World Academy of Science, Engineering and Technology
International Journal of Mathematical and Computational Sciences
Vol:4, No:5, 2010

Self-evolving Artificial Immune System via Developing T and B Cell for Permutation Flow-shop Scheduling Problems

Pei-Chann Chang, Wei-Hsiu Huang, Ching-Jung Ting, Hwei-Wen Luo, Yu-Peng Yu

Abstract**—**Artificial Immune System is applied as a Heuristic Algorithm for decades. Nevertheless, many of these applications took advantage of the benefit of this algorithm but seldom proposed approaches for enhancing the efficiency. In this paper, a Self-evolving Artificial Immune System is proposed via developing the T and B cell in Immune System and built a self-evolving mechanism for the complexities of different problems. In this research, it focuses on enhancing the efficiency of Clonal selection which is responsible for producing Affinities to resist the invading of Antigens. T and B cell are the main mechanisms for Clonal Selection to produce different combinations of Antibodies. Therefore, the development of T and B cell will influence the efficiency of Clonal Selection for searching better solution. Furthermore, for better cooperation of the two cells, a co-evolutional strategy is applied to coordinate for more effective productions of Antibodies. This work finally adopts Flow-shop scheduling instances in OR-library to validate the proposed algorithm.

Keywords**—**Artificial Immune System, Clonal Selection, Flow-shop Scheduling Problems, Co-evolutional strategy

I. Introduction

COMBINATORIAL optimization problems (COPs) are usually problems with high complexity, numerous algorithms were proposed for applying to this specific problem. Tsai et al. [3] and Chun et al. [2] had mentioned the algorithms for global optimization problems are of increasing importance in modern engineering design and systems operation in various areas. In solving global optimization problems, the particular challenge is that an algorithm may be trapped in the local optima of the objective function when the dimension is high and there are numerous local optima. Genetic algorithms Holland [31] and Goldberg [32], powerful tools based on biological mechanisms and natural selection theory, have received considerable attention regarding its potential as an optimization technique for complex problems and have been successfully applied in various areas. The main feature of the GAs as an optimization method is their implicit parallelism, which is a result of the evolutionary process.

However, there are two major issues in GAs; one is lack of the global search ability and another is the premature convergence. Therefore, numerous of algorithms were proposed for solving the phenomenon. Initially, the improvements in the GAs have been sought in the optimal proportion and adaptation of the main parameters, namely probability of mutation, probability of crossover, population size, and crossover operator. Therefore, some researchers have proposed several GA-based approaches to solve the phenomenon, one of these proposed GA-based algorithm is hybrid Genetic Algorithm and Immune System. The organisms named antibodies in the immune system which are responsible for protecting the body to against harmful organisms named antigen. Campelo [1] mentioned the immune system is able to detect a huge number of antigens using a fairly limited repertory of gene combinations. To carry out this recognition task, segments of genes are combined to accomplish the specificity of almost all the invader antigens known. A self-recognition task keeps the immune system from attacking itself, because immune cells are capable of recognizing themselves.

Also, upon repeated exposure to a certain antigen, the immune system develops more effective and faster responses over time. From an information processing perspective, the immune system can be seen as a parallel and distributed adaptive system. It is capable of learning; it has memory and is capable of associative information retrieval in recognition and classification tasks. Particularly, it learns to recognize patterns, it remembers patterns that it has been shown in the past and its global behavior is an emergent property of many local interactions. All these features of the immune system provide, in consequence, great robustness, fault tolerance, dynamism, and adaptability. These are precisely the properties of the immune system that encourage researchers to try to emulate it in a computer, proposed by Coello [7].

Artificial immune systems (AIS) began in the mid 1980s with Farmer *et al*. [8] and Bersini *et al*. [9] on immune networks. However, it was only in the mid 90s that AIS became a subject area in its own right. Kephart *et al*. [5] published their first papers on AIS in 1994, and Dasgupta [12]

Pei-Chann Chang is with the Department of Information Management, Yuan Ze University, Taoyuan 32026, Taiwan, R.O.C.
(Corresponding Author's E-mail: iepchang@saturn.yzu.edu.tw).
Wei-Hsiu Huang is with the Department of Industrial Engineering and Management, Yuan Ze University, Taoyuan 32026, Taiwan, R.O.C.
Ching-Jung Ting is with the Department of Industrial Engineering and Management, Yuan Ze University, Taoyuan 32026, Taiwan, R.O.C.
Hwei-Wen Luo is with the Department of Information Management, Yuan Ze University, Taoyuan 32003, Taiwan, R.O.C.
Yu-Peng Yu is with the Department of Information Management, Yuan Ze University, Taoyuan 32003, Taiwan, R.O.C.

conducted extensive studies on Negative Selection Algorithms. De Castro and Von Zuben's [30] and Nicosia [29] & Cutello's [4] work (on Clonal selection) became notable in 2002. AIS are new ideas, such as danger theory and algorithms inspired by the innate immune system, are also now being explored. Although some doubt that they are yet offering anything over and above existing AIS algorithms, this is hotly debated, and the debate is providing one the main driving forces for AIS development at the moment.

AIS are computational systems inspired by the principles and processes of the vertebrate immune system. The algorithms typically exploit the immune system's characteristics of learning and memory to solve a problem. AIS is concerned with abstracting the structure and function of the immune system to computational systems, and investigating the application of these systems towards solving computational problems from mathematics, engineering, and information technology. AIS are adaptive systems, inspired by theoretical immunology and observed immune functions, principles and models, which are applied to problem solving [6]. AIS is distinct from computational immunology and theoretical biology that are concerned with simulating immunology using computational and mathematical models towards better understanding the immune system, although such models initiated the field of AIS and continue to provide a fertile ground for inspiration. Finally, the field of AIS is not concerned with the investigation of the immune system as a substrate computation, such as DNA computing.

II. Flow-shop Scheduling Problems

A. Relative Literatures of Flow-shop Scheduling Problems

Base on [11], we can induct flow-shop scheduling problem is one of the most well studied combinatorial problems in the area of scheduling in the operations research. Baker [15] summarized the assumptions of permutation flow-shop scheduling problems. Therefore, most of the research works emerged to develop effective heuristics and meta-heuristics. Chang [10] developed a two-phase sub population genetic algorithm to solve the parallel machine-scheduling problem. In the first phase, the population will be decomposed into many sub-populations and each sub-population is designed for a scalar multi-objective. In the second phase, non-dominant solutions will be combined after the first phase and all sub-population will be unified as one big population. Not only the algorithm merges sub-populations but the external memory of Pareto solution is also merged and updated. Framinan *et al*. [16] reported a review and the classification of the heuristics for permutation flow-shop scheduling problems. Hejazi and Saghafian [18] presented a complete survey of flow-shop scheduling problems and contributions from 1954 to 2004. This survey considered some exact methods, constructive heuristics, meta-heuristics, and evolutionary approaches. This paper is a good reference for n/m/p/Cmax. Ruiz and Maroto [19] provided a comprehensive review and evaluation of permutation flow-shop heuristics. Through our reading these review articles, we found it apparent that heuristics developed for PFSPs have proposed a remarkable contribution.

Heuristics are developed for some specified situations. They may not work in some unexpected situations. Among the meta-heuristics, genetic algorithms have attracted a lot of attention because of many convincing results. Some researches did the pioneering work by applying genetic algorithms in solving flow-shop scheduling problems. Iyer and Saxena [21] proposed an improved genetic algorithm to solve permutation flow-shop scheduling problem. It is not difficult to search for more related genetic algorithm applications in the field of PFSPs. Wang and Cheng [22] considered the two-machine flow-shop problems with setups in a no-wait processing environment to minimize the maximum lateness. Some dominance properties were derived which worked as a heuristic. As a result, the performance of their proposed heuristic is able to work efficiently although those applications are similar in algorithmic structures. For the more detailed information about the review of the flow-shop scheduling, please refer to Ruiz and Maroto [19] who did an extensive comparison in the flow-shop scheduling problems, including tabu search, simulated annealing, genetic algorithms, iterated local search and hybrid techniques.

If the flow-shop scheduling problem with multiple objectives is considered, it becomes even more complicated. Therefore, researchers have started to develop effective heuristics and meta-heuristics to solve this problem. Genetic algorithms have attracted a lot of attention among the meta-heuristics. Jaszkiewicz [23], Ishibuchi *et al*. [24], Reeves [13], Murata *et al*. [25], and Chen [26] did the pioneering works by employing genetic algorithms in solving flow-shop scheduling problems. Reeves and Yamada [20], Zheng and Wang [17], and Chang [27] hybridized other techniques with genetic algorithms to improve the genetic search. Chang [28] studied bi-criterion single machine scheduling with a learning effect by a parametric analysis. From the parametric analysis, a minimum set of Pareto solutions is obtained. It follows that it could be treated as initial solutions for MO algorithms since the parametric analysis doesn't guarantee the algorithm to find out all Pareto solutions.

B. Definition of Flow-shop Scheduling Problems

Flow-shops are useful tools in modeling manufacturing processes. A permutation flow-shop is a job processing facility, which consists of several machines and several jobs to be processed on the machines on different machines. In a permutation flow-shop, all jobs follow the same processing order. Our objective is to find a set of compromise solutions so that the makespan is minimized. The flow-shop scheduling problem is a typical assembly line problem where n different jobs have to be processed on m different machines. All jobs are processed on all the machines in the same order. The

processing time of the jobs on machines are fixed regardless of the order in which the processing is conducted. The problem is characterized by a matrix $P = (p_{ij})$, $i = 1\ldots n, j = 1\ldots m$, of processing time. Each machine processes exactly one job at a time and each job is processed on exactly one machine at a time. The problem then is to find a sequence of jobs of minimizing the makespan which is the completion time of the last job in the sequence on the last machine. If *Ci* denotes the completion time for job *i*, we are trying to minimize max *Ci*. There are many other criteria that can be considered for the purpose of optimization. We refer the reader to Bagchi [14] for a detailed discussion of scheduling using GA. For details of the flow-shop and other scheduling and sequencing problems we refer the reader to Baker. The flow-shop scheduling can be formerly defined as follows: if $p(i, j)$ is the processing time for Job i on Machine j, and a job permutation $\{\pi_1, \pi_2, \ldots, \pi_n\}$, where there are n jobs and m machines, accordingly the completion times $C(\pi_i, j)$ is calculated as follows which are proposed by Reeves [13]:

$$C(\pi_1, 1) = p(\pi_1, 1) \quad (1)$$
$$C(\pi_i, 1) = C(\pi_{i-1}, 1) + p(\pi_i, 1), for\ i = 2, \ldots, n \quad (2)$$
$$C(\pi_1, j) = C(\pi_1, j-1) + p(\pi_1, j), for\ j = 2, \ldots, m \quad (3)$$
$$C(\pi_i, j) = \max\{C(\pi_{i-1}, j), C(\pi_i, j-1)\} + p(\pi_i, j) \quad (4)$$
$$for\ i = 2, \ldots, n;\ for\ j = 2, \ldots, m$$

The makespan is finally defined as:

$$Cmax(\pi) = C(\pi_n, m) \quad (5)$$

Subsequently, the objective is to find a permutation π^* in the set of all permutations $\prod$ so that

$$Cmax(\pi^*) \leqq Cmax(\pi)\ \forall \pi \in \prod \quad (6)$$

A more general flow-shop scheduling problem can be defined by allowing the permutation of jobs to be different on each machine. However, what work has been done to show on the more general flow-shop scheduling problem has tended to small improvement in solution quality over the permutation flow-shop scheduling problems (PFSP) while increasing the complexity of the problem substantially. The size of the solution space increases from *n!* to $(n!)^m$. Other objective functions for the PFSP also received a lot of attention. For example, the mean flow-time (the time a job spends in the process), or the mean tardiness (assuming some deadline for each job) are to be minimized.

Other real problems from the manufacturing industries such as their jobs may have non-identical release dates, and there may be sequence-dependent setup times, and limited buffer storage between machines and so on. These characteristics of the real world problems will make the problem more complicated to be solved within a reasonable time frame. However, GA approaches provide a more realistic view to the problem. Since it can generate alternatives of sequences (in the evolving process, each chromosome represents a feasible solution to the problem) to the decision maker, a more applicable sequence can be decided to solve the current problem with satisfactory results.

III. Self-evolving Artificial Immune System

Permutation Flow-shop sequencing problem as mentioned by Reeves [13] is known to be NP-hard. This kind of problems usually needs high computational time and the solution quality decreases very rapidly when the problem complexity rises gradually. Meta heuristics are usually applied for this kind of problems however they may get trapped in the early convergence problem. AIS in this work consists of two major mechanisms which are T cell and B cell. This paper proposes a architecture to make these two cells cooperate for better solutions, named Self-evolving Artificial Immune System (SEAIS), the architecture chart is shown as Figure 1.

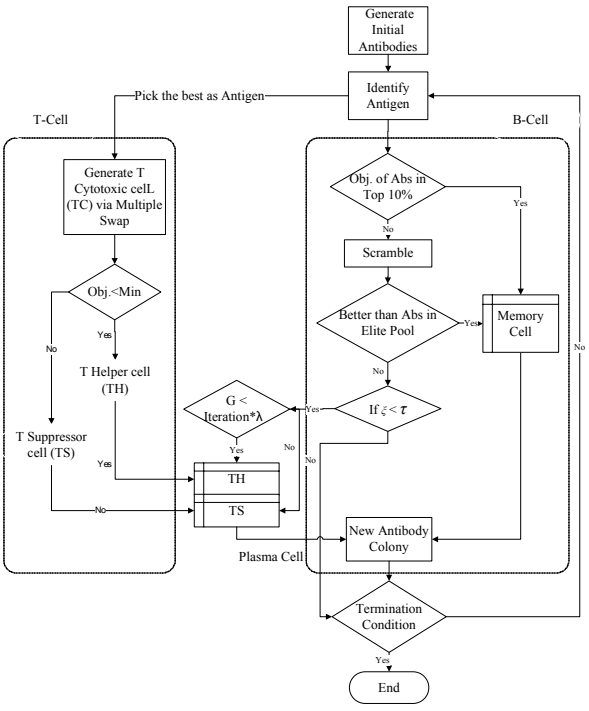

Fig. 1 The Architecture of SEAIS

After the initial Antibodies are generated, the Antibody colonies start to identify Antigen. The Antibody with best objective value will be selected to be the winner in this iteration, which is so-called Antigen in this paper. This phase will decide the candidate to enter T-cell for neighboring searching. The remaining Antibodies will enter the B-cell for widespread searching. Therefore, this research applies the two cells for evolving respectively in Local and Global search. In T-cell, there are three major tasks, which are generating T Cytotoxic cell (TC), or called T killer cell and generating T Helper cell (TH) and T Suppressor cell (TS). TC in T-cell is

responsible producing the cell with the ability of eliminating Antigen. TH and TS are the controlling cells used to adjust based on the convergence pressure. From Figure 2, TH is responsible to stimulate cells active when the evolving iterations have not reach the specific value. TS in T-cell is to restrain the cells' division when the evolving is ineffective. This is called co-evolutional strategy in this work. The purpose of this mechanism is used for the cooperativeness of controlling T and B cell.

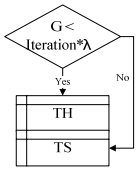

Fig. 2 Co-evolutional strategy for T and B cell

In B-cell, two major tasks are concerned. One is to produce the Memory cell which is used to eliminate when the similar Antigen appears. Another one is to generate Plasma cells, which is used to produce Antibodies. In this research, Plasma cells are from the archive of TH and TS. The Plasma cells and Memory cells will be gathered into an archive stores new Antibody colony. The pseudo codes of SEAIS are described in the following:

```
1. Generate initialize population (Abs)
2. While not terminal condition do
3.  Compute obj_function(Abs)
4.  Assign Ag=Ab with minimal obj_function in Abs
5.  For each Ab in population
6.  If Objective value of Ab superior in population*10% then put into
        a pool
7.  End for
8.  While the stopping criteria do
9.      Select the current Ag to do Swap Mutation
10. If the obj_value of the new_Ab < the current min then put into the
        best_pool
11. Else  put them into diverse_pool
12. End if
13. End while
14. If generation % K ==0 then put best_pool into population
15. Else put diverse_pool into population
16. End if
17. Select the Ab of the population to do Scramble Mutation
18. Compute the Affinity of every Ab
19. If (Affinity of < τ) then put the Ab into population
20. population= some Abs of Scramble Mutation + best_pool or
        diverse_pool
21. End if
22. End while
```

To compute the Affinity between Antibody and Antigen, the diversities should be considered respectively. Based on the definition of Antigen and Antibody of AIS as shown in Gong et al. (2008), two criterions governing the relationship between Antigen and Antibody are explained as follows:

A. Affinity between Antibody and Antibody population

An Antibody is regarded as of a candidate solution of an Antigen. The colony of Antibodies is defined as $A_{bi}=(A_{bi1}, A_{bi2},\ldots, A_{bil})$, and the length of amount is defined as l. A_{bl} is the coding of variable x, which is denoted by $A_{bl}=e(x)$, and A_{bi} is called the decoding of Antibody A_{bil}, expressed as $A_{bi}= e^{-1}(A_{bil})$.

Set A_{bi} is called Antibody space, namely $A_{bil} \in A_{bi}$. An Antibody population is an n-dimensional group of Antibody A_{bil}, where the positive integer n is the Antibody population size.

$$A_{bi} = (A_{b1l}, A_{b2l}, ..., A_{bnl}),\ A_{bnl} \in A_{bi} \tag{7}$$

B. Affinity between Antigen and Antibody population

The affinity between an Antibody and an Antigen is used to evaluate the identification of Antibody for Antigen. The A_b-A_g affinity is defined as ξ_l, which is between Antibody b=e(x) and the Antigen Ag formula (11) is defined as:

$$\begin{aligned}&\xi_l(A_b, A_g) \square F(e^{-1}(A_{b_i}))\\ &= (f_1(e^{-1}(A_{b_i})), f_2(e^{-1}(A_{b_i})),..., f_p(e^{-1}(A_{b_i})))^T\end{aligned} \tag{8}$$

Base on the definition of Affinity above, we set $\delta_{i,j}$ to record the difference between A_g and A_{bi} on specific position of gene strings. We continuously sum up all $\delta_{i,j}$ to be the diversity between A_g and A_{bi} which is regarded as the reciprocal of Similarity (shown as Figure 3).

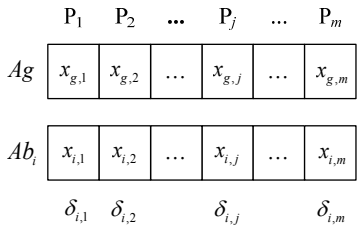

Fig. 3 The schematic diagram for Affinity computation between Antigen and Antibody Colony

Therefore, the greater value of Diversity, the smaller value of Similarity, which means, the increasing value of Affinity represents the value of Diversity decreased. The mathematical models are described as follows:

$$\delta_{i,j} = x_{i,j} - x_{g,j} \qquad for \quad \forall i \in \text{int} \tag{9}$$

$$\xi_i = \left| \sum_{j=1}^{m} \delta_{i,j} \right|^{-1} \tag{10}$$

Affinity ξ is applied to control the injection of T-cell, therefore it should be computed in advance as shown as equation (14). ξ is composed by two differences, one is

between Antibody and the Antigen that is defined as $\delta_{i,m}(Ag_s, Ab)$. Another difference is between the best Antibody and Antibodies that is defined as $\delta_{i,m}(Ab_{winner\,s}, Ab)$.

$$\xi = \sum_{i} \sum_{i,m} [\, \delta_{i,m}(Ag_s, Ab) + \delta_{i,m}(Ab_{winner\,s}, Ab)] \qquad (11)$$

IV. Experimental Results

In Table I, the test instances are rec01~rec11 from Reeves's instances. The objective function is to minimize the Completion times for sequencing *n* jobs on *m* machines are defined as C*, each algorithm test the same instances for comparing the quality of solution.

TABLE I Experimental Result for rec01~ rec11

			SGA		SA		SEAIS	
Instance	n,m	C*	Min	Error Rate	Min	Error Rate	Min	Error Rate
rec01	20,5	1247	1249	0.16	1249	0.16	1249	0.16
rec03	20,5	1109	1111	0.18	1111	0.18	1111	0.18
rec05	20,5	1242	1245	0.24	1245	0.24	1245	0.24
rec07	20,10	1566	1584	1.15	1584	1.15	1584	1.15
rec09	20,10	1537	1567	1.95	1547	0.65	1537	0.00
rec11	20,10	1431	1469	2.66	1453	1.54	1441	0.70

In the result of Table I, the first series of instances rec03 and rec09 are compared with SGA and SA, which are plotted in Figure 4. In this series, the efficiencies are better than SGA.

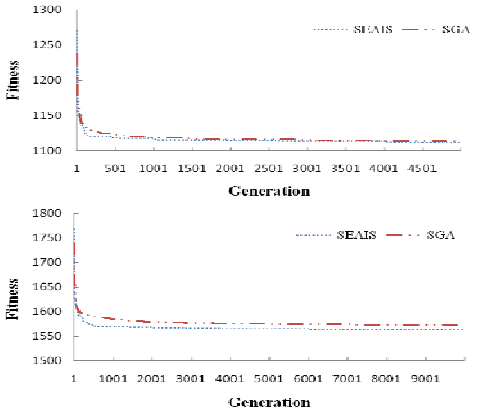

Fig. 4 The convergence effect of rec03 and rec09

The second series is belong to middle complexity of instances, which are from rec13~rec29. From the observation of Table II, the error rates of SEAIS are the smallest among all algorithms. Compare to the result of series 1, the error rates are significantly smaller. From this result, the efficiency of SEAIS is better when the complexities of problems are higher.

TABLE II Experimental Result for rec13~ rec29

			SGA		SA		SEAIS	
Instance	n,m	C*	Min	Error Rate	Min	Error Rate	Min	Error Rate
rec13	20,15	1930	1970	2.07	1966	1.87	1937	0.36
rec15	20,15	1950	1992	2.15	1973	1.18	1966	0.82
rec17	20,15	1902	1963	3.21	1922	1.05	1933	1.63
rec19	30,10	2093	2164	3.39	2158	3.11	2120	1.29
rec21	30,10	2017	2069	2.58	2050	1.64	2050	1.64
rec23	30,10	2011	2082	3.53	2068	2.83	2036	1.24
rec25	30,15	2513	2630	4.66	2599	3.42	2563	1.99
rec27	30,15	2373	2468	4.00	2455	3.46	2417	1.85
rec29	30,15	2287	2392	4.59	2394	4.68	2339	2.27

Fig. 5 shows SEAIS is effective when search the solution in rec13 and rec27.

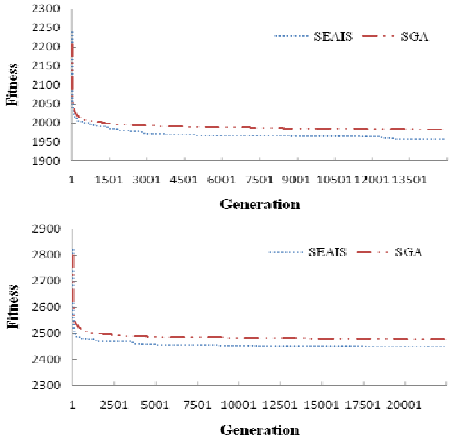

Fig. 5 The convergence effect of rec13 and rec27

The final series of instances are from rec31~rec41, which with the largest complexity. Form Table III, the minimal objective values SEAIS searched are better than the other algorithms. The error rates of SEAIS are also the best among the compared algorithms.

TABLE III Experimental Result for rec31~ rec41

			SGA		SA		SEAIS	
Instance	n,m	C*	Min	Error Rate	Min	Error Rate	Min	Error Rate
rec31	50,10	3045	3210	5.42	3194	4.89	3131	2.82
rec33	50,10	3114	3169	1.77	3177	2.02	3128	0.45
rec35	50,10	3277	3284	0.21	3306	0.88	3277	0.00
rec37	75,20	4951	5281	6.67	5337	7.80	5206	5.15
rec39	75,20	5087	5301	4.21	5396	6.07	5234	2.89
rec41	75,20	4960	5270	6.25	5371	8.29	5180	4.44

In Figure 6, the result shows that SGA is better than SEAIS when meet the instance rec39. Nevertheless, when we check from Table III, we found the error rate of SEAIS is smaller than SGA. This represents that the solution qualities of

SEAIS are better than SGA.

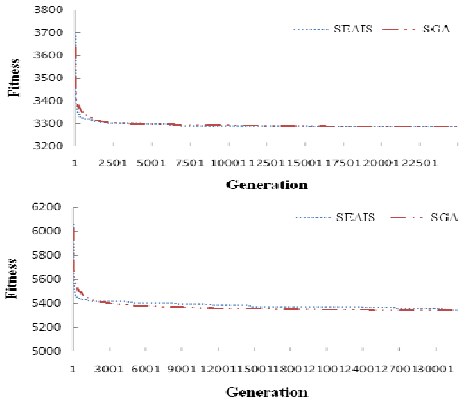

Fig. 6 The convergence effect of rec35 and rec39

V. Conclusion

SEAIS proposed in this paper is validated that it is effective for solving combinatorial problems. In this research, the improvement of the AIS is possible by collecting a set of antibody candidates in the archive is introduced. The experimental results in this paper validate the philosophy of SEAIS has introduced a good mechanism for Clonal selection in AIS. In this work, two mechanisms respectively evolve for local and global search. The proposed co-evolutional strategy can help each other for reasonable searching, such as adjusting the convergence pressure. The extension research on the technique of mining data structure will be done in the near future which will help the T-cell produce more effective Cytotoxic cells to make B-cell gather better Plasma cells for better solutions.

References

[1] F. Campelo, F. G. Guimar˜aes, and H. Igarashi, "Overview of Artificial Immune Systems for Multi-objective Optimization," *EMO 2007, Lecture Notes in Computer Science,* pp. 937-951, 2007.

[2] J. S. Chun, H. K. Jung and S. Y. Hahn, "A Study on Comparison of Optimization Performances between Immune Algorithm and other Heuristic Algorithms," *IEEE Transactions on Magnetics,* vol. 34, No. 5, September 1998.

[3] J. T. Tsai, W. H. Ho, and T. K. Liu, and J. H. Chou, "Improved immune algorithm for global numerical optimization and job-shop scheduling problems," *Applied Mathematics and Computation 194 (2007)* pp. 406-424, 2007.

[4] V. Cutello, G. Nicosia, M. Pavone, J. Timmis, "An Immune Algorithm for Protein Structure Prediction on Lattice Models," *IEEE Transactions on Evolutionary Computation,* vol. 11, no. 1, pp. 101-117, 2007.

[5] J. O. Kephart, "A biologically inspired immune system for computers," *Proceedings of Artificial Life IV: The Fourth International Workshop on the Synthesis and Simulation of Living Systems,* MIT Press. pp. 130–139, 1994.

[6] L. N. de Castro, J. Timmis, "*Artificial Immune Systems: A New Computational Intelligence Approach,*" Springer. pp. 57–58, 2002.

[7] C.A.C. Coello, N.C. Cortés, "Hybridizing A Genetic Algorithm with An Artificial Immune System for Global Optimization," *Engineering Optimization, Volume 36, Number 5,* pp. 607-634(28), 2004.

[8] J. D. Farmer, N. Packard, A. Perelson, "The immune system, adaptation and machine learning," Physica D, vol. 2, pp. 187-204, 1986.

[9] H. Bersini, F. J. Varela, "Hints for adaptive problem solving gleaned from immune networks," *Parallel Problem Solving from Nature,* vol. 496, pp. 343-354, 1991.

[10] P.C. Chang**,** S.H. Chen and K.L. Lin, " Two Phase Sub-Population Genetic Algorithm for Parallel Machine Scheduling problem" *Expert Systems with Applications*, vol. 29(3), pp. 705-712, 2005.

[11] S. H. Chen, "*The Self-Guided Genetic Algorithm,*" PhD thesis, Yuan Ze University, Taoyuan, 2008.

[12] D. Dasgupta (Editor), "*Artificial Immune Systems and Their Applications,*" Springer-Verlag, Inc. Berlin, January 1999.

[13] C. R. Reeves, "A Genetic Algorithm for Flowshop Sequencing," *Computers and Operations Research,* vol. 5, pp. 5-13, 1995.

[14] T. Bagchi, "*Multiobjective Scheduling by Genetic Algorithms,*" Springer ,1999.

[15] K. Baker, "*Introduction to sequencing and scheduling,*" Wiley, 1974.

[16] J. Framinan, J. Gupta, R. Leisten, "A review and classification of heuristics for permutation flow-shop scheduling with makespan objective," *Journal of the Operational Research Society,* vol. 55(12), pp.1243-1255, 2004.

[17] D. Zheng, L. Wang, "An Effective Hybrid Heuristic for Flow Shop Scheduling," *The International Journal of Advanced Manufacturing Technology,* vol. 21(1), pp. 38-44, 2003.

[18] S. R. Hejazi, S. Saghafian, "Flowshop-scheduling problems with makespan criterion: a review," *International Journal of Production Research,* vol. 43(14), pp. 2895-2929, 2005.

[19] R. Ruiz, C. Maroto, "A comprehensive review and evaluation of permutation flowshop heuristics," *European Journal of Operational Research,* vol. 165(2), pp. 479-494, 2005.

[20] C. Reeves, T. Yamada, "Genetic Algorithms, Path Relinking, and the Flowshop Sequencing Problem," *Evolutionary Computation,* vol. 6(1), pp. 45-60, 1998.

[21] S. Iyer, B. Saxena, "Improved genetic algorithm for the permutation flowshop scheduling problem," *Computers and Operations Research,* vol. 31(4), pp. 593-606, 2004.

[22] X. Wang, T. C. E. Cheng, "A heuristic approach for tow-machine no-wait flowshop scheduling with due dates and class setups," *Computers and Operations Research,* vol. 33(5), pp. 1326-1344, 2006.

[23] A. Jaszkiewicz, "Genetic local search for multi-objective combinatorial optimization," *European Journal of Operational Research,* vol. 137(1), pp. 50-71, 2002.

[24] H. Ishibuchi, T. Yoshida, T. Murata, "Balance between genetic search and local search in memetic algorithms for multiobjective permutation flowshop scheduling," *Evolutionary Computation,* IEEE Transactions on vol. 7(2), pp. 204-223, 2003.

[25] T. Muruta, H. Ishibuchi, "Performance evolution of genetic algorithms for flowshop scheduling problems," *Proceedings of First IEEE International Conference on Evolutionary Computation,* 1994.

[26] C. L. Chen, R. V. Neppalli, N. Aljaber, "Genetic algorithms applied to the continuous flow shop problem," *Computers & Industrial Engineering,* vol. 30(4), pp. 919-929, 1996.

[27] P. C. Chang, J. C. Hsieh, C. H. Liu, "A case-injected genetic algorithm for single machine scheduling problems with release time," *International Journal of Production Economics,* vol. 103(2), pp. 551-564, 2006.

[28] P. C. Chang, S. H. Chen, V. Mani, "Parametric Analysis of Bi-criterion Single Machine Scheduling with a Learning Effect," *International Journal of Innovative Computing, Information and Control,* vol. 4(8), pp. 2033-2043, 2008.

[29] V. Cutello and G. Nicosia, "An Immunological Approach to Combinatorial Optimization Problems," *Lecture Notes in Computer Science,* Springer vol. 2527, pp. 361-370, 2002.

[30] L. N. de Castro and F. J. Von Zuben, "Artificial Immune Systems: Part I -Basic Theory and Applications," *School of Computing and Electrical Engineering,* State University of Campinas, Brazil, No. DCA-RT 01/99, 1999.

[31] J. H. Holland, "Genetic Algorithms and the Optimal Allocation of Trials,". *SIAM J. Comput*, vol. 2(2), pp. 88-105, 1973.

[32] D. E. Goldberg, "*Genetic Algorithms in Search, Optimization and Machine Learning*," Addison-Wesley, Reading, MA , 1989.