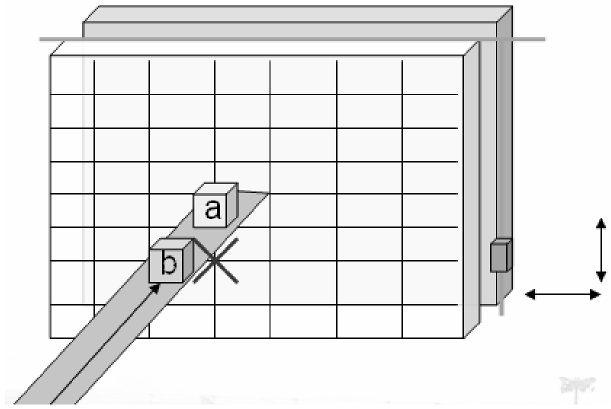


Fig. 7 Graphical Representation of the constraints at the Input station

B. Experiments

The proposed AAIS and AAIS_CX are used to determine the optimal order picking sequence for an AS/RS. To evaluate the algorithms, their performances are compared with that of nearest neighbor heuristics (NNB), genetic algorithm (GA), ant colony system (ACS). To obtain a fair comparison with AAIS_CX, an ant colony system is also added with the same heuristics based crossover operation (ACS_CX) to solve the problem. The pseudo code of ACS_CX is shown in figure 8. In order to satisfy constraint (12), special heuristics is embedded in the procedures of GA, ACS, ACS_CX, AAIS and AAIS_CX to ensure that every solution remains feasible throughout the search process.

```
Procedure ACS_CX
begin
  Initialize the pheromone matrix;
  While not (terminate condition) do begin
  Perform ACS;
  While not (children_pop_size) do begin
      Random selection of parents;
      Heuristics based crossover operations;
      Mutation operations;
      If better than the best solution then
          Replace the best solution;
End while
Global update pheromone;
End while
End
```

Fig. 8 Pseudo code of ACS_CX

In the AS/RS, there is only one aisle, with 16x9 storage racks on each side of the aisle. There are 3 input and 3 output stations located on different floors. The output stations are located at the racks (3, 5), (3, 7) and (3, 9), while the input stations are located at the racks (6, 1), (6, 3) and (6, 7). 18 randomly generated test cases are used in the experiments. The cases consist of 20, 50 and 100 orders. The percentage of storage, retrieval and reshuffle orders are 35%, 55% and 10%, respectively.

In the experiments, ACS and ACS_CX algorithm runs 1000 iterations with 10 ants for cases with 20 and 50 orders, and only 500 iterations for cases with 100 orders to keep the computation time at a reasonable level. As suggested in the literature [15], the parameters q_o and τ_o are set as 0.9 and $1/C^{NNB}$, respectively, β is chosen from the range [2, 5], and both ρ and ξ are chosen from the range [0.1, 0.9]. GA runs 100 iterations with a population size of 100 for test cases with 20 and 50 orders, and 200 iterations for cases with 100 orders to achieve better results. AAIS and AAIS_CX run 500 iterations with a population size of 100, and a clone size of 200 respectively. All the algorithms are programmed in JAVA and run on a Pentium IV 3.2 GHz computer with 512M Ram.

C. Results and Discussion

Tables 4 - 6 summarize the best and the average of the best solutions obtained by running each of the algorithms 10 times, as well as the average of the corresponding computation time needed to achieve the best solutions. Figures 9 and 10 show the convergence behavior of the search processes of GA, ACS, AAIS and AAIS_CX in one typical run.

The results show that, when the number of orders is 20 or 50, AAIS performs better than ACS and GA in 5 out of 10 test cases. However, when the number of orders has increased to 100, it has better performance in all test cases. In addition, AAIS achieves the "optimal" results in the shortest time in most cases. The results also show that AAIS_CX has the best performance among all six algorithms in all test cases, even though it needs longer computation time to derive the "optimal" results when compared with other algorithms. However, the differences become smaller as the number of orders increases. Moreover, it can be seen from figures 9 and 10 that both AAIS and AAIS_CX have better exploration ability in searching for the "optimal" solution. The aging concept prevents the search process of both algorithms from premature convergence, while the clonal selection process keeps exploring areas with promising results. Hence, both AAIS and AAIS_CX are efficient methodologies for solving order picking problems, especially when the number of orders is large.

VI. CONCLUSIONS

In this paper, an Age Artificial Immune System (AAIS) has been proposed. The aging concept is used to govern the cloning and survival of antibodies. To further enhance the performance of the algorithm, a crossover operator is added to AAIS to form the Age Artificial Immune System with Crossover (AAIS_CX). The algorithms have been tested by solving an order picking problem for an AS/RS with multiple input/ output stations. It is shown that the performance of AAIS_CX is better than that of AAIS, GA, ACS and ACS_CX in all test cases. Indeed, the proposed AAIS and AAIS_CX are efficient and effective means for optimizing order picking sequences.

Although AIS has been shown to be efficient in optimization, the parameters of the algorithm, such as survival, clonal rates, and numbers of mutation, need to be fine-tuned for good performance. Therefore, future research can focus on designing algorithms in which the parameters change adaptively to the environment. In addition, the convergence behaviour of the proposed algorithms should also be investigated.

REFERENCES

[1] K. L. Mak, J. S. K. Lau, and X. X. Wang. A genetic scheduling methodology for virtual cellular manufacturing systems: an industrial application. *International Journal of Production Research,* vol. 43, 2005.

[2] L. N. d. Castro and F. J. V. Zuben, Artificial Immune Systems: Part I-Basic Theory and Applications. *Technical Report TR-DCA 01/99,* 1999.

[3] X. Wang, X. Z. Gao, and S. J. Ovaska, Artificial Immune Optimization Methods and Applications - a Survey, 2004

[4] L. N. d. Castro and F. J. V. Zuben, Learning and Optimization Using the Clonal Selection Principle. *IEEE Trans. on Evolutionary Computation,* vol. 6, 2002. 239-251

[5] S. E. Naruaki Toma, Koji Yamada, An Immune Co-evolutionary Algorithm for N-th Agent's Traveling Salesman Problem. *Proceedings 2003 IEEE International symposium on Computational Intelligence in Robotics and Automation,* July 16-20, 2003

[6] V. Cutello, G. Nicosia, and M. Pavone, Exploring the Capability of Immune Algorithms: A Characterization of Hypermutation Operators. *Proceedings of Artificial Immune Systems: Third International Conference, ICARIS 2004, Catania, Sicily, Italy, September 13-16, 2004. Lecture Notes in Computer Science* vol. 3239, 2004. 263-276

[7] G. N. Vincenzo Cutello , Giuseppe Nicosia and Mario Pavone, Clonal Selection Algorithms: A Comparative Case Study Using Effective Mutation Potentials. *Lecture Notes in Computer Science Artificial Immune Systems* vol. *3627: Proceedings of 4th International Conference, ICARIS 2005, Banff, Alberta, Canada, August 14-17, 2005.*

[8] Yaghoub Khojasteh Ghamari and S. Wang, A Genetic Algorithm for Order Picking in Automated Storage and Retrieval Systems with Multiple Stock Locations. *IEMS*, vol. 4, no. 2, 2005, 136-144.

[9] M.H. Han, L.F, McGinnis, J.S. Shieh, J.A. White, On Sequencing Retrievals in an Automated Storage/ Retrieval System, *IIE Transactions*.1987. 56 -66

[10] John J. Kanet and Richard G. Ramirex, Optimal Stock Picking Decisions in Automated Storage and Retrieval Systems. *OMEGA Int. J. of Mgmt Sci.*, vol. 14, no. 3, 1986. 239-244.

[11] O. V. Krishnaiah Chetty and M. Sarveswar Reddy, Genetic Algorithm for studies on AS/RS integrated with machines. *Int J Adv Manuf Technol* vol. 22, 2003. 932-940.

[12] Y.L. Yin and H. Rau, Dynamic selection of sequencing rules for a class-based unit-load automated storage and retrieval system. *International Journal of Adv. Manufacturing Technology*, published online 2005.

[13] H. Felix Lee and S. K. Schaefer, Sequencing Methods for Automated Storage and Retrieval Systems with Dedicated Storage. *Computers ind. Engng*, Vol. 32, No. 2, 1997. 351-362

[14] Jeroen P. van den Berg and A..J. R.M. (NOUD) Gademann, Optimal routing in an automated storage/retrieval system with dedicated storage, *IIE Transactions*, vol.31, 1999. 407-415.

[15] M. Dorigo and T. Stutzle, *Ant Colony Optimization,* United States of America: The MIT Press, 2004

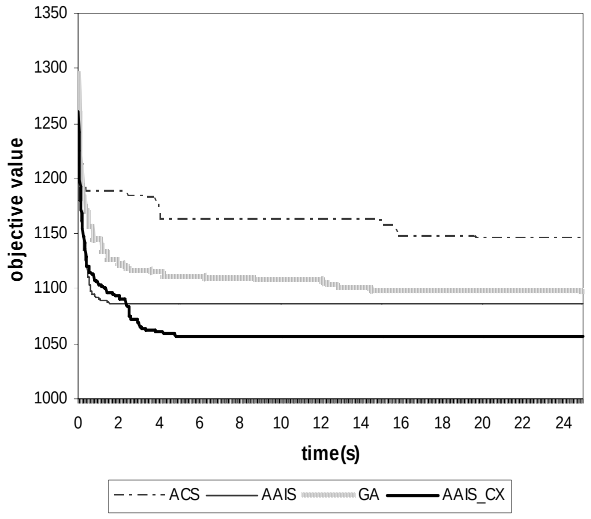


Fig. 9 Best-of-all solutions against time in the case 7 (50 orders) for GA, ACS, AAIS and AAIS_CX

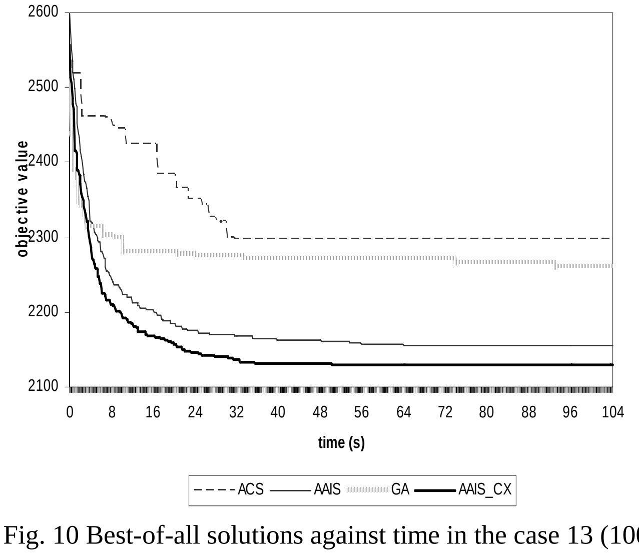


Fig. 10 Best-of-all solutions against time in the case 13 (100 orders) for GA, ACS, AAIS and AAIS_CX

Table 4 Results of NNB, AAIS, AAIS_CX, ACS and GA in cases of 20 and 50 orders

Case /order	NNB	AAIS			AAIS_CX			ACS			GA		
		avg	best	time	avg	best	time	avg	best	time	avg	best	Time
1/20	436	371	370	1.1	**349**	**345**	9.4	359	356	0.9	359	350	5.6
2/20	425	375	374	1.7	**362**	**359**	10.5	376	371	1.1	375	372	5.8
3/20	488	395	394	0.9	**383**	**382**	13.4	394	386	1.2	387	383	7.1
4/20	643	562	549	2.6	**538**	**536**	13.0	564	557	1.1	560	556	5.1
5/20	579	517	512	0.3	**506**	**503**	8.3	522	517	1.2	517	512	8.1
6/20	570	454	446	5.1	**445**	**443**	10.0	477	466	1.1	470	472	5.7
7/50	1212	1080	1065	4.9	**1063**	**1058**	23.0	1117	1102	7.9	1092	1077	20.2
8/50	1203	1096	1094	8.6	**1070**	**1064**	12.2	1144	1140	11.4	1099	1083	24.7
9/50	1175	1043	1033	5.9	**1017**	**1010**	17.1	1060	1043	17.2	1036	1017	23.0
10/50	1430	1375	1338	6.6	**1324**	**1315**	23.9	1356	1338	15.7	1339	1312	20.7
11/50	1355	1259	1250	5.9	**1230**	**1222**	19.7	1278	1265	8.8	1258	1258	16.8
12/50	1220	1171	1152	8.3	**1124**	**1110**	20.3	1160	1137	14.7	1143	1129	25.3

Table 5 Results of NNB, AAIS, AAIS_CX, ACS and GA in cases of 100 orders

Case /order	NNB	AAIS			AAIS_CX			ACS			GA		
		avg	best	time	avg	best	time	avg	best	time	avg	best	Time
13/100	2395	2155	2146	22.3	**2139**	**2132**	38.3	2297	2276	32.2	2263	2257	94.4
14/100	2445	2206	2176	24.2	**2162**	**2156**	36.9	2303	2263	30.5	2292	2280	81.9
15/100	2443	2231	2201	26.4	**2192**	**2173**	47.8	2364	2343	24.5	2337	2322	74.0
16/100	2731	2561	2552	30.1	**2524**	**2520**	46.6	2660	2621	27.3	2621	2605	82.2
17/100	2701	2469	2439	37.6	**2431**	**2419**	43.9	2577	2539	31.8	2529	2471	102.3
18/100	2698	2485	2479	36.4	**2434**	**2427**	42.7	2606	2566	33.7	2522	2508	73.9

Table 6 Results of NNB, AAIS_CX, ACS_CX in cases of 50 and 100 jobs

Case/ job	NNB	AAIS_CX			ACS_CX		
		avg	best	time	avg	best	time
7/50	1212	**1063**	1058	23.0	1078	**1050**	13.8
8/50	1203	**1070**	**1064**	12.2	1079	1066	19.6
9/50	1175	**1017**	**1010**	17.1	1027	1017	18.3
10/50	1430	**1324**	**1315**	23.9	1325	1319	15.9
11/50	1355	**1230**	**1222**	19.7	1257	1248	17.2
12/50	1220	**1124**	**1110**	20.3	1126	1113	17.7
13/100	2395	**2139**	**2132**	38.3	2248	2225	37.3
14/100	2445	**2162**	**2156**	36.9	2271	2254	38.8
15/100	2443	**2192**	**2173**	47.8	2293	2266	34.2
16/100	2731	**2524**	**2520**	46.6	2605	2589	36.3
17/100	2701	**2431**	**2419**	43.9	2462	2430	32.1
18/100	2698	**2434**	**2427**	42.7	2509	2492	36.9