適用於工程最佳化之免疫演算法

An Immune Algorithm for Engineering Optimization

研究生：闕仲輝（Chung-Huei Chueh）

指導教授：陸冠群（Prof. Guan-Chun Luh）

大同大學

機械工程研究所

博士論文

Dissertation for Ph.D. Degree

Department of Mechanical Engineering

Tatung University

中華民國九十三年七月

July 2004

大同大學

機械工程研究所

博士論文

適用於工程最佳化之免疫演算法

An Immune Algorithm for Engineering Optimization

研究生：關 仲 輝 (Chung-Huei Chueh)

經考試合格特此證明

博士學位論文考試委員

召集人:

指導教授:

所長:

中華民國九十三年七月

An Immune Algorithm for Engineering Optimization

Submitted to the Graduate School of Tatung University
in Partial fulfillment of the requirements for the degree of
Doctor of Philosophy

By
Chung-Huei Chueh

Degree to be awarded and Approved by
Dissertation Oral Committee:

Chairman of Committee

Advisor

Chairman, Graduate School

July 2004

ACKNOWLEDGEMENTS

I express my thanks to my dissertation advisor, Dr. Guan-Chun Luh for his instruction. And I also would like to thank many people for their help and guidance during my years as a graduate student at Tatung University. Most of all, I offer my deepest thanks to my family-who believed in me, and supported me all my life.

ABSTRACT

THIS DISSERATTION focuses on developing a novel immune algorithm called for finding Pareto-optimal solutions simultaneously maintaining diversity to single- and multi-objective optimization problems (SOOPs and MOOPs) based fully on the features of a biological immune system. The applications in this dissertation include unconstrained/constrained test functions and truss-structure sizing multi-objective optimization, structural topology single-objective with multi-modally optimization, and single-objective job-shop scheduling optimization problems. The use of proposed immune algorithm as opposed to the evolutionary algorithm (e.g., genetic algorithm, GA, evolution strategy, ES) provides this methodology with superior diversification and local search abilities. Inter-relationships within the proposed algorithm resemble antibody-antigen relationships in terms of specificity and adaptiveness, antibody clonal proliferation, antigen discrimination, and the antibody memory characteristics of adaptive immune responses. Besides, the features for producing antibodies in biological immune system such as gene fragment rearrangement and several antibody diversification schemes (including somatic recombination, somatic mutation, gene conversion, gene reversion, gene drift, and nucleotide addition) are incorporated into the proposed immune algorithm in order to improve the balance between exploitation and exploration. Moreover the concept of cytokines is also combined to algorithm for constraint handling.

By using several performance metrics and comparison with the other approaches, the effectiveness of proposed immune algorithm are evaluated by unconstrained/constrained test functions and several engineering applications (truss sizing, structural topology, and scheduling). The simulated results demonstrated that the proposed immune algorithm provides better effect than other methods and suitable for searching in optimizations.

摘要

本論文提出一個完全以生物免疫系統為基礎的演算法則-免疫演算法(Immune Algorithm)，並應用於多目標(multi-objective)最佳化、單目標多值域(multi-modal)最佳化與實際工程最佳化設計問題(如: 桁架, 結構拓樸及 scheduling 等)全域最佳解之搜尋。不同於其他演化式演算法，例如遺傳演算法(Genetic Algorithms)、演化策略法(Evolution Strategy)，本免疫演算法具有較佳的多樣性與局部搜尋能力。藉由結合生物免疫系統中適應性免疫反應之特徵，例如抗原與抗體之專一性(specificity)與適應性(adaptiveness) 、抗原識別(discrimination)、抗體之株落增殖(clonal proliferation)、抗體之記憶性(memory)與抗體激素(cytokine)等，以及抗體片段重組和抗體多樣性機制，包含自體突變(somatic mutation)、自體重組(somatic recombination)、基因轉換(gene conversion)、基因倒置(gene inversion)、基因飄移(gene shift)與核甘酸插入(nucleotide addition)等，使得本免疫演算法於最佳化搜尋時，同時兼具全域與局部搜尋之能力，並且能在全域與局部搜尋之間達到平衡。

為了驗證本免疫演算法之搜尋效能，本論文以無限制條件測試函數、具限制條件測試函數、實際工程結構設計等問題進行多目標與單目標多值域最佳解之搜尋。在經由與其他演化式演算法比較後其結果顯示，以本免疫演算法搜尋之結果確實優於其他演算法，同時亦證實本論文所提之免疫演算法適用於最佳化搜尋問題。

TABLES OF CONTENTS

II APPLICATIONS

LIST OF FIGURES

LIST OF TABLES

NOMENCLATURE

Ab_i : *ith* antibody of whole population.

$AbAb_{ij}$: Affinity value between the *ith* and *jth* antibodies.

$AbAg_{ik}$: Affinity value between *ith* antibody and *kth* antigenic epitope.

$\overline{AbAg}_{ik}$: Normalized affinity vakues.

$affinity_{ik}$: Normalized affinity vakue between *ith* antibody and *kth* antigenic epitope.

Ag_k : *kth* antigenic epitope.

$amount_j$: Summation of *jth* antibody violated amount.

av_i : Avidity values which binding of affinities between antigens and antibodies as well as between antibodies only for multi-objective optimization problems.

CK_i : Cytokine value of *ith* antibody, treat as the penalty term for constraint violation.

$count_j$: Total number of the *jth* antibody violated constraint condition.

d_{ij} : The Euclidean distance between the *ith* and *jth* antibodies in objective space.

f_k : *kth* objective function.

$f_k(\mathbf{x}_i)$: *kth* objective value of the *ith* solution.

g_a : The allowable constraint value.

g_j : The equality and/or inequality constraint values.

N_{Ab} : Number of antibodies/solutions.

N_{obj} : Number of antigens/objectives.

N_C : Total number of equality and inequality constraint conditions.

r_i : Rank values represent combinatorial intensity between *ith* antibody and all

antigens.

$\bar{r}_i$: Rank values (r_i) added by constraint violation values (CK_i).

S_i : Similarity among antibodies.

$\mathbf{x}_i$: *ith* solution.

δ_{Ab} : Threshold value which illustrates the allowable difference between antibodies.

CHAPTER 1

INTRODUCTION

1.1 Optimization

The optimization is the process of searching for one or more feasible solutions which correspond to extreme values of one or more objectives in a problem until no other superior solution can be found. When an optimization problem modeling a physical system considering only one objective, the task of finding the optimal solution is referred to as single-objective optimization problems or SOOPs. There exist single-objective optimization methods that work by using calculus-based or deterministic search principles such as gradient-based and heuristic-based techniques and stochastic search principles, which allow optimization method to find globally optimal solutions more reliably including. Evolutionary algorithm and simulated annealing are two of such stochastic methods. While an optimization problem involves more than one objective, the task of finding one or more solutions is known as multi-objective optimization problems or MOOPs. Much of the current focus is on single-objective engineering optimization, even though most real-world problems require that several objectives be satisfied simultaneously. A challenging MOOPs-related problem concerns the goodness of fit of a solution, since all solutions have their own range of fitness values (usually one per

objective). Trade-offs are common, since any solution may be good for some objectives but not for others. The frequency of conflicting objectives has made multi-objective optimization an important aspect of engineering and design.

Over the past decades, numerous approaches such as tabu searches [Hansen, 1997; Gandibleux *et al.*, 1996], simulated annealing (SA) [Suppapitnarm *et al.*, 2000], Ant-Q Algorithms [Mariano and Morales, 1999], fuzzy logic [Rao *et al.*, 1992], neural networks (NN) [Balicki, 1998], and evolutionary algorithms such as evolution strategies (ESs) [Knowles and Corne, 1999] and genetic algorithms (GAs) [Deb, 2001; Zitzler, 2001; Coello, 2002; Osyczka, 2002] have been developed for solving the optimization problems. In which Genetic algorithms — powerful tools based on biological evolution mechanisms and natural selection theory [Goldberg, 1989] — have received considerable attention as the single- and multi-objective optimal design efforts. The genetic algorithms are based on the mechanism of natural selection and evolution and are applied in searching for the global optimum for many applications. They combine survival of the fittest individual among population with a structured and randomized information exchange to form a search algorithm with some of the innovative flair of human search. GAs start from a set of random strings to represent the individuals of population and proceed repeatedly from generation to generation through three basic genetic operators: reproduction (or selection), crossover, and mutation. In each generation, the number of copies of every

individual is reproduced in proportional to its value of fitness function for next generation. Because the value of fitness function represents the probability of survival, the selection procedure keeps strong individuals and eliminates the weak ones to emulate the evolution of nature. The reproduction operator is the source of exploitation. Crossover operator recombines genetic information of two individuals to produce the offspring for the next generatio3n. The main purpose of crossover is to exchange genetic information between parent pairs without losing any important schemata. In short, crossover operator can be viewed as a two-step process. In the first step, the individuals of mating pairs are chosen form the mating pool of population. Then transaction of chromosome segments between mating pairs is performed in the second step. The purpose of mutation is to introduce genetic diversity into the population. A random number is generated for every bit in all chromosomes of the current population and it is checked with the probability of mutation. If the random number is less than the probability of mutation, the selected bit has to undergo mutation, *i.e.*, change from 1 to 0 or vice versa. The total number of bits to be mutated is set by the mutation rate. Both the crossover and mutation operators are the sources of exploration. They will disrupt some of the schemata on which they operate. In the process of genetic search, there is a tradeoff between exploitation (*i.e*. reproduction) and exploration (*i.e*. crossover and mutation). The difficulty of genetic algorithms is seeking the balance between exploitation and exploration that determine the convergence

and diversity of the optimal search. Hence, the genetic algorithms are useful in finding a global optimum in cases where several local optima are present.

Schaffer's (1985) vector-evaluated genetic algorithm (VEGA)—the first GA application developed for solving MOOPs—uses GAs to find multiple trade-off solutions from a single simulation run. Hajela and Lin (1992) designed a Weight-Based Genetic Algorithm (WBGA) for multi-criteria optimization. A domination approach to solving MOOPs was used by Murata and Ishibuchi (1995), and Fonseca and Fleming (1993) in their Multi-Objective Genetic Algorithm (MOGA) and by Srinivas and Debs (1994) to create their Non-dominated Sorting Genetic Algorithm (NSGA). Other approaches based on GAs include the Multi-Niche Crowding Genetic Algorithm (MNCGA) (Rao Vemuri and Cedeno, 1995), Niche Pareto Genetic Algorithm (NPGA) [Horn *et al.*, 1994], Reduced Pareto Set Genetic Algorithm (RPSGA) [Osyczka and Kundu, 1995], Neural Evolution Strategy SYstem (NESSY) [Koppen and Rudlof, 1997], spatial predator-prey model approach (Laumanns *et al.*, 1998), Strength Pareto Evolutionary Algorithm (SPEA) [Zitzler and Thiele, 1998], Hybrid GA [Lo and Chang, 2000], Diploid GA [Viennet *et al.*, 1996], and Multi-Sexual GA [Lis and Eiben, 1997].

1.2 Structural Optimization

The structure optimal design is a very interesting topic in the field of engineering optimization. The optimal design of structures including sizing, shape (*i.e*. configuration) and topology forms the basic issues for the structural design process. In sizing optimization, the parameterized shape and topology are considered as fixed, while an optimal set of sizing parameters, such as the cross-section areas of trusses, are found. With shape optimization, only changes to the boundary conditions of the design can be made with the topology of structure being held constant. Different from shape optimization, topology optimization not only changes structural boundary but also modifies the interior material of structure. In other words, holes in the interior of structure can be created. Hence, the topology optimal design may be the most important and difficult topic in structural optimization. In the structure optimization, optimal design of truss-structures has always been a fast developing area of research in the field of engineering optimization and has made notable progress in the last decade. Numerous techniques and methodologies have been developed to find optimal truss-structures, especially biological-inspired methods imitating natural phenomena and physical processes. Among these are simulated annealing [Moh and Chiang , 2000], particle swarm optimization [Fourie and Groenwold, 2002], evolutionary strategy [Gutkowski *et al.*, 2001], fuzzy logic [Shih and Yu, 1995], immune algorithm [Ishida *et*

al., 1995] and genetic algorithms [Coello and Christiansen, 2000; Narayanan, 1998; Deb and Gulati, 2001; Erbatur *et al.*, 2000; Ponterosso and Fox, 1999; Fadel and Li, 2002], the most famous of these methods being genetic algorithms. Further, most practical design tasks require that the sizing of variables be chosen from a list of discrete commercial values as opposed to continuous values. This results in a discrete optimization problem of greater complexity more difficult to solve using traditional methods [Templeman, 1988; Loh and Papalambros, , 1991; Loh and Papalambros**,** 1991]. However, this is not an issue for genetic algorithm due to their binary-coded nature. Note that GA theory can be equally applied to continuous optimization problems.

Besides, in the past decades a number of innovative approaches to structural topology optimization have been developed. The domain variation (also termed sensitivity analysis) is the first approach proposed by Kibsgaard (1992) for topological optimization. It consists of successive small variations of the initial design domain, and is based on the computation of the gradient of the objective function with respect to the domain. This approach has two major defects: first, it requires a good initial guess, as it demonstrated to be unstable for large variations of the domain; second, it does not allow modification of the initial domain topology (e.g. add or remove holes). Another popular method, the homogenization method [Bendsøe and Kikuchi, 1988; Suzuki and Kikuchim, 1991; Tenek and Hagiwara, 1993; Lin and Chou, 1999] first proposed by Bendsøe and

Kikuchi (1988) consists in dealing with a continuous density of material. In the end of this method, the final density is forced toward value 1 or 0 (material present or absent). However, this approach requires the design of the homogenized operator, as thoroughly described in Allaire and Kohn (1993), and is insofar limited to the linear elasticity case. In addition, it cannot address loadings that apply on the actual boundary of the shape to be determined, and hardly handles optimization for multiple loadings [Kane and Schoenauer, 1996]. Recently, a simple approach to shape and topology optimization termed Evolutionary Structural Optimization (ESO) method has been developed by Xie and Steven (1993). The original concept of ESO method is to gradually remove lowly stressed elements not needed from the structure after each finite element analysis, the element removal criteria is established by sensitivity analysis. Hence, the topology of the resulting design is gradually improved to achieve the optimal design. A fundamental potential drawback of this method pointed out by Liu *et al.* (2000) is the strong dependence of the solution on the mesh of finite element from which it is evolved and on the sequence of the element removal. Although the capability to add or reinstate elements has recently been added to the ESO through the Bidirectional Evolutionary Structural Optimization (BESO) method [Querin *et al*, 1998], this addition is still restricted to previous element positions or to the area/volume predefined by the mesh of finite element.

A possible approach to overcome these difficulties of topological optimization mentioned above is to adopt stochastic optimization methods such as the simulated annealing [Kirkpatrick *et al.*, 1983], the genetic algorithms [Goldberg and Samtani, 1986] and the immune algorithm [Bersini and Varela, 1991]. Anagnostou *et al*. (1992) developed a simulated annealing based approach for structure optimal configuration design. More recently, a lot of researchers have extensively employed genetic algorithm based methods for structural optimization in the optimal design of discretized trusses sizing [Rajeev and Krishnamoorthy, 1992; Wu and Chow, 1995], shape [Jenkins, 1991; Woon *et al.*, 2001], and topology [Kane and Schoenauer, 1996; Chapman, 1994; Jakiela, 2000].

1.3 Job-Shop Scheduling Optimization

The job-shop scheduling problem (JSSP) is one of the well-known NP-hard combinatorial optimization problems. The problem can be described as: there are a list of *j* jobs and a number *m* of machines that perform operations on jobs. Each job involves a particular collection of tasks, and each task needs to be performed on a given machine for a given period of time. In general, the task of scheduling is the allocation of jobs over time when limited resources are available, where the objective should be optimized and constraints must be satisfied. There are several constraints on jobs and

machines [Blazewicz et al., 1996]: *i*) there are no precedence constraints among operations of different jobs; *ii*) operations cannot be interrupted and each machine can handle only one job at a time; *iii*) each job can be performed only on one machine at a time. While machine sequence of the jobs is pre-assigned, the problem is to find the job sequences on the machines which minimize the makespan, *i.e.* the maximum of the completion times of all operations. Since the processing time and constraints are fixed, and no stochastic occur, the search space consists of $(j!)^m$ feasible schedules.

During the last three decades, various approaches have been applied to solve JSSP, including the following: mathematical programming (linear programming, goal programming, dynamic programming, etc.), branch-and-bound methods, and some heuristic/probabilistic search methods. It has been recognized that scheduling optimization using mathematical programming is very difficult, because of lengthy computational time. In addition, several branch-and-bound methods [Applegate and Cook, 1991; Brucker *et al.*, 1994; Carlier and Pinson, 1989] have been developed for solving the JSSP to optimality. These methods require a large amount of computation time and therefore it become more difficult to achieve an optimal solution when the variety of parameters (*i.e.* jobs or machines) and constraints is incremented. In recently years, there has been an increasing interest and growing rapidly in methods based on heuristic such as simulated annealing (SA) [Van Laarhoven *et al.*, 1992; Kolonko, 1999],

tabu search (TS) [Dell' Amico and Trubian, 1993; Ponnambalam *et al.*, 2000], neural network [Foo *et al.*, 1995], and genetic algorithms (GAs) [Davis, 1985; Cheng *et al.*, 1996; Maturana *et al.*, 1997; Murata *et al.*, 1996; Croce *et al.*, 1995; Wang and Zheng, 2002], which are capable of producing goodness solutions with a reasonable computational effort. In the past few years, GAs have been widely applied in the production of scheduling field. A GA exhibits parallelism, contains certain redundancy, and historical information of past solutions, and is suitable for implementation on massively parallel architecture.

1.4 Summary

Even though, genetic algorithms are considered powerful in terms of global optimization, but they have several drawbacks regarding local searches. Tazawa *et al.* (1996) identified two of them as *i*) lack of local search ability, and *ii*) premature convergence. A number of researchers have experimented with optimization approaches inspired from biological immune system to overcome these particular drawbacks implicit in genetic algorithms. Biological immune system (IS) is responsible for protecting the living body against the foreign antigens and other toxins that may be harmful. It exhibits abilities to specificity, learning and memory, and adaptation and discrimination, and presents as a remarkable natural defense mechanism. The immune system

eliminates the harmful materials or foreign antigens mainly by producing soluble antibodies, which recognize and then bind the molecules of foreign antigens. In addition, the immune system is capable of remembering infection, hence, a second exposure to identical or similar antigen is deal with more efficiently. For these reasons, and many others, the biological immune system can be viewed as a mechanism of vast potential for inspiration in variety of domains. Based on the features of a biological immune system, a new biologically inspired technique, so-called artificial immune system (AIS), has been developed for a computational tools and applied to a myriad of computational scenarios during the recently years. The applications of AIS are various including pattern recognition and classification [Carter, 2000], search and optimization methods [Mori *et al.*, 1993; Bersini and Varela, 1994; Hajela and Yoo, 1999; Hajela *et al.*, 1997; Hajela and Lee, 1996; Endoh *et al.*, 1998; Luh *et al.*, 2003; Luh and Chueh, 2004], fault diagnosis and anomaly detection [Aisu and Mizutani, 1996; Dasgupta and Forrest, 1999], machine learning [Hunt and Cooke, 1996], control [Krishnakumar, 1996], scheduling [Fukuda, 1993; Tomoyuki, 2003], nonlinear system identification [Luh and Cheng, 2001], robotics [Jun *et al.*, 1999; Luh and Cheng, 2002], data mining [Knight and Timmis, 1999], computational security [Kephart, 1994; Kim and Bentley, 1999], and so on.

Based on these research efforts in the field of search and optimization methods, Bersini and Varela (1991) proposed a genetic immune recruitment mechanism (GIRM) to

improve GA's local search ability. However, this mechanism takes no measures to counteract premature convergence. Mori *et al*. (1993) developed an immune algorithm using the sharing-like method of GA to prevent premature convergence, but it has no control mechanism to balance between the local search and the global search. Chun *et al.* (1999a) used an immune algorithm for optimizing the shape of electromagnetic devices. Tazawa *et al.* (1996) proposed an immunity-based genetic algorithm (IGA) with improved and faster global convergence, and Hajela *et al*. (1997) followed up with a separate GA-based biological immune system model for enhancing the convergence characteristics and constraints associated with the use of GAs for structural optimization. Several researchers, including Fukuda *et al*. (1998) and Chun *et al*. (1999b), have attempted to apply immune algorithms (IAs) to multimodal and multi-objective optimization problems. Chun *et al*. used an IA to search for diverse solutions to design problems for electromagnetic devices, with optimal solutions aggregating in memory cells. In their modification of a GA-based search procedure for solving MOOPs in a structural system, Yoo and Hajela (1999) made use of a utility function and weighting mechanism to convert a multi-criteria problem into a single-objective problem. It is important to emphasize, however, that a genetic algorithm serves as the framework for all of the hybrid approaches mentioned in the above literatures. The basic role of an immune algorithm is to support diversity via different levels of inter-antibody; even

though natural immune systems have a powerful capacity to diversify, to learn, memorize, and process information, and to discriminate between self and non-self when reacting to foreign pathogens [Dasgupt and Forrest, 1999; de Castro and Jonathan, 1999; Coren *et al.*, 1999].

To highlight the significant features of immune systems, a novel immune algorithm based fully on imitating of biological immune system has been developed in this dissertation for the purpose of optimal searching in the various optimization fields including single-objective, multi-objective, and multi-modally optimizations with different solution encoding system such as one-dimensional & two-dimensional binary-encoded string, and not-bit string (or integer) representations. Within the field of multi- objective optimizations, numerous unconstrained/constrained test functions suggested by Zitzler (1998) and Deb *et al.* (2001) were performed to validate the significant effectiveness of the proposed immune algorithm, with Pareto-optimal solution performances quantitatively measured by five performance metrics. Via using several performance metrics and comparison with different evolutionary approaches, the results indicated that the proposed immune algorithm in the field of multi-objective optimization (named **MOIA**) generally performs better than SPEA (strength Pareto evolutionary algorithm), MOGA (multi-objective genetic algorithm), NPGA (niche Pareto genetic algorithm), and NSGA (non-dominated sorting genetic algorithm) for these test functions.

For the field of constrained multi-objective and multi-modally optimizations, two revised immune algorithm named CMOIA (Constrained MOIA) and MMIA (multi-modal immune algorithm) have also been proposed for the optimal searching in multi-objective truss-structure sizing optimizations and single-objective multi-modal structural topology optimizations considering constraints. By comparison with some other approaches, the results shown that proposed immune algorithm is capable of finding accurate results and keeping the diverse of the solutions. Finally, in the single-objective optimization of job-shop scheduling, through the validation from several benchmark problems with different number of jobs and machines, the proposed immune algorithm is also suitable in such scheduling optimization.

1.5 Structure of the Dissertation

This dissertation is divided into eight main Chapters. **CHAPTER 1** introduces the goal and purpose of the dissertation, and also depicted its structure. **CHAPTER 2** reviews a large number of works from the literatures which are most related to this dissertation research. In **CHAPTER 3**, a general overview of How Biological Immune System Works is presented, considering its anatomy, molecules, organs, and main cells. In addition, the proposed immune algorithm and its nine major steps are presented detail in the **CHAPTER 4**. The scheme of these steps such as the mechanisms of gene

rearrangement and antibody diversity was inspired by biological immune system. **CHAPTER 5** depicted the applications of proposed immune algorithm to the multi-objective optimization with antibody/solution represented by a one-dimensional binary-encoded string. The applications in this chapter including unconstrained/constrained numerical test function optimization as well as two truss-structure sizing multi-objective optimizations considering 10-bar plane truss with continuous design variables and 25-bar space truss with discrete design variables, both sizing optimizations subjected to the maximum allowable stresses. The simulated results are compared with other algorithms and discussed in the rear of this chapter. An application to the single-objective with multi-modal structural topology optimization using immune algorithm is depicted in **CHAPTER 6**. Two 2-dimensional asymmetric topology problems subjected to constrained stresses are optimized in this application. Different antibody representation from applications in the CHAPTER 5, the antibody is represented by two-dimensional binary-encoded matrix. The single-objective job-shop scheduling optimizations for proposed immune algorithm are illustrated in the **CHAPTER 7**. In this chapter, the antibody is represented by the not-binary encoding string (*i.e.* integer encoding). Several benchmark test problems with different number of jobs and machines were calculated and compared in this chapter. Finally, **CHAPTER 8** makes overall conclusions in proposed immune algorithm.

CHAPTER 2

LITERATURE REVIEW

2.1 Introduction

This dissertation focuses on developing a novel immune algorithm for optimal search in the areas of numerical function, structure, and scheduling. In each of these areas, there is an immense body of literature. Hence, this chapter reviews the prior work in these areas that is most related to this dissertation research and organized as follows. Section 2.2 reviews the work on the artificial immune systems (AIS) which are applied to optimization problems. In section 2.3, the literature review is focused on the evolutionary approaches in the field of multi-objective optimization.

2.2 Artificial Immune System

This section reviews the works on artificial immune systems specially designed to solve constrained, multi-modal, multi-criteria/multi-objective, and combinatorial optimization problems.

Bersini and **Varela** (1991) developed a search technique based on the features of network sensitivity and metadynamics to apply to the function optimization. This approach consists of an affinity measure and a fitness function. The affinity measure

was used to evaluate the degree of similarity among individuals of the population, while the fitness function was responsible for evaluating the quality if each individual in relation to the environment. Noted that the adopted of measuring the similarity among individuals is a process similar to fitness sharing in genetic algorithm. In addition, individual candidates suffered genetic operators by crossover and mutation borrowed from evolutionary algorithm. The authors shown results on a simple problem using a binary Hamming distance among individuals and normalized fitness function. The results were presented by comparing their approach with the standard genetic algorithm (SGA). Besides, the authors also offered a genetic immune recruitment mechanism (GIRM) which introducing clonal selection of immune system into genetic algorithm to improve the local search ability of genetic algorithm, but failed to add preventive measures against premature convergence.

Mori *et al.* (1993) proposed an immune algorithm for a multi-modal function optimization hybridizing ideas from idiotypic network theory, immune diversity, clonal selection, and genetic algorithm. Their algorithm is based on an entropy measure to maintain the diversity of a receptor of antibodies. Sharing and genetic operators – crossover and mutation are used to promote genetic recombination and variation in the antibody and prevent the premature convergence, but it has no control mechanism to balance between the local search and the global search. The algorithm is of general

purpose as a hybrid of an evolutionary and immune-inspired approach. Its applications have been several, from function optimization to scheduling.

Tazawa *et al.* (1996) proposed an immunity genetic algorithm (IGA) combining immune system (IS) with genetic algorithm (GA). Authors highlight two mechanisms of IS – the clonal selection and idiotypic network. IGA has a fixed number of solutions and generates new solutions as to the GA by using crossover and mutation operators. After new solutions are generated, IGA selects solutions that form new population like clonal selection of IS. Besides, IGA divided a population into several subpopulations and controls the number of similar solutions like idiotypic network in order to balance between the local and global search. The algorithm was applied to floorplan design problem of VLSI layout and compared the results with those of GA.

Fukuda *et al.* (1998) proposed an immune algorithm (IA) based upon the somatic theory and network hypothesis of immune system (IS) to solve the multi-modal function optimization problem partly using a genetic algorithm. The somatic theory describes that somatic recombination and mutation contribute to increase the diversity of antibodies. The network hypothesis describes that a mutual recognition network among the antibodies contributes to control of the clonal proliferation. The proposed algorithm is shown to be effective for searching for a set of solutions as well as local solutions. Test functions with multi-peak and Shubert function are illustrated to show the abilities of

immune algorithm for multi-modal optimization.

Chun *et al.* (1998) applied a slightly modified of the immune algorithm developed by Mori *et al.* (1993) to several function optimization problems and compared its performance with that of evolution strategy and genetic algorithm. In addition to apply to function optimization, author also applied his algorithm to the optimal design of a surface permanent magnet synchronous motor and a pole shape of an electromagnet [Chun *et al.*, 1997], and compared the performance with other methods. Based on their results, the author claimed that the modified immune algorithm is very suitable for solving multi-modal optimizations.

Hajela and **Yoo** (1999) took inspiration from the immune system to address several problems in optimal design: *i*) how to enhance the convergence speed of genetic algorithm (GA), *ii*) how to handle constraints in a GA-based search, and *iii*) how to adapt the GA search to large scale design problem. For these reasons, author proposed an algorithm combined with capabilities of pattern recognition and adaptation in immune system to improve the performance of GA in structural optimization problems. Like the majority of GA applications, authors used a binary encoding for the strings representing the immune components, *i.e.* a binary Hamming distance. The antibodies corresponding to the unfeasible designs, while the antigens were equivalent to the feasible ones. The goal of the algorithm was to adapt the unfeasible antibodies to feasible antigen, so as to

reduce the constraint violation of GA-based search. The fitness of an individual was determined by its ability to recognize either a specific or a broad group of antigens, given by a function that measure the number of matching bits between a pair of strings. Thus, affinity was measured by similarity instead of complementarity. The algorithm was applied to several tasks, including the optimal design of a 10-bar truss structure for minimum weight and with pre-defined allowable on maximum stresses of tension and compression in the bar elements.

Toma *et al.* (1999) proposed an algorithm based on the immune network and MHC peptide presentation. The immune network was used to produce adaptive behaviors for the n-TSP agents, and antigenic presentation by MHC molecules was employed to induce competitive behaviors among these agents. The agents processed a sensor, mimicking MHC peptide representation by macrophages. T cells were used to control the behavior of agents and B cells were used to produce behaviors. The system operated as follows: first macrophages acquired a city number at random and presented to the B- and T-cells. If a T cell recognized this number, it tried to help B cell by sending stimulatory signals. If B- and T-cells both recognized the same number, the B cell produced an antibody and traveled, then MHC was changed. This representation was based on an integer shape-space, and the affinity of each agent with the environment was directly proportional to the distance traveled by the agent.

2.3 Evolutionary Algorithms

In this section, the reviewed literatures are focused on several evolutionary algorithms which are most commonly used in multi-objective optimization problems (MOOPs), such as VEGA, WBGA, MOGA, NSGA, NPGA, and SPEA and so on. In the implementation of MOOPs, the pioneering work of applying evolutionary algorithm into multi-objective optimization problems is implemented by Schaffer (1985) with his algorithm named vector evaluated genetic algorithm or VEGA. After Schaffer's VEGA, Goldberg (1989) realized a better implementation of domination principle in an evolutionary algorithm and suggested a new non-dominated sorting procedure. Since an evolutionary algorithm needs one fitness function for reproduction, the aim was to find a single metric from a number of objective functions. Goldberg's suggestion was to use the concept of the domination to assign more copies to non-dominated individuals in a population. Since diversity is another concern, the use of a niching strategy among solutions of a non-dominated class was also suggested by Goldberg. Realizing the potential of a good multi-objective evolutionary algorithm which can be derived from Goldberg's suggestion, at least three independent groups of researchers have developed different various of multi-objective evolutionary algorithm, *i.e.* multi-objective genetic algorithm (MOGA), niched Pareto genetic algorithm (NPGA), and non-dominated sorting genetic algorithm (NSGA). These algorithms differ in the way a fitness is assign to each

individual. In addition to VEGA, MOGA, NSGA, and NPGA, few other evolutionary algorithms have also been reviewed in this section.

VEGA

The first implementation of a multi-objective evolutionary algorithm was suggested by Schaffer (1985) to find a set of non-dominated solutions. He modified the simple genetic algorithm (SGA) with selection, crossover, and mutation by performing independent selection cycles according to each objective. Hence, he called his algorithm as the vector evaluated genetic algorithm or VEGA. VEGA evaluated an objective vector instead of a scalar objective function with each element of the vector representing each objective function. Since a number of objectives have to be evaluated, he divided the population at every generation into O equal subpopulations, and each subpopulation is assigned a fitness based on a different objective function. Then, each of the O objective functions is used to evaluate some members in the population. Even though VEGA uses a simple idea and is easy to implement and has capability of finding non-dominated solutions, it has several disadvantages in maintaining a good spread of solutions and bias towards some solutions in the obtained non-dominated front. In VEGA, a solution is evaluated only with one objective, but all of the others are also important in the context of multi-objective optimization. During the simulation run, solutions near the optimum of

corresponding objective function would be preferred by the operators of selection and crossover in a subpopulation. Such preference takes place in parallel with other objective functions in different subpopulation. Therefore, even in convex search space problem, the operators between individual champion solutions could not find diverse solutions in the population, eventually, the VEGA converges to individual champion solutions only.

MOGA

Fonseca and Fleming (1993), whom first introduced a Multi-objective genetic algorithm (called MOGA), used the non-dominated classification of a GA population for finding non-dominated solutions and simultaneously maintaining diversity in the non-dominated solutions. In the MOGA, differs from a SGA, the fitness is assigned to each solution in the population, while rest operators of the algorithm (*e.g.* stochastic universal selection, single-point crossover, and bit-wise mutation) are the same as that in a SGA. To a solution *i*, its fitness is equal to one plus the number of solutions which dominate solution *i*. In this way, the non-dominated solutions are assigned with a fitness value equal to 1. In order to maintain diversity of among non-dominated solutions, they have also introduced a niche count calculated by summing the sharing function among solutions. Finally, the shared fitness value which reduced the fitness value of each

solution was defined by dividing the assigned fitness value by the niche count. Then, the selection with shared fitness values, crossover, and mutation were applied to create a new population.

NSGA

Among the Pareto-based multi-objective evolutionary algorithm, Srinivas and Deb (1994) have implemented Goldberg's concept (non-dominated sorting) more directly. The idea behind the non-dominated sorting procedure is that ranking selection method is used to obtain good solutions and niche method is employed to maintain stable subpopulation of good solutions. Since, the algorithm is based on the non-dominated sorting procedure, they called this algorithm as the non-dominated sorting genetic algorithm, NSGA. NSGA differs from SGA only in the way the selection operator works, while crossover and mutation operators remain as usual. Once again, the dual objectives in a multi-objective optimization algorithm are maintained by using a fitness assignment scheme which prefers non-dominated solutions and by using a sharing strategy which preserves diversity among solutions of each non-dominated front. The fitness assignment procedure different from MOGA begins from first/best non-dominated set and successively proceeds to dominate sets in current population. Any solution *i* of the first non-dominated set is assigned a fitness equal to its population size. Since, all

solutions in the first non-dominated set are equally important in terms of their closeness to the Pareto-optimal front. Besides, the diversity of each solution is maintained by degrading the assigned fitness based on the number of neighboring solutions (*i.e.* niche count) and sharing function. Therefore, degrading fitness of each solution is evaluated by its niche count and sharing function with a sharing parameter. After the degrading fitness values are assigned, the roulette-wheel selection, crossover, and mutation operators are applied as usual to the whole population.

NPGA

Horn, Nafpliotis, and Goldberg (1994) have proposed a multi-objective genetic algorithm based on the concept of Pareto dominance and they called niched-Pareto genetic algorithm (NPGA). NPGA differs from VEGA, MOGA and NSGA in the selection operator. NPGA use the Pareto domination tournaments instead of non-dominated sorting and ranking selection method in solving multi-objective optimization problems. In this method, a comparison set comprising of a specific number of individual is chosen at random from the population at the beginning of each selection process. Two random individuals are chosen from the population for selecting a winner in a tournament selection. Both individuals are compared with the members of the comparison set for domination with respect to the objective functions. There are two

scenario occurred in this tournament selection: *i*) If one of them is non-dominated and the other is dominated, then the non-dominated one is selected; *ii*) If both are either non-dominated or dominated, a niche count is found for each individual in the entire population. Both individuals which with small niche count is selected. Since, this non-dominance is evaluated by comparing an individual with a randomly chosen population set, the success of this algorithm highly depends on the number of this population set.

WBGA

Hajela and Lin (1992) proposed a weight-based genetic algorithm (WBGA) for multi-criterion optimization. In the WBGA, each individual in a population is assigned with a different weight vector, the weighted sum of the normalized objective function values are then added together with assigned weight vector to calculate the fitness of an individual. Because each weight vector will result in one Pareto-optimal solution, the number of weight vector is governed by the maximum number of desired Pareto-optimal solutions. Besides, a sharing strategy with niche count is proposed by computing the distance metric between two solutions in order to maintain diversity in the weight vector. Therefore, the fitness is degraded by this sharing strategy to calculate the shared fitness value. Since fitness is degraded when using the sharing function concept, the

proportional selection method needed to be used. The crossover and mutation operators are then applied on whole population as usual.

SPEA

Zitzler and Thiele (1998) proposed an elitist evolutionary algorithm, they called the strength Pareto evolutionary algorithm (SPEA). SPEA introduced elitism concept by explicitly maintaining an external population (elite individuals) preserved a fixed number of the non-dominated solutions that are found during beginning of the simulation run. In each generation, newly found non-dominated solutions are compared with the external population and the resulting non-dominated solutions are saved in this external population. In order to restrict the population to over-grow, the size of external population is bound to a limit number. Not all elite individuals can be preserved in the external population when the size of external population exceeds a limit number, elite individuals which are less crowded are kept by using clustering algorithm. Besides, the elite individuals in the external population are also participated in the genetic operators with current population for the help of influencing the population towards good region in the search space. During the assignment of fitness, in addition to the assigning of fitness to current population, fitness is also assigned to the external population. SPEA assigns a fitness to each elite individual *i* of external population first and called this fitness as the strength.

The strength is proportional to the number of individual in current population that an elite individual *i* dominate. Thereafter, the fitness of individual *j* in current population is then assigned as one plus the sum of the strength values of all elite individuals which weakly dominate individual *j*. This fitness assignment provides that a individual with a smaller fitness is better. With the fitness values, a tournament selection is applied the current and external (combination) population to choose individuals with smaller fitness. Thereafter, a crossover and mutation operators are used to create the new population from this combination population.

CHAPTER 3

HOW BIOLOGICAL IMMUNE SYSTEM WORKS

3.1 Introduction

The biological immune system (IS) is a complex of cells, molecules and organs that represent an identification mechanism capable of perceiving and combating dysfunction from our own cells (infectious self) and the action of exogenous infectious microorganisms (infectious non-self) such as viruses, bacteria, and other parasites (so-called invading antigens) [Jerne, 1974]. The most important function of a biological immune system is to protect living organisms from invading antigens. The body identifies foreign antigens through two inter-related systems: the innate immune system and the adaptive immune system. A model of relationship among immune system components is depicted in Fig. 1. Phagocytes, the main cells participated in innate immune system, are white blood cells capable of destroying most of antigens on first contact. The adaptive immune system uses lymphocytes that can quickly change in order to destroy antigens that have entered the bloodstream. A major difference between these two systems is that adaptive cells are more antigen-specific and have greater memory capacity than innate cell. B-lymphocyte (or B-cell) and T-lymphocyte (or T-cell), two main types of lymphocyte, play a significant role in adaptive immune system.

The T-cell matures in the thymus while the B-cell matures in the bone marrow. Cells of the B and T lymphocyte series differ in many functional aspects but share one of the important properties of immune response that they exhibit specificity toward an antigen. Thus the major recognition and reaction functions of the immune response are contained within the lymphocytes. There are two branches of adaptive immunity: humoral immunity and cell mediated immunity that have different sets of participants and different sets of purposes but with one common aim: to eliminate the antigen. These two branches interact with each other and collaborate to achieve the final goal of eliminating the antigen. B-cells are included in the humoral immunity to synthesize antibodies in the process of clonal proliferation once they are activated by antigen and Helper T-cells while T-cells take part in the cell mediated immunity. T-cells do not synthesize antibodies but instead synthesize and release various cytokines that affect other cells. One class of the T-cells, named the Killer T-cells, destroys the infected cell whenever they recognize the infection. The other class that trigger clonal proliferation, stimulate/suppress antibody formation is called the Helper T-cells. A breakdown in any of their activities can result in allergic reactions and autoimmune disease. Lymphocytes float freely in blood and lymph node and patrol everywhere for foreign antigens, then gradually drift back into the lymphatic system, to begin the cycle all over again.

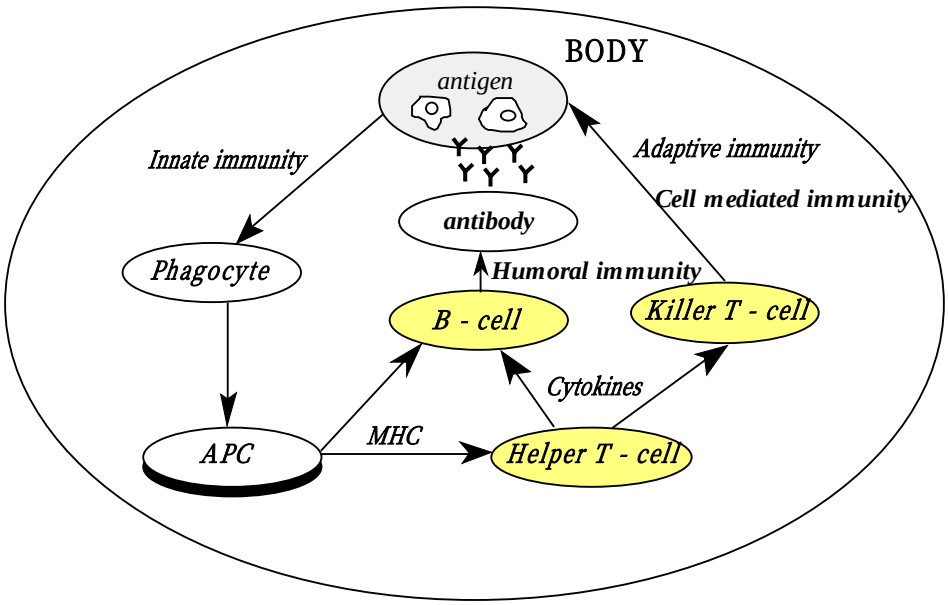


Fig. 1 Illustration of the biological immune system

3.2 Immune System Works

As shown in Fig. 1, when an infectious foreign pathogen attacks the human body, the innate immune system is activated as the first line of defense. Innate immunity is not directed in any way towards specific invaders, rather against any pathogens that enter the body [de Castro and Jonathan, 1999]. Hence, it is so-called non-specific immune system. The most important cells in the innate immunity are phagocytes such as macrophages, monocytes and dendritic cells. Macrophages possess the capability of ingesting and digesting several microorganisms and antigenic particles. Some macrophages have the ability to present antigens to other cells, thus being termed antigen-presenting cells (APCs). The APC interprets the antigen appendage and extracts the features by processing and presenting antigenic peptides on its surface to lymphocytes.

These antigenic peptides are a kind of molecule called MHC (Major Histocompatibility Complex) to distinguish the non-self molecules (infectious non-self) from the those native self molecules (infectious self) and plays a leading role in inducing the expression of co-stimulatory signals in APCs that will lead to T-cell activation, promoting the boost of the adaptive immune system. Moreover, B-cells are also affected by Helper T-cells during the adaptive immune responses. The Helper T-cell plays a remarkable key role for deciding the immune system toward the cell mediated immunity (by Th1 Helper T-cells) or the humoral immunity (by Th2 Helper T-cells) [Roitt and Brostoff, 1998], and connects the non-specific immune response to make a more efficiency specific immune response. The T cells work, primarily, by secreting soluble substances, know as cytokines and their relatives that constitute powerful chemical messengers. Lymphokines or interleukin (IL) are the cytokines secreted by lymphocytes. The cytokines promote cellular growth, activation and regulation. In addition, cytokines can also kill target cells and stimulated macrophages. In the other hand, B-cell becomes stimulated and created antibodies during clonal proliferation in the germinal center when a B-cell recognized an antigen. Recognition is achieved by inter-cellular binding, which is determined by molecule shape and electrostatic charge. The secreted antibodies are the soluble receptor on the surface of B-cell and these antibodies can be distributed throughout the body. As shown in Fig. 2*b*, an antibody-combining site or termed

paratope can bind with an antigenic determinant or termed epitope. Moreover, the immune system produces the diverse antibodies by recognizing the idiotype of the mutual receptors of the antigens between antigen and antibodies and between antibodies. The strength of binding between antigens and antibodies and that amongst antibodies can be evaluated by the value of affinity, or degree of match. In terms of affinities, the immune system self-regulates the production of antibodies and diverse antibodies.

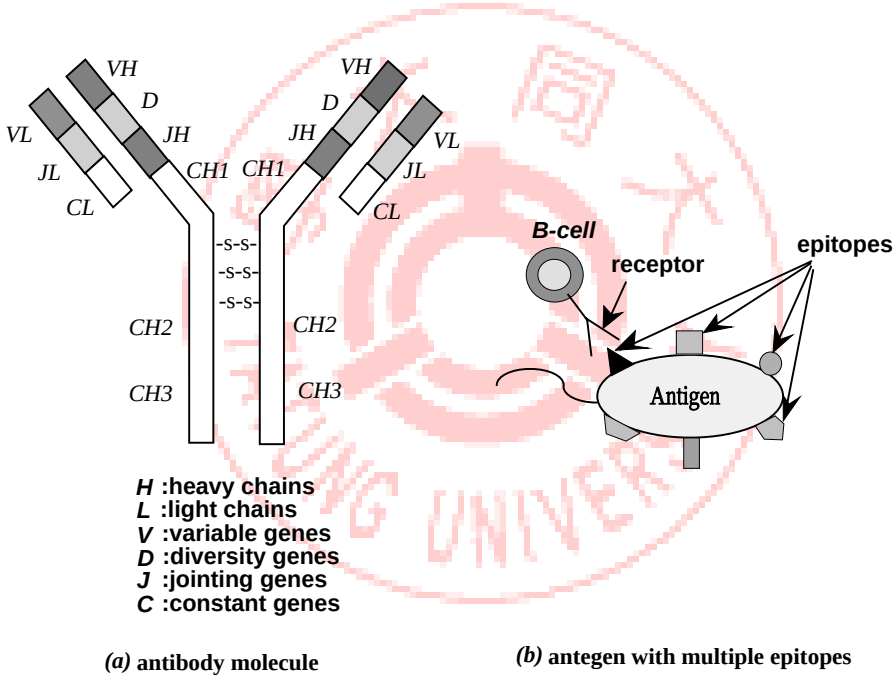


Fig. 2 Antibody molecule and multiple-epitope antigen

3.3 Antibody Structure

One of the major functions of the immune system is the production of soluble proteins that circulate freely and exhibit properties that contribute specifically to

immunity and protection against foreign material. These soluble proteins are the antibodies, or called immunoglobulins (Ig), and expressed as secreted and membrane-bound forms. Secreted antibodies are produced by plasma cells – the terminally differentiated B cells during clonal proliferation within germinal center. Membrane-bound antibody is present on the surface of B cells where it serves as the antigen-specific receptor. The basic unit of an antibody molecule is composed of four polypeptide chains: two identical light chains and two identical heavy chains as depicted in Fig. 2*a*. The grouping of two different types of gene fragments (*VL, JL*) constructs the light chains and the combination of three different types of gene fragments (*VH, D, JH*) forms the heavy chains. In addition, the variable region (*V*-region) is responsible for the antigenic recognition and binding, whereas the constant region (*C*-region) cannot bind antigen, but it is responsible for the biological functions of the antibody molecule after antigen has been bound to the *V*-region. The *V*- and *C*-region of an antibody molecule are coded by different gene fragments. For the purpose of enormous diversity, many different *V*-region genes can be linked up to a single *C*-region. The combining of *V*- and *C*-region gene fragments (rather than having a single gene coding for every individual antibody molecule) significantly reduces the amount of genetic information required to encode different antibody molecules. Additionally, antibody gene fragments could move and rearrange themselves within the genome (inherited DNA) of a

differentiating cell. A *V*-region gene fragment can be located in one position in the DNA of an inherited chromosome (the germ-line DNA), and then move to another position on the chromosome during differentiation. This differentiation brings together an appropriate set of gene fragments for the *V*- and *C*-region. The set of rearranged gene fragments is then transcribed and translated into a complete *H* or *L* chain. Consequently, the genetic materials (gene fragments) required to produce an antibody are encoded in a set of antibody library named germ-line DNA library, each library containing a set of components or fragments of antibodies. Besides, the *V* (variable), *D* (diversity), and *J* (joining) gene fragments are individual libraries that contribute to the production of functional antibody. For each library, those can be created from the lymphocytes of donors who differentiate with higher affinity developed in the immune system. Note that the functional genes of antibody do not exist in the germ-line DNA libraries, except only the gene fragments. A functional gene is generated when germ-line DNA is rearranged randomly.

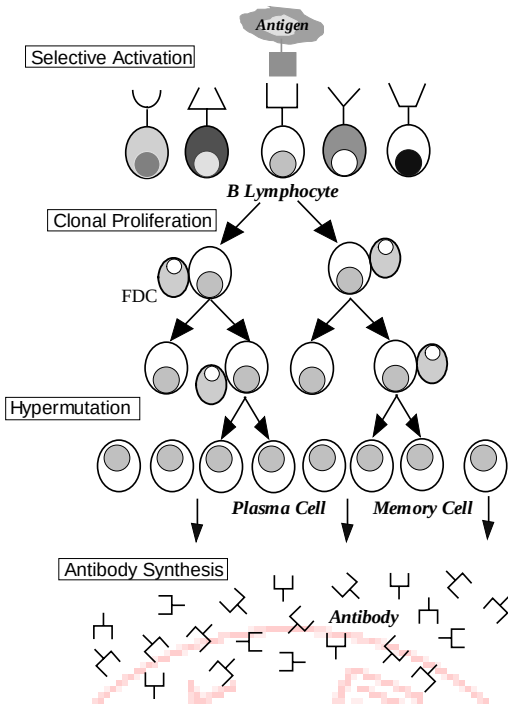


Fig. 3 Clonal selection principle

3.4 Clonal Selection

After binding to antibody receptors, an antigen stimulates the B cell to differentiate and mature into plasma cells and memory antibodies through the process known as clonal proliferation or clonal selection. As shown in Fig. 3, the clonal proliferation of the B cell occurs inside the lymph nodes within a special microenvironment named germinal center where antigenic peptide is presented on the surface of the follicular dendritic cells (FDCs). The proliferated B cells that are able to combine with FDCs survive and become plasma cells to secrete large amount of the same kind antibodies. The principle of clonal selection is the theory used to describe the basic properties of an adaptive immunity to an antigenic stimulus. It establishes the idea that only those cells capable of recognizing an antigenic stimulus will proliferate and differentiate into effector cells, like

the plasma cells. Therefore, the germinal center constantly selects high affinity B cells and simultaneously fosters the B cells apoptosis (a process of cell death) that bind the antigen ineffectively [Krawinkel *et al.*, 1983]. A hypermutation mechanism takes place on the variable region of B cell during the process of clonal proliferation. The hypermutation plays a critical role in creating diverse antibody, increasing affinity and enhancing specificity of antibody. This occurs at an extremely high rate, about 5-6 orders of magnitude higher than the normal mutation rate [Harris *et al.*, 1999]. In addition to differentiating into plasma cells, B cell can as well discriminate into long-lived B memory cells. When a living body is exposed to similar antigens again, the memory antibodies start differentiating into large amounts of lymphocytes capable of producing high affinity antibody by pre-selecting specific antigen [Perelson *et al.*, 1978]. Both mutational and selectional events in B-cell clonal proliferation processes allow these lymphocytes to increase their antibodies diversity and improve their capability to recognize the selective antigens (increasing their affinities with selective antigens). In clonal proliferation, random changes (e.g. hypermutation) are introduced to the *V*-region genes, and occasionally one such change will lead to an increase in the affinity of the antibody. These higher-affinity matured cells are then selected to enter the pool of memory cells. The antibody is not only diversified through a hypermutation process but mechanisms whereby rare B-cells with high affinity mutant receptors can be selected

to dominate the immune response (donor of B-cell). Due to the random nature of the somatic mutation process, a large proportion of mutating genes become non-functional or possibly develop harmful anti-self specificities which attack our own body cells. On the contrary, those cells with low affinity receptors, or the self-reaction cells, must be efficiently eliminated. In terms of affinities, the immune system self-regulates the production of antibodies and diverse antibodies.

3.5 Antibody Diversity

The number of different genes for *V*-region in the germ line constitutes the baseline from which antibody is derived and represents the minimum number of different antibodies that could be produced. How B cells can develop a vast antibody of antigenic specificities. This explained one of the key features of the immune response: diversity—the ability to respond to many different epitopes, even if they had not been previously encountered. Current estimates show that although the human genome contain about 10^5 genes, it is able to produce antibody repertoire that can recognize at least 10^{16} antigens. The enormous diversity of the antibody developed by the immune system is the key to its antigen recognition capabilities. Three major categories are reported to increase the diversity of antibodies: *i*) combinatorial diversity via multiple copies of *V*, *D*, and *J* gene fragments encoded in the germ-line DNA libraries, and

somatic recombination inaccuracy [Roitt and Brostoff, 1998]; *ii*) junctional diversity via small variations in the precise point of juncture of gene fragments and small insertions of nucleotides at juncture sites [Manser *et al.*, 1987]; *iii*) mutational diversity via somatic mutation such as point mutation, short deletions and repertoire shift (gene conversion) which can occur within assembled antibody genes to further expand antibody diversity [de Castro and Von Zuben, 1999].

3.6 Summary

The immune system is a remarkable natural defense mechanism. It exhibits characteristics such as *i*) Specificity: the ability to discriminate among different antigenic epitopes, and to respond only to those that necessitate a response rather than making a random response. *ii*) Learning and Memory: the ability to recall previous contact with a particular antigen, such that subsequent exposure leads to a more rapid and more effective immune response. *iii*) Discrimination between self and non-self: the ability to response to those antigens that are foreign/non-self and to prevent responses to those antigens that are part of own body/self. For these reasons, and many others, the immune system can be viewed as a mechanism of vast potential for inspiration in variety of domains including pattern recognition, optimization, anomaly detection, machine learning, control system, scheduling, fault diagnosis, nonlinear system identification, robotics, and so on.

CHAPTER 4

IMMUNE ALGORITHM

4.1 Introduction

In this dissertation, a novel scheme – Immune Algorithm (IA) based on emulating a biological immune system is developed to solve the optimization problems. Analogous to the biological immune system, the proposed immune algorithm has the capability of seeking Pareto-optimal solutions while maintaining a high-level of diversity in the search space. Corresponding to the optimization problem, the antigens (Ag_k) and antibodies (Ab_i) serve as objectives (f_k) and associated solutions ($\mathbf{x}_i$) in a computational model, respectively, and are expressed as follows:

$$\begin{aligned} Ab_i \equiv \mathbf{x}_i &= \left(x_1, x_2, \cdots, x_{N_{Ab}}\right)_i \qquad i = 1,2,\cdots, N_{Ab} \\ Ag_k &\equiv f_k \qquad\qquad\qquad\qquad k = 1,2,..., N_{obj} \end{aligned} \tag{4.1}$$

$$AbAg_{ik} \equiv f_k(\mathbf{x}_i) \tag{4.2}$$

where Ab_i represents the *ith* antibody of the whole population, or the *ith* solution ($\mathbf{x}_i$) composed of a set of $x_{N_{Ab}}$ design variables ($x_1, x_2, \cdots, x_{N_{Ab}}$), Ag_k represents the *kth* antigenic epitope, or *kth* objective function (f_k); $AbAg_{ik}$ indicates the affinity value between an *ith* antibody and an *kth* antigenic epitope, or equivalently the *kth* objective value of the *ith* solution ($f_k(\mathbf{x}_i)$). N_{Ab} is the number of antibodies/solutions, whereas N_{obj} is the number of antigens/objectives. Fig. 5 illustrates the relative scheme of

antibody/solution ($Ab_i / \mathbf{x}_i$) and antigen/objective ($AbAg_{ik} / f_k(\mathbf{x}_i)$) defined in this algorithm. The antibodies evolve continuously to search for the fittest ones, *i.e.* the most matched with specific antigens.

Besides, similar to the evolutionary algorithms especially the genetic algorithm, the proposed IA starts from a pre-defined number of random strings to represent the antibodies of population and proceed repeatedly from generation to generation through four basic immune operators: *clone*, *donor selection*, *antibody rearrangement*, and *antibody diversity*. In addition, each antibody is classified into several different kind of gene such as light-chain and heavy-chain gene mimicking the structure of antibody in the biological immune system. The *clone* operator proliferates the stimulated antibodies which presented higher combinatorial intensity with antigen in whole antibody population with hypermutation. The hypermutation event only occurred on the gene of light chine (usually defined as lower bits of binary code if binary encoding system is used) in order to prevent excessive discrepancies. After the clonal proliferation, the proliferated antibodies which increasing its combinatorial intensity are defined as mature antibodies and become plasma antibodies and memory antibodies both with identical gene structure, while proliferated antibodies which decreasing its combinatorial intensity are defined as immature antibodies and then neglected. The plasma antibodies will combine with original antibody population and wait for donor selection. The memory antibodies

preserve and update in the memory pool. Besides, a part of memory antibodies will be induced to the germ-line DNA library for offering its gene fragment to construct new antibody. The *donor selection* operator is the source of exploitation/convergence. By using of tournament selection method, the antibodies which presented higher combinatorial intensity with antigen will be selected as donor for constructing the germ-line DNA library. Hence, the members of germ-line DNA library are composed of the memory and donor antibodies. The antibody *rearrangement* operator rearranges the antibody fragment randomly chosen from germ-line DNA library for producing new antibody. The purpose of antibody *diversity* operator is to introduce genetic diversity of antibody into the population thorough somatic point mutation, somatic recombination, gene conversion, gene inversion, gene shift, and nucleotide addition inspired from biological immune system.

Therefore, according to different optimization problem, the antibody can be encoded by one-dimension bit-code string (*e.g.*, test functions and truss-structure sizing multi-objective optimization in chapter 5), two-dimension bit-code string/matrix (*e.g.*, structural topology multi-modal optimization in chapter 6), or not bit-code string (*e.g.*, job- shop scheduling optimization in chapter 7). Once the antibody has been defined, the combinatorial intensity between antibody and antigen is then calculated. In the scenario of multi-objective, the combinatorial intensity between antibody and antigens is

represented by the rank value. While in the scenario of multi-modally optimization, the combinatorial intensity is composed of the objective function value and similarity value among antibodies. However, the combinatorial intensity is replaced by objective function value directly when consider the single-objective optimization. The cytokine value of the antibody is treated as the penalty term for constraint violation if considering the constrained optimization problem. Next, several antibodies which present higher combinatorial intensity (*i.e.*, non-dominated antibodies in the multi-objective optimization or the best antibody in the single-objective optimization) will move to germinal center for clonal proliferation (or so-called clonal selection) for locally improving their combinatorial intensity. Hence, the function of clonal proliferation can be regarded as the effect of local search. After clone process, the matured antibody(s) which combinatorial intensity better than un-proliferated antibody(s) note only return to population (plasma antibodies) for becoming the donor antibodies by tournament selection method, but move to memory pool as the memory antibodies for speeding up the optimal search. For producing new antibodies, different from genetic algorithm which crossover two individuals the antibody is rearranged by using gene fragments chosen randomly from the corresponding gene libraries stored in the germ-line DNA library. However, the germ-line DNA library is constructed by the donor antibodies which express higher intensity with antigen(s) and the memory antibodies. Finally,

several diversification mechanisms (*e.g.* somatic point mutation, somatic recombination, gene conversion, gene inversion, gene shift, and nucleotide addition) inspired by biological immune system are employed in order to match a large variety of antigens and prevent premature. Noted that these mechanisms are randomly adopted in the antibody diversification process.

The corresponding biological immune system, proposed immune algorithm (IA) and genetic algorithms (GAs) terminologies are summarized in Table 1. In the rest of this chapter we will describe detailed the algorithm procedure represented by the flowchart in Fig. 4.

Table 1 Corresponding terminology of biological immune system, proposed immune algorithm (IA), and genetic algorithms (GAs)

Biological immune system	**IA**	**GAs**
Antigen	Objective ($f(\mathbf{x}_i)$)	Objective
Antibodies	Antibodies/solutions ($\mathbf{x}_i$)	Chromosomes/solutions
Antibody structure	Antibody length (bit-string)	Chromosome length
Number of antibody	Antibody size	Population size
Affinities between antigen and antibodies	Affinities/Objective values $AbAg_{ik}$	Fitness values
Affinities between antibodies	Similarity between solutions $AbAb_{ij}$	Distance between solutions
Avidity between antigens and antibodies	Avidity av_i	None
Idiotype value between antibodies	Similarity of solutions S_i	Niche/sharing
Hyper-mutation	Mutation with higher mutation rate	Mutation
Plasma antibodies (in clonal proliferation)	Improved local search	None
Memory antibodies	Pareto-optimal set	None
Germ-line DNA fragment	Schemata	None
Gene fragment rearrangement	Binary-code segment recombination	Parental gene recombination/crossover
Antibody diversification	Six diversification schemes	Mutation operator
None	None	Crossover
Cytokine	Constraint conditions handling	Penalty

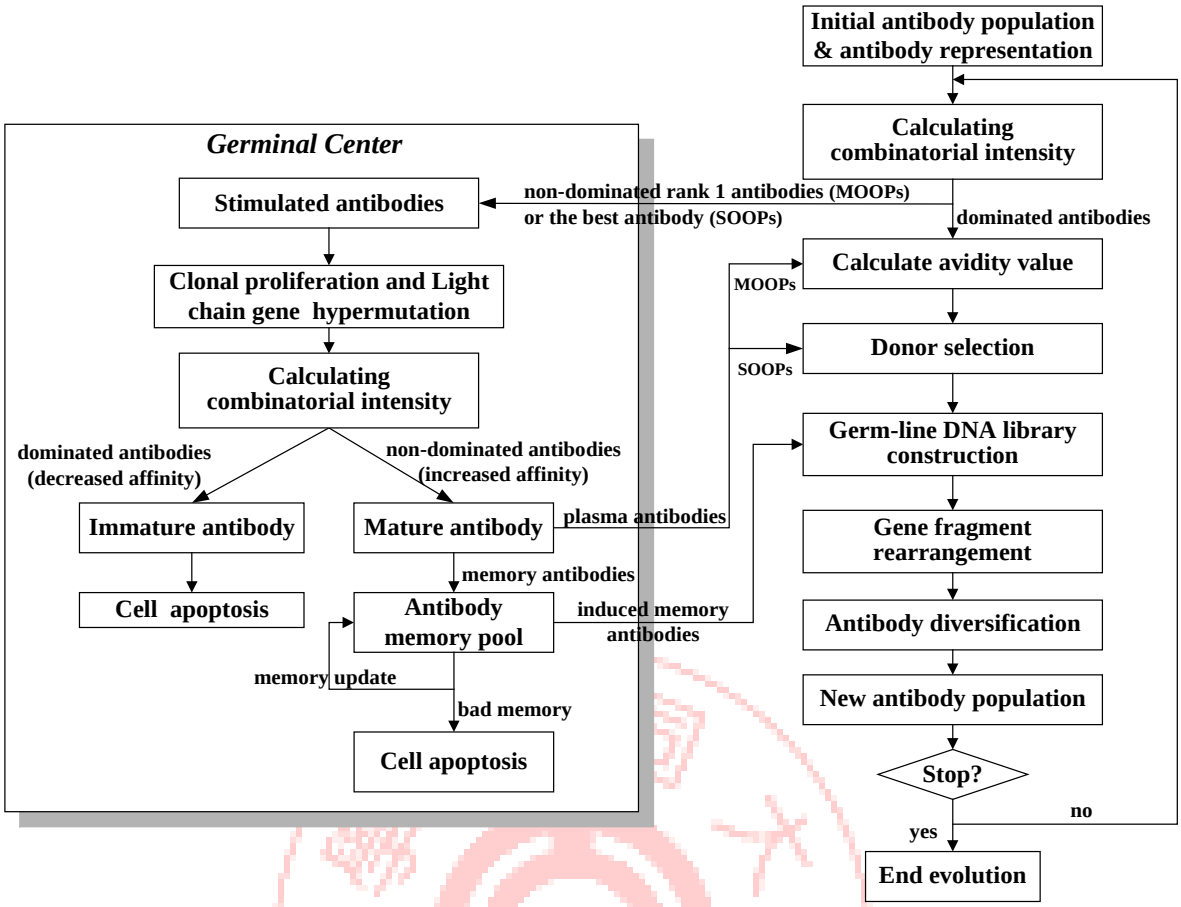


Fig. 4 Immune algorithm flowchart

4.2 Major Steps of Immune Algorithm

4.2.1 [Step 1] Establishing initial antibody population

Similar to evolutionary algorithms, the initial antibody population utilizing a pre-defined number of random string is generated randomly. For a binary-encoded antibody, each variable (x_i) in an antibody encoded by a pre-defined number of bits is separated into light-chain genes and heavy-chain genes mimicking the structure of antibody in biological immune system as depicted in the top figure of Fig. 5. The gene length or gene amount of each chain is determined by the user-defined *light/heavy*

chain-length ratio. Noted that the properties of antibody such as encoding method (*e.g.* binary-encoded or not binary-encoded, and one-dimensional binary string or two-dimensional binary string), defining of light-chain and heavy-chain gene, and/or classification of genes (*e.g.* variable gene and constant gene) should be revised for applying to various optimization problems.

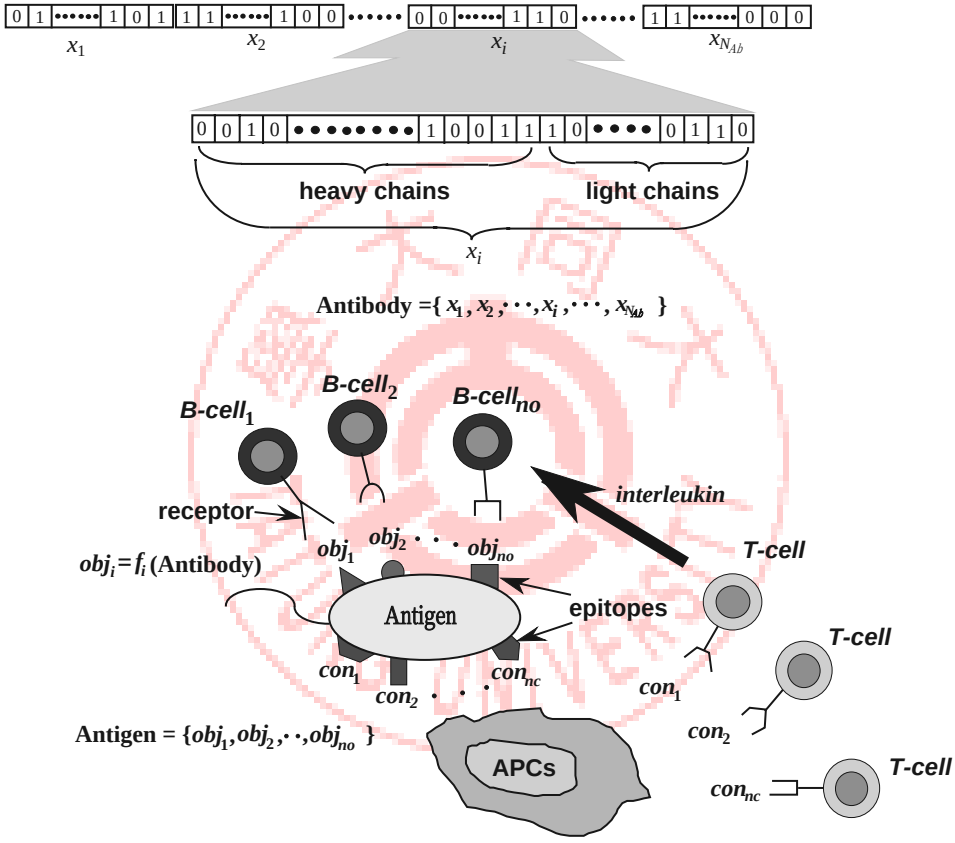


Fig. 5 Antibody-antigen representation

4.2.2 [Step 2] Calculating combinatorial intensity

In the proposed immune algorithm, the combinatorial intensity between *ith* antibody and antigens is represented by the rank values r_i for the multi-objective optimization problems (MOOPs) and by the affinity/objective values ($AbAg_{ik}$) for single-objective

optimization problems (SOOPs), expressed as:

$$\begin{aligned} r_i &= rank\left(affinity_{i1}, affinity_{i2}, \cdots, affinity_{ik}\right), \quad i = 1,2,..., N_{Ab}, \quad k = 1,2,..., N_{obj} \\ affinity_{ik} &= \overline{AbAg}_{ik} \end{aligned} \tag{4.3}$$

where $affinity_{ik}$ indicates the normalized affinity/objective values ($\overline{AbAg}_{ik}$). Note that normalization values $\overline{AbAg}_{ik}$ are utilized to prevent objective values from being numerically dominant in the optimization process. Besides, for used in the constrained optimization problems, the combinatorial intensity between *ith* antibody and antigens is replaced by the values $\bar{r}_i$, and expressed as:

$$\bar{r}_i = r_i + CK_i \tag{4.4}$$

where $\bar{r}_i$ is defined as the rank values (r_i) added by constraint violation values (CK_i). However, the cytokine value (CK) of the antibody is treated as the penalty term for constraint violation. Resembling the biological immune system, the cytokine can either stimulate or suppress the promotion of antibodies dependent on whether the antigen is non-self or self (reward feasible or penalize infeasible solutions). Computation of the cytokine is expressed as follows:

$$CK_i = \sum_{j=1}^{N_C} amount_j \times \sum_{j=1}^{N_C} count_j, \qquad i = 1,2,\cdots, N_{Ab} \tag{4.5}$$

$$\text{and} \qquad amount_j = \begin{cases} \left|\dfrac{g_j}{g_a}\right| & if \quad |g_j| > g_a \\ 0 & else \end{cases}$$

$$count_j = \begin{cases} 1 & if \quad |g_j| > g_a \\ 0 & else \end{cases}$$

where CK_i represents the cytokine value for *ith* antibody; N_C is the total number of

equality and inequality constraint conditions; $amount_j$ and $count_j$ correspond to the normalized values of the summation of *jth* antibody violated amount and total number of the *jth* antibody violated constraint condition, respectively; g_j denotes the equality and/or inequality constraint values whereas g_a indicates the allowable constraint value. Note that the larger the cytokine value the higher degree of constraint violation. Obviously, the antibodies will to be well received for evolution if the cytokine values are equal to zero. Consequently, non-dominated (*i.e.* first rank) antibodies in the MOOPs or the best antibody in the SOOPs will thus be selected into the germinal center for clonal proliferation, with the remaining dominated antibodies proceeding to Step 4 to calculate their avidity values (MOOPs only) or to Step 4 to wait for donor selection (SOOPs).

4.2.3 [Step 3] Clonal proliferation

In biological immune systems, only antibodies stimulated by antigens enter the germinal center for clonal proliferation. In the proposed immune algorithm, stimulated antibodies – non-dominated antibodies (MOOPs) or the best antibody (SOOPs) determined in [Step 2] are chosen for hypermutation during the clonal proliferation process, with a user-defined *hypermutation rate* and *proliferation number* (Fig. 4). To prevent excessive discrepancies, hypermutation only takes place with lower bits of binary code - the equivalent of light chains. After the hypermutation process, mature

antibodies (*i.e.*, non-dominated or the best proliferated antibody(s)) that have a greater combinatorial intensity than un-proliferated antibody(s) are differentiated into plasma antibodies and memory antibodies preserved and updated in the memory pool. Noted that, both plasma and memory antibody have identical gene structure i.e., the genes of plasma antibody(s) is the same with the genes of memory antibody(s). Further, the resulting bad memory antibodies and immature proliferated antibodies are neglected as the immature cell apoptosis process in biological immune systems. The surviving mature antibodies – plasma antibodies together with the dominated antibodies from antibody population derived from [Step 2] will undergo the next step to calculate their avidity values (MOOPs) or go to [Step 5] for donor antibody selection (SOOPs). Moreover part of the non-dominated antibodies in the memory pool would be re-induced to the germ-line DNA library according to the user-defined *inducing ratio*.

In this step, clonal proliferation is equivalent to the local search effect in optimization process for finding non-dominated solutions. Obviously, the larger number of proliferations is the wider space searches with trade-off of time consuming. In addition, inducing memory antibodies (global non-dominated solutions) to the germ-line DNA library will increase the exploitation effect.

4.2.4 [Step 4] Calculating avidity

In biological immune systems, affinity refers to the binding strength between a single antigenic determinants (epitope) and an individual antibody-combining site (paratope). Avidity refers to the overall strength of binding between multivalent antigens and antibodies. However, avidity is more than a simple sum of individual affinities. In this dissertation, avidity value (av_i) is the binding of affinities between antigens and antibodies as well as between antibodies only for multi-objective optimization problems. It is computed as the inverse of the combinatorial intensity (rank value r_i) between *ith* antibody and all antigens multiplied by its similarity value (S_i) with other antibodies – in other words,

$$av_i = \frac{1}{r_i \cdot S_i} \tag{4.6}$$

where, S_i representing the similarity of an *ith* antibody with other antibodies, is expressed as

$$S_i = \frac{\sum_{j=1}^{N_{Ab}} count_{ij}}{N_{Ab}}, \quad i = 1, 2, \cdots, N_{Ab} \quad , j = 1, 2, \cdots, N_{Ab}, \tag{4.7}$$

$$\text{and} \quad count_{ij} = \begin{cases} 1, & if \ AbAb_{ij} \geq \delta_{Ab} \\ 0, & else \end{cases},$$

$$AbAb_{ij} = 1/(1 + d_{ij})$$

where δ_{Ab} is a user-defined threshold value which illustrates the allowable difference between antibodies, $AbAb_{ij}$ is the affinity value between the *ith* and *jth* antibodies, and d_{ij} is the Euclidean distance between the *ith* and *jth* antibodies in objective space. Noted

that the larger the Euclidean distance d_{ij}, the larger the difference between *ith* and *jth* antibodies. Since, $0 \le AbAb_{ij} \le 1$ and when $AbAb_{ij} = 1$ (*i.e.* $d_{ij} = 0$), the *ith* antibody is identical to the *jth* antibody.

Higher avidity value means that antibody has higher activation with non-self antigen and lower similarity with the other antibodies. The higher the avidity value, the higher probability is selected to germ-line DNA library as the donor antibodies for gene fragment rearrangement. Besides, r_i corresponds to the convergence of solutions to the Pareto front and S_i corresponds to the diversity among obtained non-dominated solutions. Hence, the algorithm prefers low rank (*i.e.* high affinity) and low similarity solutions (*i.e.* diverse antibodies).

4.2.5 [Step 5] Donor antibodies selection

Similar to the building of germ-line DNA libraries in an immune system, the proposed immune algorithm uses a tournament selection method to select donor antibodies exhibiting higher avidity values (MOOPs) or affinity values (SOOPs) to assemble germ-line DNA libraries. Some antibodies (according to the predefined *tournament size*) are chosen randomly for competition and the winner is survived and subsequently turns into a donor antibody.

4.2.6 [Step 6] Germ-line DNA libraries construction

As explained in Chapter 3, the genetic material required to produce antibody molecules is stored in germ-line DNA libraries, each one containing a fragment of an antibody gene. In the proposed immune algorithm, the germ-line DNA library components include donor antibodies derived from Step 5 and part of memory antibodies induced from memory pool, at an *inducing ratio* defined by the user.

4.2.7 [Step 7] Gene fragment rearrangement

In a biological immune system, antibodies are produced through a random rearrangement of fragments selected from the germ-line DNA library. As to the proposed immune algorithm, antibodies are established using gene fragments randomly selected from corresponding light- and heavy-chain libraries of each design variable. The gene fragment rearrangement, synthesizes the antibodies by different gene fragments encoded in the germ-line DNA libraries which were composed of the fragments from the donor and memory antibodies. Note that the gene fragment rearrangement operator employed in proposed algorithm is comparable with the crossover utilized in genetic algorithms. Instead of crossing over two individuals in GA, proposed algorithm recombine building blocks (*i.e.* fragments from the fittest antibodies) directly. This suggests the superior capability of proposed algorithm in discovering accurate and diverse

non-dominated solutions rapidly. Therefore several diversification schemes are required to prevent the premature effect due to schemata recombination.

4.2.8 [Step 8] Antibody diversification mechanisms

Matching a large variety of antigens requires an equal level of diversity in antibody type. In the proposed immune algorithm, this was achieved by mimicking the following six mechanisms found in biological immune systems. All the schemes described below have the exploration effect in optimization search processes. It should be noted that the six diversification mechanisms described in this step are adopted randomly in the antibody diversity process.

1. *Somatic point mutation.* In terms of binary string representation, this means reversing a bit from 1 to 0 or vice versa according to a pre-defined *diversity probability*. The result is a slight alteration of an antibody gene for local search purposes.
2. *Somatic recombination.* As shown in Fig. 6, two light chains in the variables x_i and x_k adopted for recombination are randomly selected, after which a partial crossover between them was performed according to a randomly *diversity probability*.

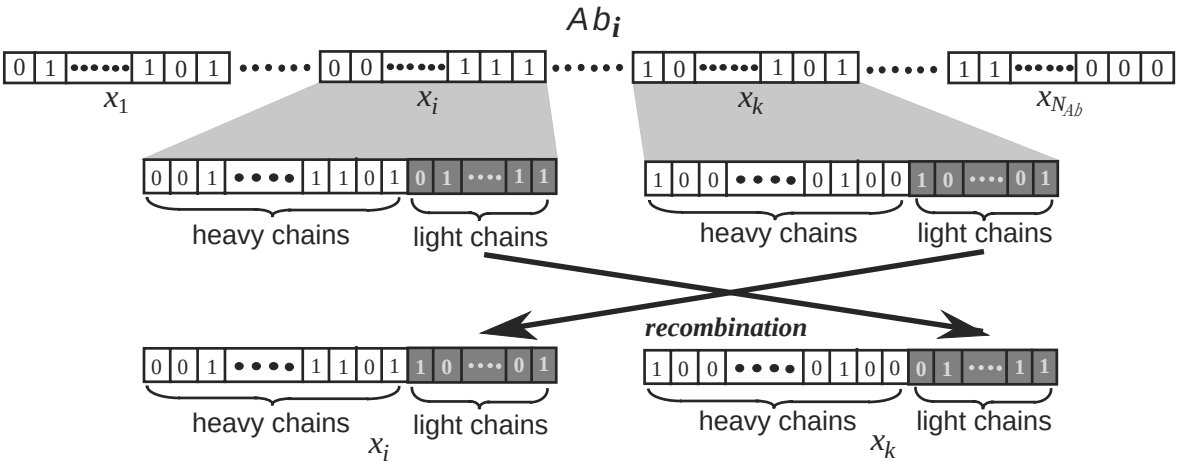


Fig. 6 Somatic recombination illustration

3. *Gene conversion*, *gene inversion*, and *gene shift*. Following predefined *diversity probability*, gene conversion, gene inversion, and gene shift were completed using a randomly picked heavy chains antibody variable (see Fig. 7 - 9). Note that the starting and ending sites were randomly generated, and the number of bit-shift genes was predefined. This type of diversification scheme results in a global search effect. In gene conversion (depicted in Fig. 7), the gene segment between the starting and ending sites of a randomly picked heavy chain was forced to converse (mutate) their genes (bits) from 1 to 0 or vice versa. As to the gene inversion operator shown in Fig. 8, randomly chosen gene segment inverses sequentially its gene positions from front to rear or from rear to front. Fig. 9 illustrates gene shift operation. Following the predefined *number of gene shift*, the selected gene segment right-shift their gene locations with excessive bits being reallocated from rear to front.

4. *Nucleotide addition.* As demonstrated in Figure 10, nucleotide insertion occurs in either light or heavy chains, depending on the variable. Several bits of genetic material (representing the nucleotide) were randomly inserted into chains that were reassembled so as to discard excess bits. The resulting nucleotide is a randomly created binary string with a pre-defined *number of nucleotide genes*. Increasing the bit number will diversify the antibody population further.

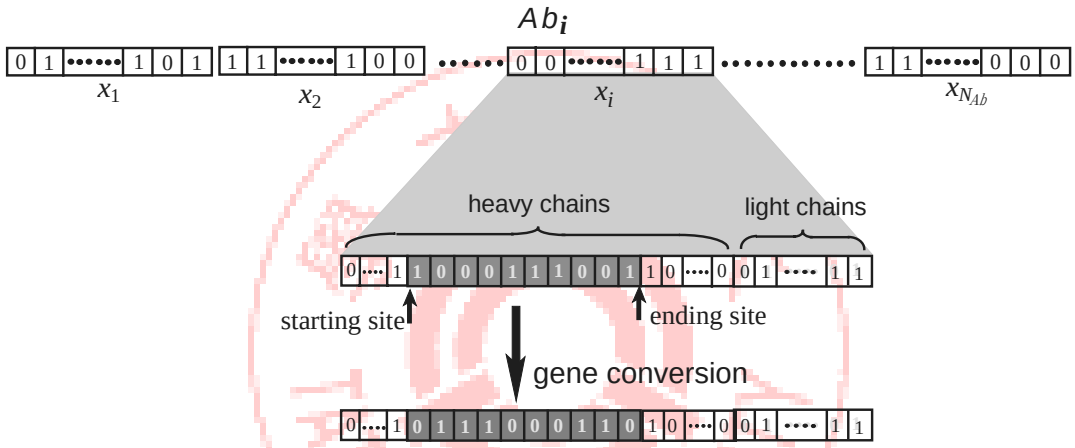


Fig. 7 Gene conversion illustration

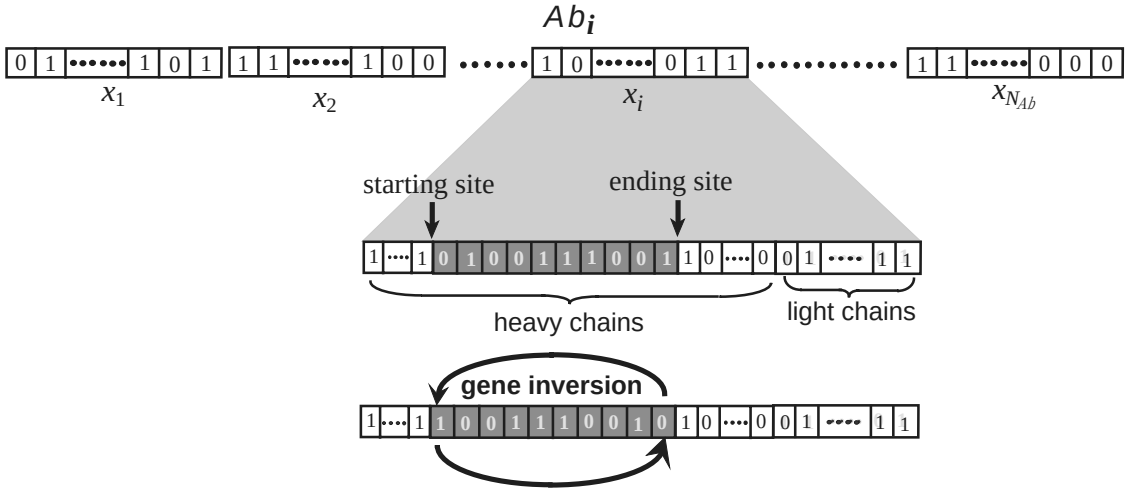


Fig. 8 Gene inversion illustration

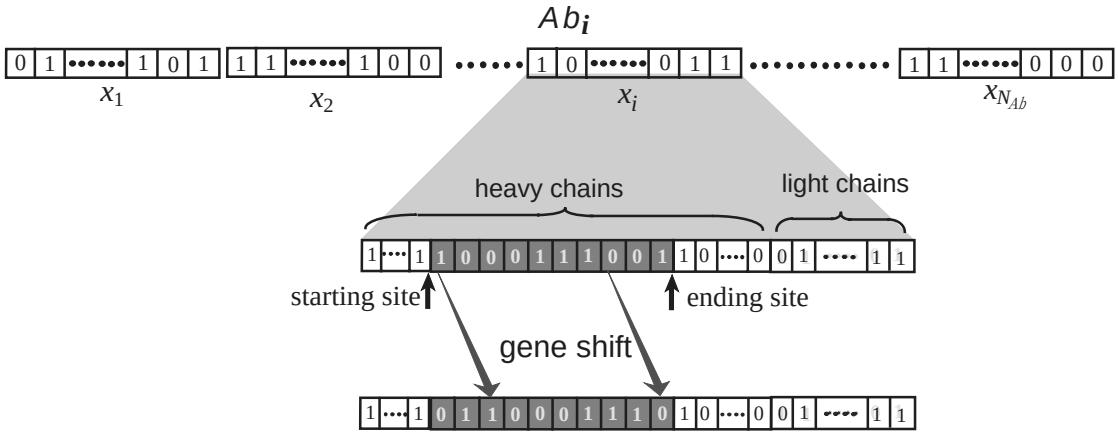


Fig. 9 Gene shift illustration

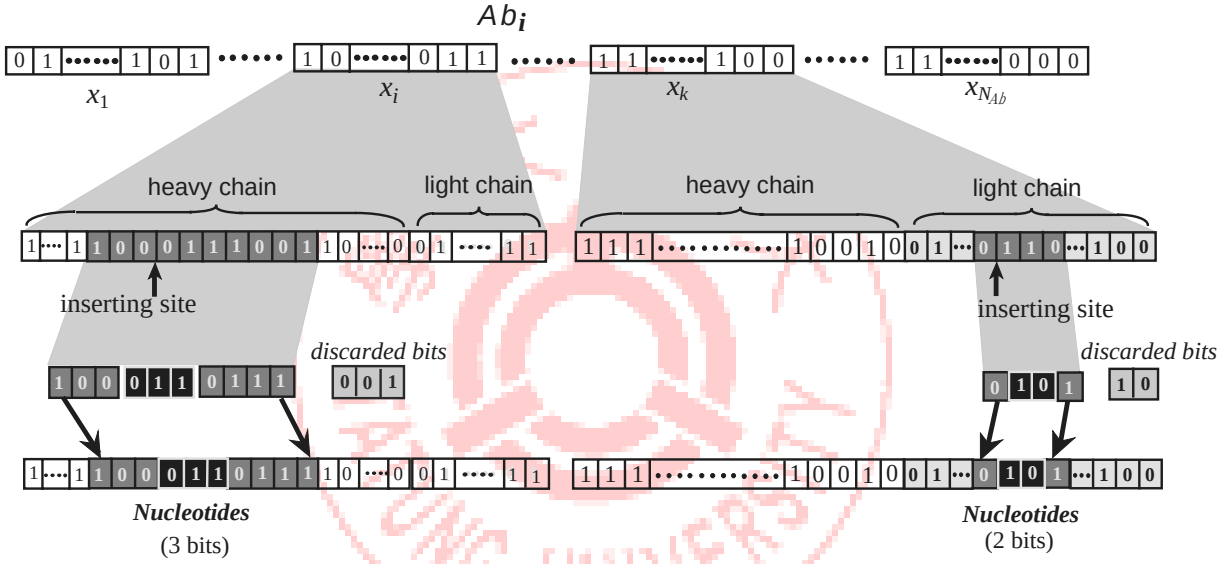


Fig. 10 Nucleotide addition illustration

4.2.9 [Step 9] Stopping criterion

The process stops when the iteration number equals a pre-defined *generation*. In the final stage, the feasible non-dominated optimal solutions are placed in the memory pool, otherwise the population returns to [Step 2] for another round.

All parameters used in the proposed immune are tabulated and described in the Table 2.

Table 2 Description of immune algorithm parameters

Parameter name	Description
Population size	The size of antibody population.
Light/heavy chain-length ratio	The proportion of light-chain genes to heavy-chain genes. Take the design variable encoded by 10-binary bit for example, if the ratio is 4/6, it means that there are 4 light-chain genes and 6 heavy-chain genes in a design variable. Note that the classification of light- and heavy-chain gene is depended on the applied problems.
Proliferation number	The number of simulated antibody(s) (derived for Step 2) proliferation. The proliferated antibody(s) accompanies hypermutation. The number of proliferation can also be regarded as the frequency of local search. More number of proliferations, more time needed for computation.
Hypermutation rate	The hypermutation occurred with antibody proliferations and it is checked with the probability of hypermutation — the hypermutation rate. During the process of antibody proliferation, a random number is generated for every light-chain gene in that antibody. If this random number is less than the hypermutation rate, the selected gene has to undergo hypermutation. Noted that this rate usually large than normal mutation rate, and more number of light-chain genes need more time for proliferations.
Inducing ratio	This parameter defines the proportion of how many memory antibodies move to the germ-line DNA library. Noted that more induced memory antibodies may speed up the convergence of optimal search.
Diversity probability	Similar to the mutation rate in the genetic algorithms this parameter is used in the antibody diversification (Step 8) for genetic diversity. A random number is generated for light- or heavy- chain genes in the antibody. If this random number is less than the diversity probability, the selected gene(s) has to undergo mutation.

Number of gene shift	This parameter is used in the gene shift mentioned in the Step 8. According to this number, the selected gene fragment right-shift their gene locus with excessive genes being reallocated from rear to front.
Number of nucleotide genes	This parameter is used in the nucleotide addition mentioned in the Step 8. These randomly created genes represented the nucleotide are inserted in the position generated randomly. Also the excessive genes will be discarded.
Generation	The criterion for stopping the evolution of immune algorithm

4.3 Summary

The procedures mentioned in this chapter describe solving the single-objective and multi-objective optimization problems by employing proposed immune algorithm. The proposed immune algorithm will first apply to the multi-objective test function optimization and truss-structure sizing optimization problems considering both constrained and unconstrained conditions expressed in the **CHAPTER 5**. In this multi-objective immune algorithm (or MOIA), the antibody is represented by one-dimensional binary-encoded string. For different optimization problems, the single-objective with multi-modal structural topology optimization is the secondly application and described in **CHAPTER 6**. In this multi-modal immune algorithm (or MMIA), the antibody is represented by two-dimensional binary-encoded string (or matrix). The third application is to use proposed immune algorithm to single-objective job-shop scheduling optimization problem and shown in the **CHAPTER 7**. For

job-shop scheduling optimization, the antibody is represented by not-bit string (integer encoding). The detail procedures for different optimizations are described in corresponding chapters.

CHAPTER 5

MULTI-OBJECTIVE OPTIMIZATION

5.1 Introduction

In this chapter, the proposed Immune algorithm will firstly apply to the numerical test function multi-objective optimization considering both unconstrained and constrained problems and two well-known benchmark of truss sizing optimization problems (*i.e.*, 10-bar plane truss with continuous design variables and 25-bar space truss with discrete design variables) considering constraints. The proposed algorithm which handling unconstrained optimizations termed multi-objective immune algorithm or **MOIA**, while termed constrained MOIA or **C-MOIA** if it solves the constrained optimizations. Noted that antibodies in these applications are all represented by one–dimensional binary-encoded string. In the multi-objective test function optimizations, six test functions without constraint and six constrained test functions suggested by Deb *et al.* (1999; 2001) were employed to validate the proposed algorithm, each test functions have two objectives which needed to be minimized simultaneously for finding their Pareto-optimal front or Pareto-optimal solutions. For the truss-structure sizing optimization, the goal is to minimize the volume (or weight) and vertical displacement of the structure simultaneously using the cross-sectional areas of the truss members as

design variables with pre-defined allowable on maximum stresses of tension and compression. Such objectives are conflicting in nature since reducing the displacement will increase the cross-sectional area, consequently increasing the volume of the structure. Meanwhile, a comparison is drawn between our implementation of the C-MOIA and some GA-based methods for 10-bar [Fadel and Li, 2002] and 25-bar [Erbatur, F. *et al.*, 2000; Ponterosso and Fox, 1999; Wu and Chow, 1995a, 1995b; Rajeev, 1992] cases. Besides, because of the unconstrained problem, the cytokine value (CK) in calculating combinatorial intensity (described in **CHAPTER 4**, subsection 4.2.2) will be neglected. Hence, the antibody-to-antigen combinatorial intensity between constrained and unconstrained optimizations are expressed as:

$$\begin{aligned} affinity_{ik} &= \overline{AbAg}_{ik} + CK_i, && \text{if constrained optimiation} \\ affinity_{ik} &= \overline{AbAg}_{ik}, && \text{if unconstrained optimiation} \end{aligned} \tag{5.1}$$

5.2 Problems Description

5.2.1 Unconstrained test functions

Six minimization test functions with different shape of Pareto-optimal front (*e.g.* convex, non-convex, discrete, and so on) described by Deb (1999) were used to evaluate the performance of the MOIA, and shown in the following:

1. Test function $F_1(\boldsymbol{x}) = (f_1(\boldsymbol{x}), f_2(\boldsymbol{x}))$ (with convex Pareto-optimal front)

$$f_1(x_1) = x_1$$

$$f_2(x) = g(x_1,..., x_n) \cdot h(f_1(x_1), g(x_2,..., x_n))$$

and $$g(x_2,.., x_n) = 1 + \frac{9}{n-1} \cdot \sum_{i=2}^{n} x_i$$

(5.2)

$$h(f_1, g) = 1 - \sqrt{f_1 / g}$$

where $\boldsymbol{x} = (x_1,..., x_n)$, $n = 30$ and $x_i \in [0,1]$. The Pareto-optimal front is formed with $g = 1$.

2. Test function $F_2(\boldsymbol{x}) = (f_1(\boldsymbol{x}), f_2(\boldsymbol{x}))$ (with non-convex Pareto-optimal front)

$$f_1(x_1) = x_1$$

$$f_2(x) = g(x_1,..., x_n) \cdot h(f_1(x_1), g(x_2,..., x_n))$$

and $$g(x_2,.., x_n) = 1 + \frac{9}{n-1} \cdot \sum_{i=2}^{n} x_i \qquad (5.3)$$

$$h(f_1, g) = 1 - (f_1 / g)^2$$

where $\boldsymbol{x} = (x_1,..., x_n)$, $n = 30$ and $x_i \in [0,1]$. The Pareto-optimal front is formed with $g = 1$.

3. Test function $F_3(\boldsymbol{x}) = (f_1(\boldsymbol{x}), f_2(\boldsymbol{x}))$ (with several non-continuous convex parts)

$$f_1(x_1) = x_1$$

$$f_2(x) = g(x_1,..., x_n) \cdot h(f_1(x_1), g(x_2,..., x_n))$$

and $$g(x_2,.., x_n) = 1 + \frac{9}{n-1} \cdot \sum_{i=2}^{n} x_i \qquad (5.4)$$

$$h(f_1, g) = 1 - \sqrt{f_1 / g} - (f_1 / g)\sin(10\pi f_1)$$

where $n = 30$ and $x_i \in [0,1]$. The Pareto-optimal front is formed with $g = 1$.

4. Test function $F_4(\boldsymbol{x}) = (f_1(\boldsymbol{x}), f_2(\boldsymbol{x}))$ (multimodality)

$$f_1(x_1) = x_1$$

$$f_2(x) = g(x_1,..., x_n) \cdot h(f_1(x_1), g(x_2,..., x_n))$$

and $$g(x_2,.., x_n) = 1 + 10(n-1) + \sum_{i=2}^{n} (x_i^2 - 10\cos(4\pi x_i)) \qquad (5.5)$$

$$h(f_1, g) = 1 - \sqrt{f_1 / g}$$

where $n = 10$, $x_1 \in [0,1]$ and $x_2,..., x_n \in [-5,5]$. The global Pareto-optimal front is formed with $g = 1$ and the next-best local Pareto-optimal front with $g = 1.25$.

5. Test function $F_5(\boldsymbol{x}) = (f_1(\boldsymbol{x}), f_2(\boldsymbol{x}))$ (binary string code and deceptive problem)

$$f_1(x_1) = 1 + u(x_1)$$

$$f_2(x) = g(x_1,..., x_n) \cdot h(f_1(x_1), g(x_2,..., x_n))$$

and $$g(x_2,.., x_n) = \sum_{i=2}^{n} v(u(x_i)) \qquad (5.6)$$

$$h(f_1, g) = 1 / f_1$$

$$v(u(x_i)) = \begin{cases} 2 + u(x_i) & if \quad u(x_i) < 5 \\ 1 & if \quad u(x_i) = 5 \end{cases}$$

where $n = 11$, $x_1 \in \{0,1\}^{30}$, $x_2,..., x_n \in \{0,1\}^5$, and $u(x_i)$ gives the number of 1s in the bit vector. The global Pareto-optimal front is formed with $g = 10$, and the next-best

deceptive Pareto-optimal front is represented by the solutions for which $g = 11$. Both global and local Pareto-optimal fronts are convex.

6. Test function $F_6(\boldsymbol{x}) = (f_1(\boldsymbol{x}), f_2(\boldsymbol{x}))$ (non-uniformity)

$$f_1(x_1) = 1 - \exp(-4x_1)\sin^6(6\pi x_1)$$

$$f_2(x) = g(x_1,..., x_n) \cdot h(f_1(x_1), g(x_2,..., x_n))$$

and $$g(x_2,.., x_n) = 1 + 9((\sum_{i=2}^{n} x_i)/(n-1))^{0.25} \qquad (5.7)$$

$$h(f_1, g) = 1 - (f_1 / g)^2$$

where $n = 10$ and $x_i \in [0,1]$. The Pareto-optimal front is formed with $g = 1$.

5.2.2 Constrained test functions

Six constrained test functions CTP2-CTP7 suggested by Deb *et al.* (2001) were employed in this study to assess the performance of the constrained multi-objective immune algorithm, or named C-MOIA. In addition, these test functions were designed to cause two different kinds of tunable difficulties in a constrained multi-objective optimization algorithm: *i*) the difficulty in the vicinity of the Pareto-optimal front (CTP2-CTP5) and *ii*) the difficulty in the entire search space (CTP6-CTP7). The test functions are shown in the following:

$$\begin{cases} \text{Minimize} \quad f_1(\mathbf{x}) = x_1 \\ \text{Minimize} \quad f_2(\mathbf{x}) = g(\mathbf{x})\left(1 - \dfrac{f_1(\mathbf{x})}{g(\mathbf{x})}\right) \\ \text{subject to} \quad c(\mathbf{x}) \equiv \cos(\theta)\left(f_2(\mathbf{x}) - e\right) - \sin(\theta) f_1(\mathbf{x}) \geq \\ \qquad\qquad a \cdot \left|\sin(b\pi(\sin(\theta)(f_2(\mathbf{x}) - e) + \cos(\theta) f_1(\mathbf{x})^c)\right|^d \end{cases} \tag{5.8}$$

The decision variable x_1 is restricted in [0,1] and the bounds of the other variables depend on the chosen $g(\mathbf{x})$ function. The constraint function $c(\mathbf{x})$ has six adjustable parameters θ, a, b, c, d, and e. In all of the above problems, additional difficulty can be introduced by using a nonlinear and difficult function $g(\mathbf{x})$ which causes difficulty in progressing towards the Pareto-optimal front. Identical parameters and $g(\mathbf{x})$ function applied by Deb *et al*. as Table 3 illustrated were employed in this paper for comparison.

Table 3 Parameters and function g(**x**) utilized in constrained test functions

Test function	Parameters						g(**x**)
	θ	a	b	c	d	e	
CTP2	-0.2π	0.2	10	1	6	1	$1+x_2$
CTP3	-0.2π	0.1	10	1	0.5	1	$1+x_2$
CTP4	-0.2π	0.75	10	1	0.5	1	$1+x_2$
CTP5	-0.2π	0.75	10	2	0.5	1	$1+x_2$
CTP6	0.1π	40	0.5	1	2	-2	$11+x_2^2-10\cos(2\pi x_2)$
CTP7	-0.05π	40	5	1	6	0	$1+x_2$

The periodic nature of the constraint boundary makes the Pareto-optimal have a number of discontinuous regions. Increasing the parameter increases the number of

disconnected regions and thus the difficulty in finding feasible solutions. Parameter a has an effect of making the transition from continuous to discontinuous feasible region far away from the Pareto-optimal region. In addition, the discrete solutions can be scattered non-uniformly by using $c \neq 1$. The parameter θ controls the slope of the Pareto-optimal regions, whereas the parameter e shifts the constraints up or down in the objective space. Moreover, small value of d may reduce each disconnected regions exist only one Pareto-optimal solution.

5.2.3 10-bar plane truss with continuous design variables

A 10-bar plane truss with the node and element numbering illustrated in Fig. 11 is adopted to evaluate the performance of the proposed C-MOIA approach. The objective is to minimize the volume of the structure and the vertical displacement at node 6 simultaneously using the cross-sectional areas of the ten truss numbers as design variables with pre-defined allowable on maximum (extension) and minimum (compression) stresses. Such objectives are conflicting in nature since reducing the displacement will increase the cross-sectional area, consequently increasing the volume of the structure. The upper and lower boundaries of each truss element are 0.1 and 30 in^2, respectively. The location of external load is shown in Fig. 11 with P = 100,000 lb. Material properties are taken as modulus of elasticity $E = 1\times10^4$ ksi. Constraints on the truss limit

the principal stress σ_j in each element below the maximum allowable stress, σ_a, of ± 25 ksi. In this dissertation, normalized constraint function is expressed as following:

$$g_j \equiv \frac{|\sigma_j|}{\sigma_a} - 1 \le 0 \qquad j = 1,...,10 \tag{5.9}$$

Note that the cross-sectional areas are assumed to be continuous numbers in this case.

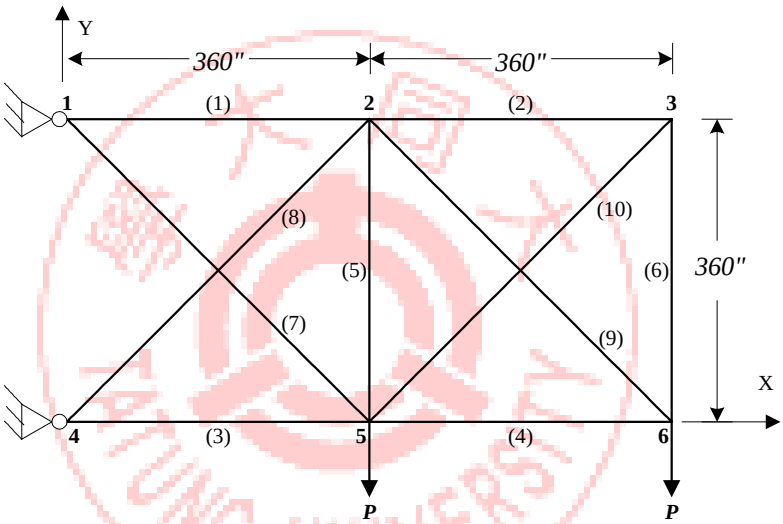


Fig. 11 10-bar plane truss structure

5.2.4 25-bar space truss with discrete design variables

The secondary truss-structure optimization considered is a 25-bar space truss with discrete design variables, which has been frequently used to test numerous optimization techniques [Michalewicz *et al.*, 1996; Hasanc *et al.*, 2001; Deb *et al.*, 2000] as Fig. 12 shown. Again, the problem is to find the cross-sectional area of each truss group such that the total structural weight and the vertical displacement at node 1 are minimized

concurrently. In many design cases, structures are composed of prefabricated elements available on the market. Thus, truss members are divided into eight groups, as tabulated in Table 4, and be selected from the following discrete set $\boldsymbol{D}$ = (0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0, 1.1, 1.2, 1.3, 1.4, 1.5, 1.6, 1.7, 1.8, 1.9, 2.0, 2.1, 2.2, 2.3, 2.4, 2.5, 2.6, 2.8, 3.0, 3.2, 3.4) (in^2). In addition, the loading given in Table 5 is applied to the space truss structure. Material properties are taken as modulus of elasticity $E = 1\times10^4$ ksi and weight density ρ = 0.1 lb/in^3. Constraints on the truss limit principal stress σ_j in each element below the maximum allowable stress, σ_a, of ±40 ksi.

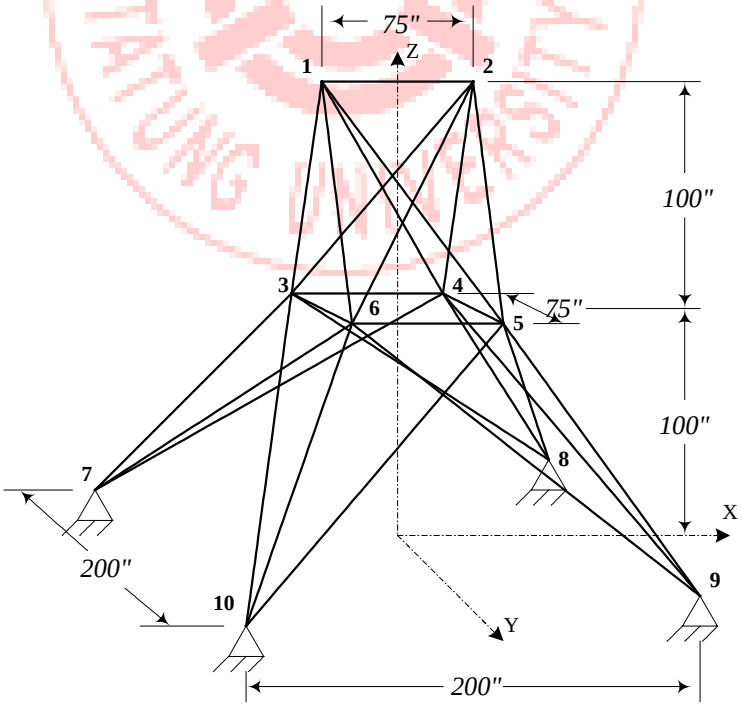


Fig. 12 25-bar space truss structure

Table 4 Group members of the 25-bar space truss

Group number	Members	Length
1	1-2	75.0
2	1-4, 2-3, 1-5, 2-6	130.504
3	2-5, 2-4, 1-3, 1-6	106.80
4	3-6, 4-5	75.0
5	3-4, 5-6	75.0
6	3-10, 6-7, 4-9, 5-8	181.142
7	3-8, 4-7, 6-9, 5-10	181.142
8	3-7, 4-8, 5-9, 6-10	133.464

Table 5 Loading conditions of the 25-bar space truss

Node	Fx (lbs)	Fy (lbs)	Fz (lbs)
1	1000	-10000	-10000
2	0	-10000	-10000
3	500	0	0
6	600	0	0

5.3 Performance Metrics

The two primary goals of multi-objective optimization are to *i*) find solutions as close to Pareto-optimal front/solutions as possible, and *ii*) discover solutions in the obtained non-dominated front that are as diverse as possible [Deb, 2001]. For the purpose of comparing with other approaches, performance criteria that have been suggested to evaluate the effectiveness of multi-objective optimization algorithms include the following five metrics: generational distance [Schott, 1995], spacing [Deb *et al.*,

2000], spread [Zitzler, 1999], set convergence [Zitzler *et al.*, 2000], and the retrieved extreme values of the Pareto front.

1. The *generational distance* (GD) metric calculates the average distance between obtained solutions and the true Pareto-optimal front. It is expressed as

$$GD = \sqrt{\sum_{i=1}^{n} d_i^2} \Big/ n$$
$$d_i = \min_{j=1}^{N} \sqrt{\sum_{m=1}^{M} (f_m^{(i)} - f_m^{(j)})^2} \qquad (5.10)$$

where n is the number of non-dominated solutions, N is the number of Pareto-optimal solutions, m is the number of objective functions, d_i is the Euclidean distance (in terms of objective space) between solution $i \in n$ and the closest Pareto solution, and $f_m^{(j)}$ is the *mth* objective function value of the *jth* member of Pareto solutions. A smaller *GD* value indicates better algorithm performance.

2. Schott [61] has proposed using a *spacing* (*S*) metric which calculates a relative distance between consecutive solutions in the obtained non-dominated set. It is expressed as

$$S = \sqrt{\frac{1}{n} \sum_{i=1}^{n} (d_i - \bar{d})^2}\ , \qquad (5.11)$$

where the distance measure d_i is the minimum value of the sum of the absolute difference in objective function values between the *ith* solution $f_m^{(i)}$ and any other

solution $f_m^{(j)}$ in the derived non-dominated set.

$$d_i = \min_{j \in n \wedge j \neq i} \sum_{m=1}^{M} \left| f_m^{(i)} - f_m^{(j)} \right| \tag{5.12}$$

where $\bar{d}$ is the mean value of the above distance measure $\bar{d} = \sum_{i=1}^{n} d_i / n$. This metric measures the standard deviation of different d_i values. A small value indicates uniform spacing between solutions.

3. The *spread* (SP) metric calculates a relative distance measure between neighboring as well as between extreme solutions in a non-dominated set. It is expressed as

$$SP = \frac{\sum_{m=1}^{M} d_m^e + \sum_{i=1}^{n} \left| d_i - \bar{d} \right|}{\sum_{m=1}^{M} d_m^e + (n-1)\bar{d}} \tag{5.13}$$

where d_i is the Euclidean distance between neighboring solutions, $\bar{d}$ is the mean value of the d_i measures, and d_m^e is the distance between the extreme Pareto front solutions and n as it corresponds to the *mth* objective function. An ideally uniform distribution produces a metric value of zero; the smaller the SP value, the more desirable the distributions.

4. The *set convergence* (***C***) metric calculates the proportion of solutions obtained through algorithm ***B*** as they are weakly dominated by solutions obtained through algorithm ***A***, that is,

$$\boldsymbol{C}(\boldsymbol{A},\boldsymbol{B}) = \frac{\left|\{b \in \boldsymbol{B} \mid \exists a \in \boldsymbol{A} : a \le b\}\right|}{\left|\boldsymbol{B}\right|} \tag{5.14}$$

All algorithm ***B***-derived solutions are weakly subordinate to algorithm ***A***-derived solutions if $\boldsymbol{C}(\boldsymbol{A},\boldsymbol{B}) = 1$. $\boldsymbol{C}(\boldsymbol{A},\boldsymbol{B}) = 0$ indicates an absence of such subordination. Both $\boldsymbol{C}(\boldsymbol{A},\boldsymbol{B})$ and $\boldsymbol{C}(\boldsymbol{B},\boldsymbol{A})$ are required for a performance comparison.

5. The *extreme distance* (ΔEXT) metric calculates the Euclidean distance between the extremes of the derived non-dominated solutions and the actual Pareto solutions in the objective space. It is expressed as follows,

$$\Delta \boldsymbol{EXT} = \left[\Delta ext_1, \Delta ext_2, \cdots, \Delta ext_{Nobj}\right]$$

$$\text{with} \quad \Delta ext_k = \sqrt{\sum_{i=1}^{N_{Obj}} (p_k^{(i)} - f_k^{(i)})^2}\,, \quad k = 1,2,\cdots,N_{obj} \tag{5.15}$$

where $p_k^{(i)}$ is the *ith* extreme value of the derived *kth* non-dominated solutions employing MOIA and $f_k^{(i)}$ is the associated *ith* extreme value of the *kth* actual Pareto solutions, respectively. $\Delta \boldsymbol{EXT} = \boldsymbol{0}$ means that the MOIA's non-dominated extremes are identical to the true objective extremes. A smaller $\Delta \boldsymbol{EXT}$ value indicates better algorithm performance.

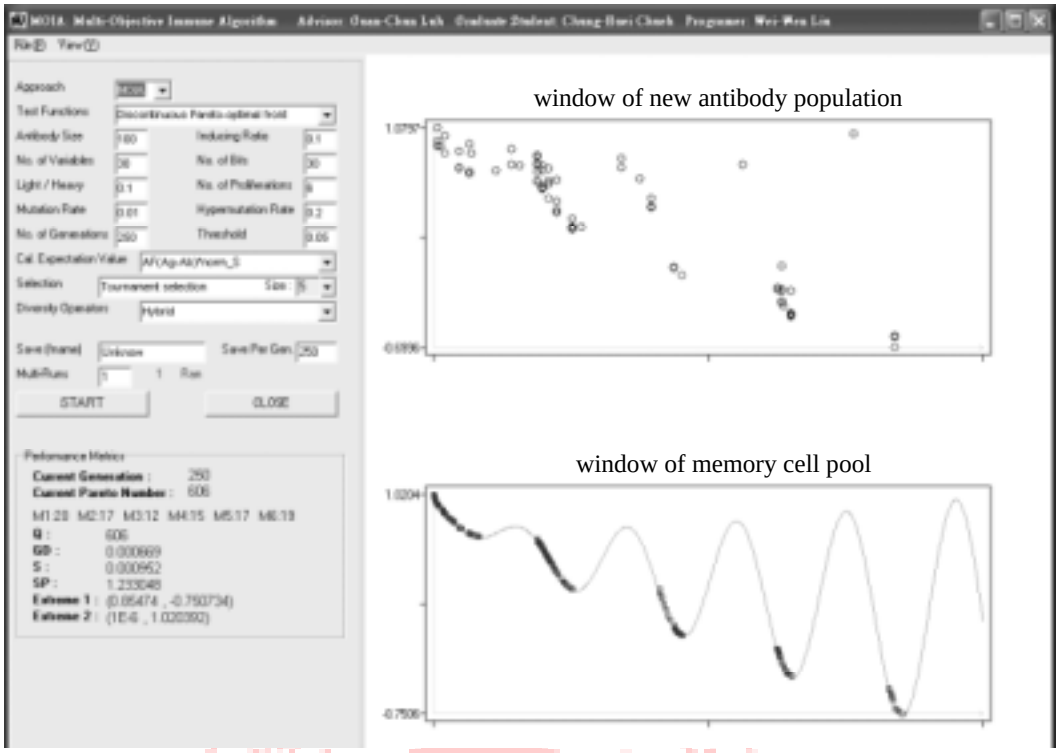


Fig. 13 MOIA window simulation

Table 6 SPEA and MOIA parameters

Parameter setting	SPEA	MOIA
Generation	250	250
Population size	100	80
Bits per variable	30	30
External population size	20	
Crossover rate	0.8	
Mutation rate	0.01	0.05
Selection method	Tournament	Tournament
Niche	Clustering algorithm	Affinity between antibodies
Elitist strategy	Yes	No
Hypermutation rate		0.2
Light/heavy chain-length ratio		3/7
Proliferation number		6
Inducing ratio		0.1
Threshold value δ_{Ab}		0.9
Bit number in gene shift		2-bit
Bit number of nucleotide		2-bit

5.4 Simulation Results and Discussions

This attempt at establishing a multi-objective optimization procedure produced the MOIA and C-MOIA programs created with C++ programming tools and a graphical user interface. The simulation window, setting parameters, and performance metrics are all shown in Fig. 13.

5.4.1 Multi-objective test function optimization

Unconstrained test functions

To evaluate the performance of the MOIA, author used several of Zitzler's (website) data sets with different optimization schemes (*i.e.*, random search algorithms, MOGA, NPGA, VEGA, NSGA, and SPEA) for comparison. Since the SPEA-derived [Zitzler *et al.*, 2000] results showed superior performance in terms of accuracy and diversity, author will limit his discussion to those solutions. Following the procedure described by Zitzler, the six unconstrained test functions were executed 30 times each; the SPEA and MOIA parameters are shown in Table 6. For each test function, the 30 data sets were unified prior to eliminating the dominated solutions. For equitable comparison, we reduce the population size in Table 6 to 80 due to the clonal proliferation in the proposed immune algorithm. According to the plots shown in Figure 14 through 19, the MOIA-derived solutions were superior to the SPEA-derived solutions in terms of accuracy and diversity,

with the single exception of test function F_6. Performance metrics for both schemes are presented as Table 7. The data in this table validates the quality of the MOIA-based performance metrics *Q*, *GD*, *S*, Δ***EXT*** and ***C***; again, the only exceptions were metrics associated with test function F_6. Furthermore, the result ***C***(MOIA,SPEA) = 0 for test functions F_1 through F_5 shows that all MOIA-derived solutions were non-dominated compared to the SPEA-derived solutions. In contrast, the result ***C***(SPEA, MOIA) = 1 for test functions F_1, F_3, F_4, and F_5 shows that all of the SPEA-derived solutions were weakly dominated by the MOIA-derived solutions. Both schemes were capable of reaching a local Pareto front, but incapable of discovering a global Pareto front (Fig. 19). Results from test function F_6 indicate that the SPEA-derived solutions were more accurate than those derived with MOIA (Fig. 19). Due to the small number of solutions, test function F_6 was repeated with a twice iteration of 500 generations (twice the number used in the initial test) in order to more fully explore the MOIA's potential. The results of this second run indicate a significant improvement in performance (Fig. 20). The performance metrics shown in Table 7 lend further support to these results.

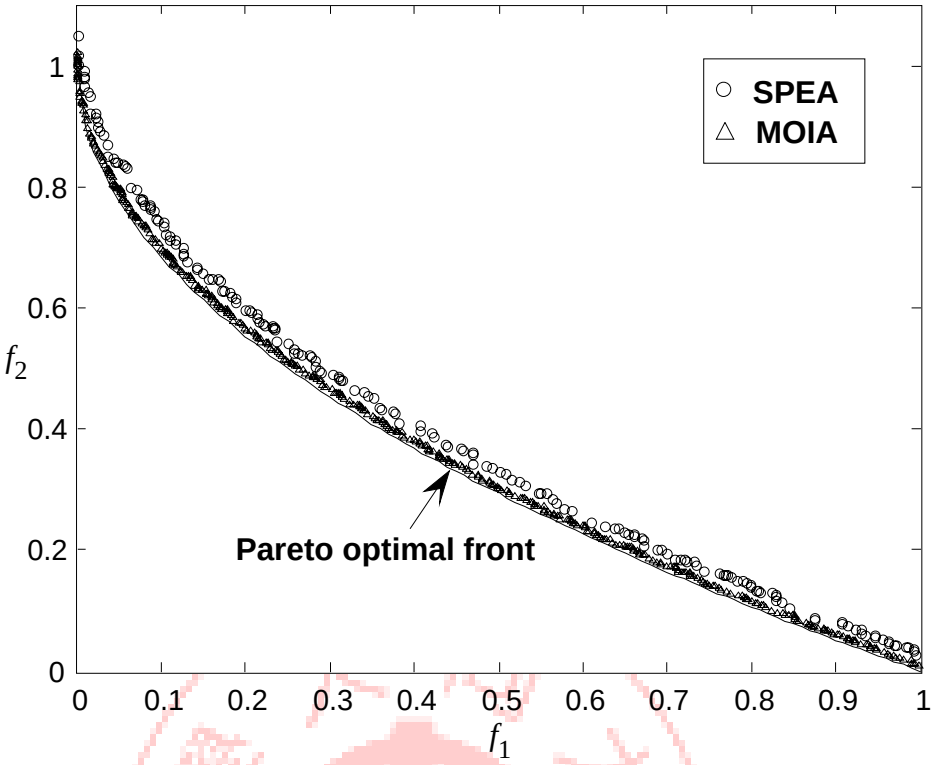


Fig 14 Simulation results for test function F_1 (convex)

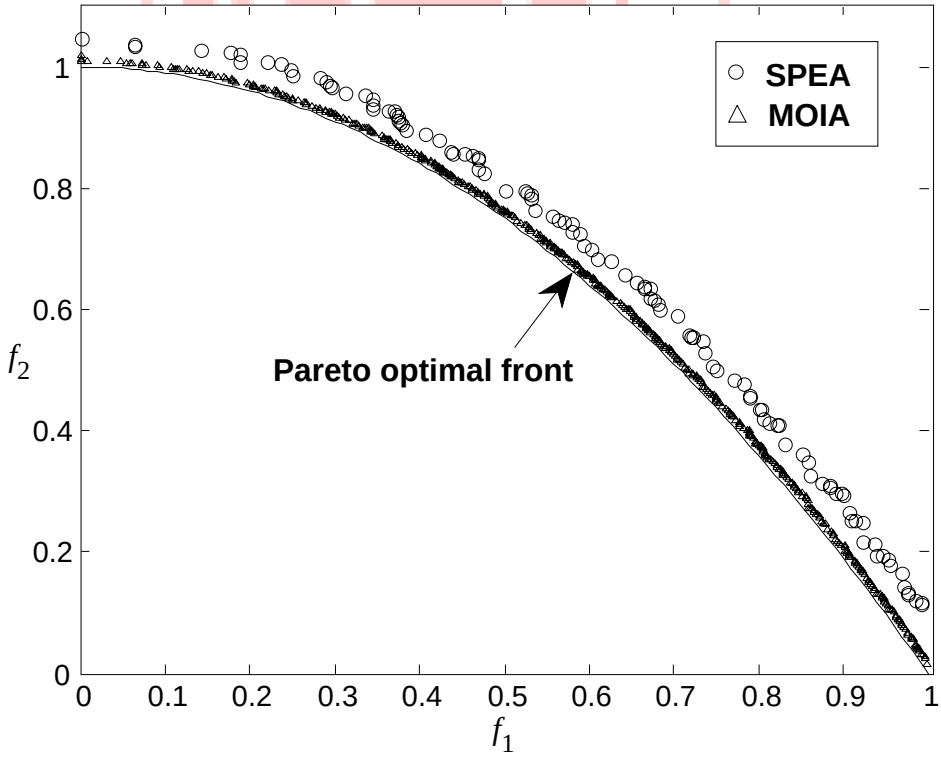


Fig 15 Simulation results for test function F_2 (non-convex)

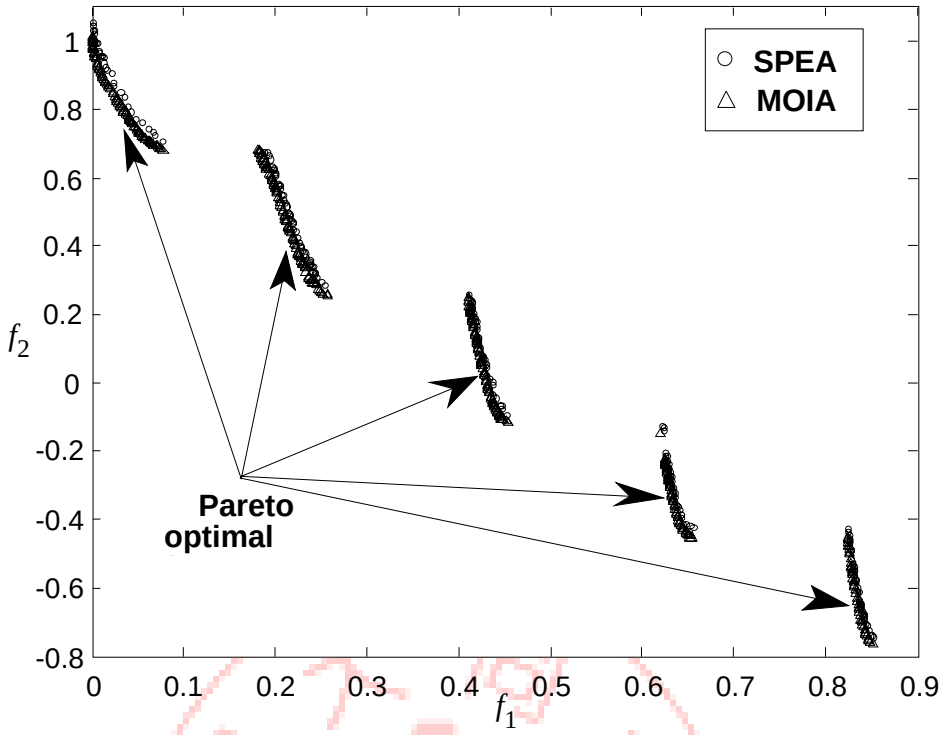


Fig 16 Simulation results for test function F_3 (discrete)

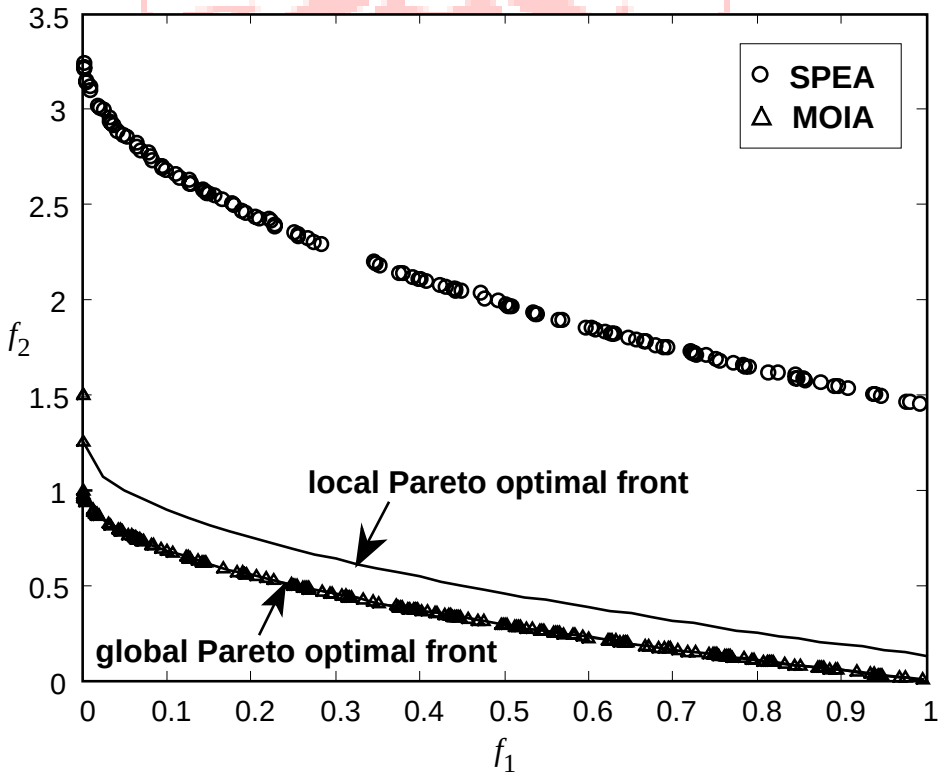


Fig 17 Simulation results for test function F_4 (multimodal)

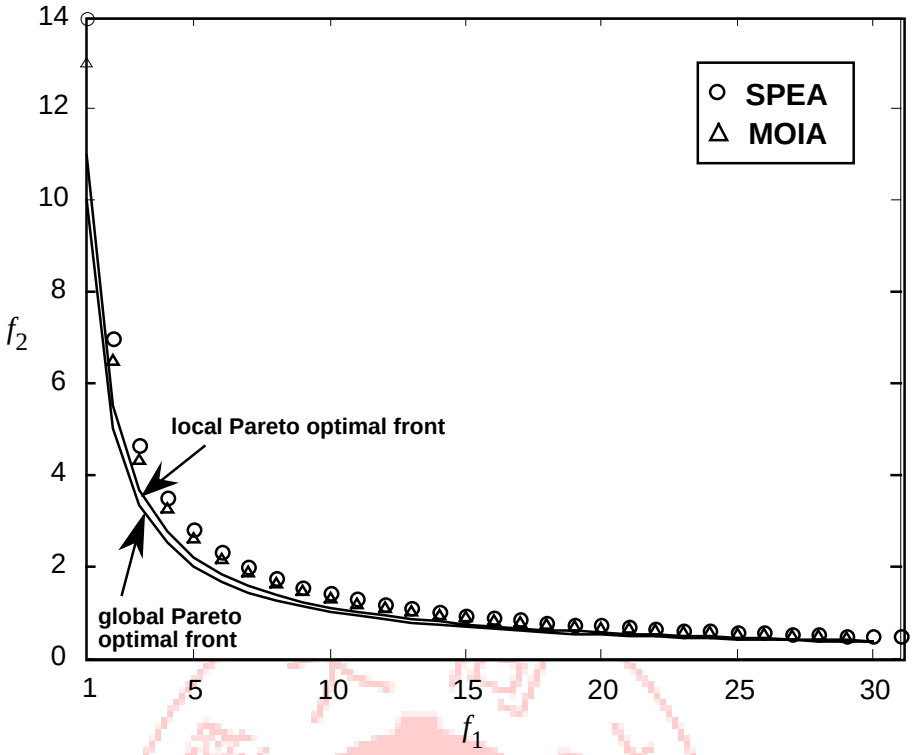


Fig 18 Simulation results for test function F_5 (deceptive)

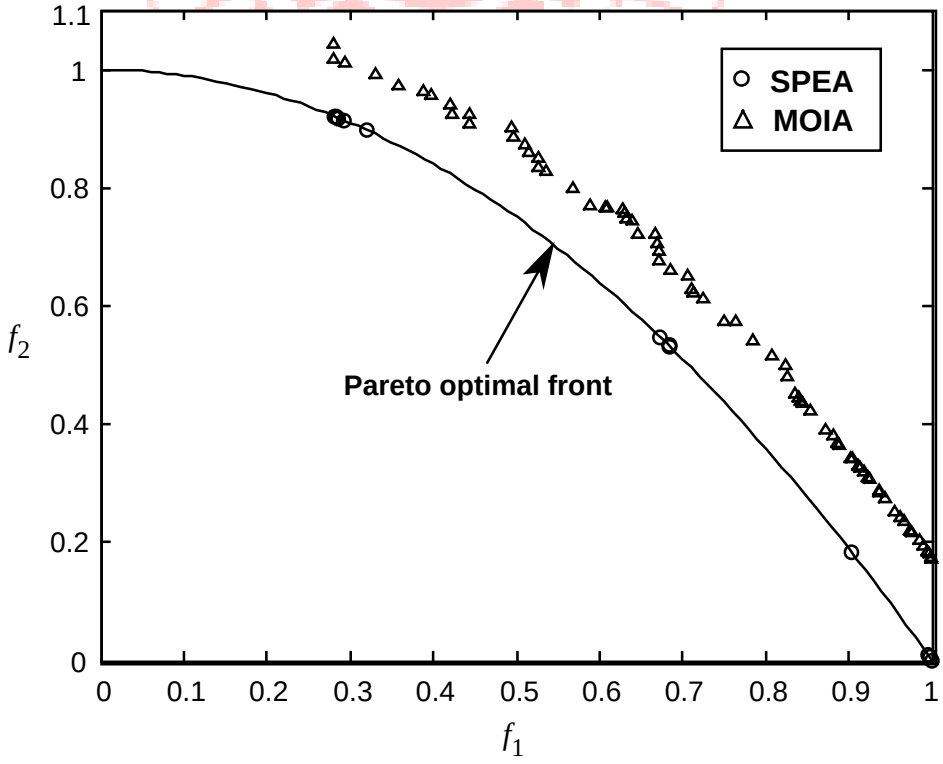


Fig 19 Simulation results for test function F_6 with 250 generations (non-uniform)

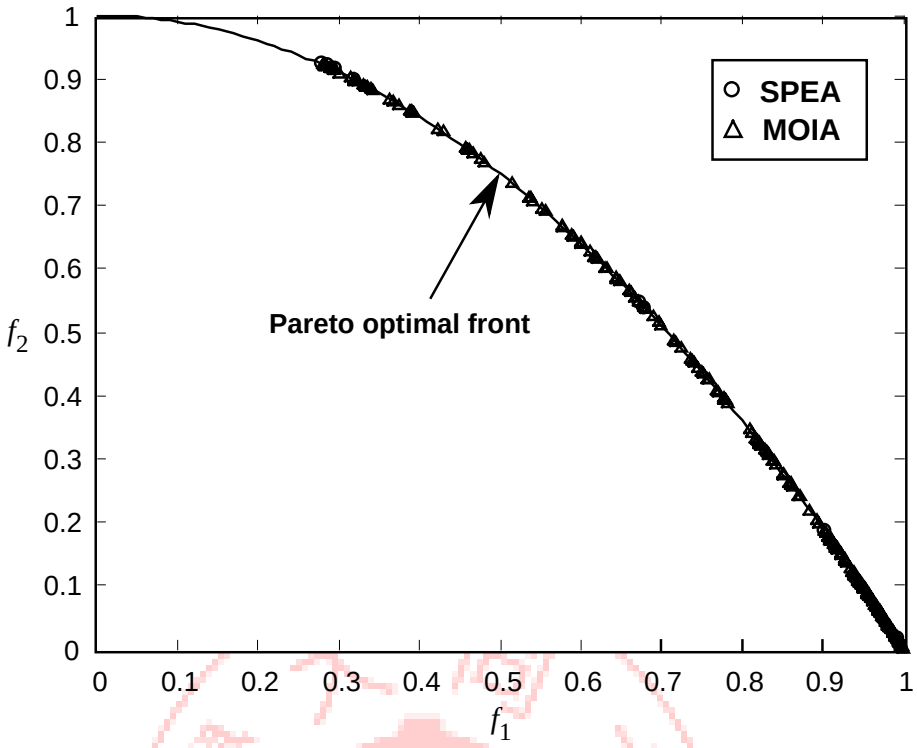


Fig 20 Simulation results for test function F_6 with 500 generations (non-uniform)

Table 7 Performance metrics for the six SPEA and MOIA test functions

	Metrics	***Q***	***GD***	***S***	***SP***	Δext_1	Δext_2	***C(MOIA,SPEA)***	***C(SPEA,MOIA)***
F_1	MOIA	537	5.38e-4	0.0018	0.5085	0.0198	0.0074	0	
	SPEA	204	3.10e-3	0.0043	0.6046	0.0509	0.0286		1
F_2	MOIA	574	5.95e-4	0.0016	0.5625	0.0190	0.0157	0	
	SPEA	112	6.80e-3	0.0099	0.6305	0.0458	0.1139		0.991
F_3	MOIA	477	6.69e-4	0.0040	0.7785	0.0141	0.4285	0	
	SPEA	202	3.60e-3	0.0048	0.7863	0.0594	0.4276		1
F_4	MOIA	293	0.0105	0.0051	1.1099	0.5051	0.0032	0	
	SPEA	156	1.4320	0.0079	0.9011	2.2520	1.4566		1
F_5	MOIA	29	0.1313	1.2253	0.4654	3.0	2.0039	0	
	SPEA	31	0.1639	1.2811	0.4813	4.0	0.1290		1
F_6	MOIA (250 generations)	92	0.0156	0.0107	0.7659	0.1248	1.1727	0.630	
	SPEA	22	2.55e-21	0.0104	1.5822	0.0019	0.0		0
	MOIA (500 generations)	142	6.18e-4	0.0056	1.3253	0.0017	0.0	0	0.4545

Table 8 C-MOIA parameters used in constrained optimization

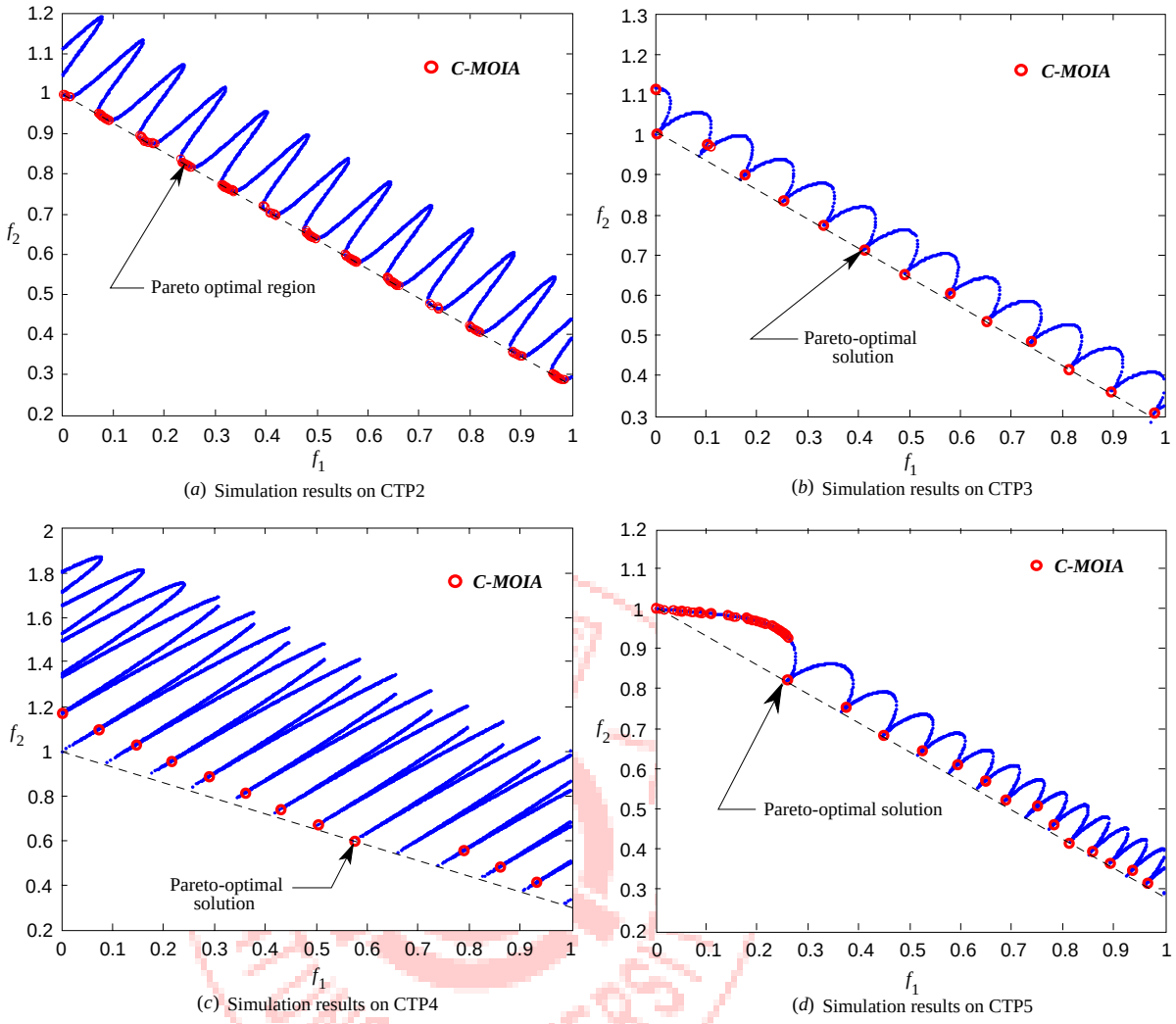
Parameter setting	CTP2-CTP7
Iteration number	500
Population size	100
Number of variables	2
Bits per variable	10
Diversity probability	0.05
Hypermutation rate	0.07
Light/heavy chain-length ratio	3/7
Number of proliferation	6
Inducing ratio	0.2
Threshold value δ_{Ab}	0.9
Bit number in Gene shift	2-bit
Bit number of nucleotide	2-bit
Tournament size	5

Constrained test functions

The associated user-defined parameters utilized in this constrained MOIA (C-MOIA) are tabulated in Table 8, each with the same parameter setting. The setting of the first four parameters (*e.g.* iteration number, population size, number of variables, and bits per variables) was referenced to the Deb *et al.* (2001) for comparison. Figs. 21 and 22 show the simulation results on test functions CTP2-CTP7, Fig. 23 and 24 show the results of CTP2-CTP7 derived by NSGA-II reprinted from Deb *et al.* (2001). Fig. 21(a) and Fig. 21(b) show that C-MOIA is able to find all disconnected Pareto-optimal solutions on CTP2 and capable of finding solution very close to the true Pareto-optimal solution in each region on CTP3. Same as the results derived by Deb *et al*., problem CTP4 caused

difficulty for C-MOIA to get near the true Pareto-optimal solutions as Fig. 21(c) depicted. However, MOIA discovers more Pareto-optimal solutions and performs much better compared to the results derived by Deb *et al.* utilizing NSGA-II. As to the non-uniformity in spacing problem CTP5, it seems to be not a great difficult for C-MOIA to get the solutions as Fig. 21(d) illustrated. Nevertheless, it should be noted that NSGA-II could not converge to the Pareto-optimal solutions in CTP2-CTP5 when f_1 approaches zero. On the contrary, it seems cause no difficulty for MOIA to converge to the feasible Pareto solutions when f_1 advances to zero.

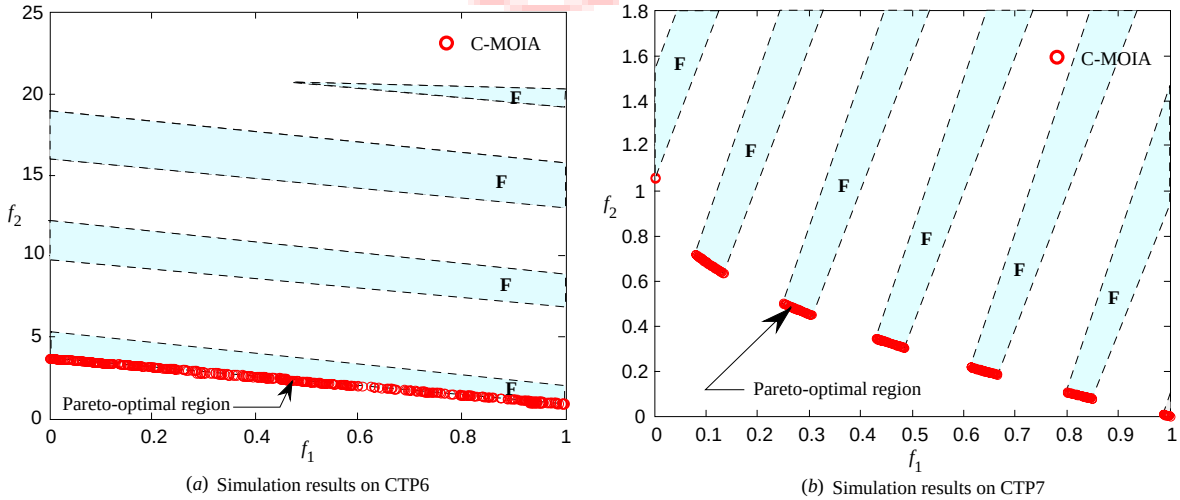
When the entire search space consists of infeasible patches parallel (CTP-6) or perpendicular (CTP-7) to the Pareto-optimal front, C-MOIA is still able to converge and close near to the feasible patches as Fig. 22 shown. Note that all feasible patches are marked with an "**F**". However, NSGA-II had the most difficulty in finding solutions closer to the true Pareto-optimal front in CTP-7 as Deb *et al*. described. The illustrated demonstrations CTP-2 to CTP-7 show that CMOIA performs better than NSGA-II in the vicinity of Pareto-optimal front as well as the entire search space.



(*a*) Simulation results on CTP2 (*b*) Simulation results on CTP3

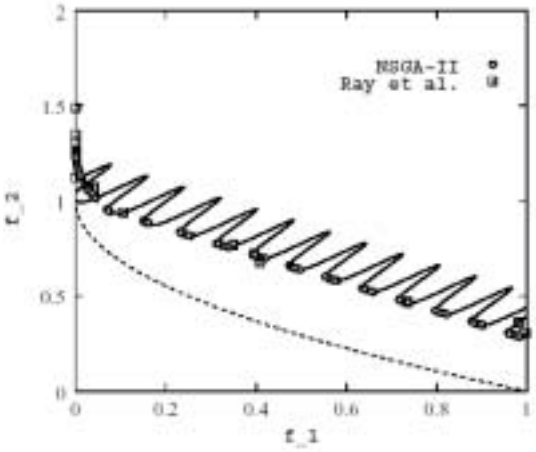
(*c*) Simulation results on CTP4 (*d*) Simulation results on CTP5

Fig. 21 Simulation results on CTP2-CTP5

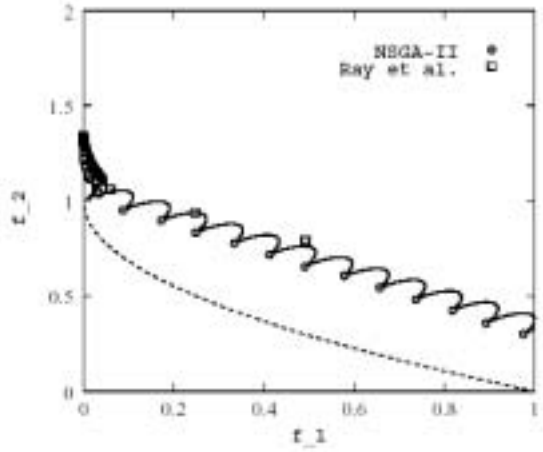


(*a*) Simulation results on CTP6 (*b*) Simulation results on CTP7

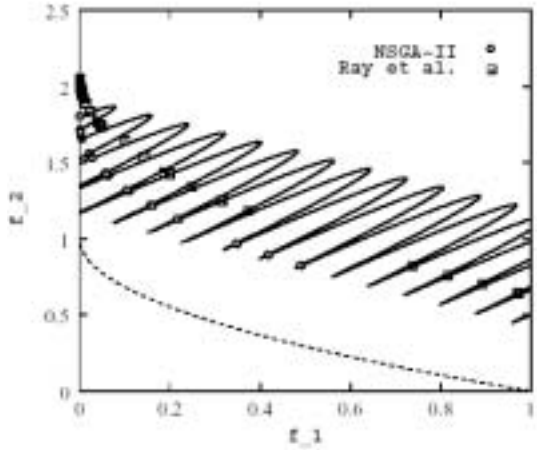
Fig. 22 Simulation results on CTP6-CTP7



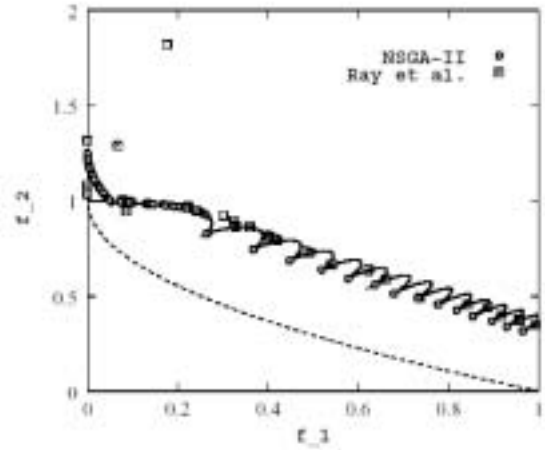
(*a*) NSGA-II results on CTP2



(*b*) NSGA-II results on CTP3

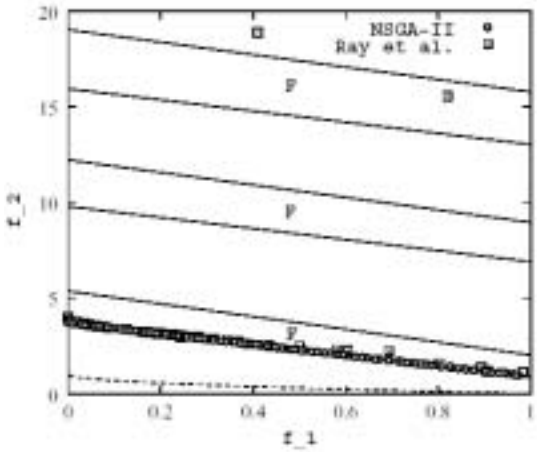


(*c*) NSGA-II results on CTP4

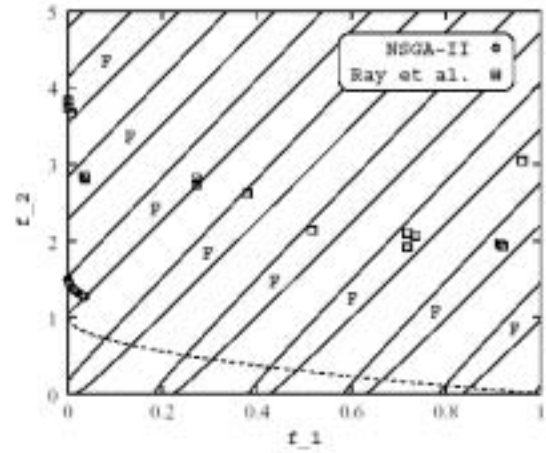


(*d*) NSGA-II results on CTP5

Fig. 23 NSGA-II results on CTP2-CTP5 [This is a reprint from Deb *et al.* (2001)]



(*a*) NSGA-II results on CTP6



(*b*) NSGA-II results on CTP7

Fig. 24 NSGA-II results on CTP6-CTP7 [This is a reprint from Deb *et al.* (2001)]

Table 9 CMOIA parameters in truss sizing optimization

Parameter setting	10-bar	25-bar
Iteration number	500	300
Population size	100	160
Number of variables	10	25
Bits per variable	10	5
Diversity probability	0.05	0.05
Hypermutation rate	0.07	0.07
Light/heavy chain-length ratio	4/6	3/2
Number of proliferation	6	5
Inducing ratio	0.2	0.2
Threshold value δ_{Ab}	0.05	0.05
Bit number in Gene shift	2-bit	2-bit
Bit number of nucleotide	2-bit	2-bit
Tournament size	5	8

5.4.2 Multi-objective truss-structure sizing optimization

The associated user-defined parameters utilized in two problems are tabulated in Table 9. The setting of the first four parameters (*e.g.* iteration number, population size, number of variables, and bits per variables) was referenced to the literatures for comparison. Note that number of bits per variable for the 25-bar truss case is 5 due to the 30 sizing variables discrete set. The results show that the light/heavy chain-length ratio is the most important parameter and 4/6 is a moderate choice. However, the ratio for 25-bar truss case is 3/2 because the bit number per variable is only 5. Obviously, the associated number of clonal proliferation is depending on the bit length of light-chain. More number of proliferation means more computation time required. Increasing

diversity probability, hypermutation rate, bit number in gene shift and nucleotide will cause diversified effect and should be determined according the optimization problem. On the contrary, the inducing ratio has converged effect. The key issue is the appropriate balance between exploitation and exploration during optimization search. These parameters were determined through numerical experiments after multiple simulation runs.

10-bar plane truss with continuous design variables

For 10-bar plane truss problem, the Pareto-optimal front with 474 feasible solutions is presented in Fig. 25. The two extreme objective values are [108413.542, 1.3611] and [17935.1162, 6.3562], respectively. Moreover, Fig. 25 illustrates the comparison with the Pareto solutions derived by Fadel and Li (2002) employing the Tchebycheff, weighting, and ε-constraint methods. It is important to emphasize that only 21 solutions were derived for these methods since they employed weighting-based method with 21 fixed and uniform-distributed weighting ratio values. The extreme values of these weighting ratios were (0.0, 1.0) and (1.0, 0.0) with interval 0.05. Obviously C-MOIA is capable of finding much more satisfactory non-dominated solutions excluding the two extreme objective values.

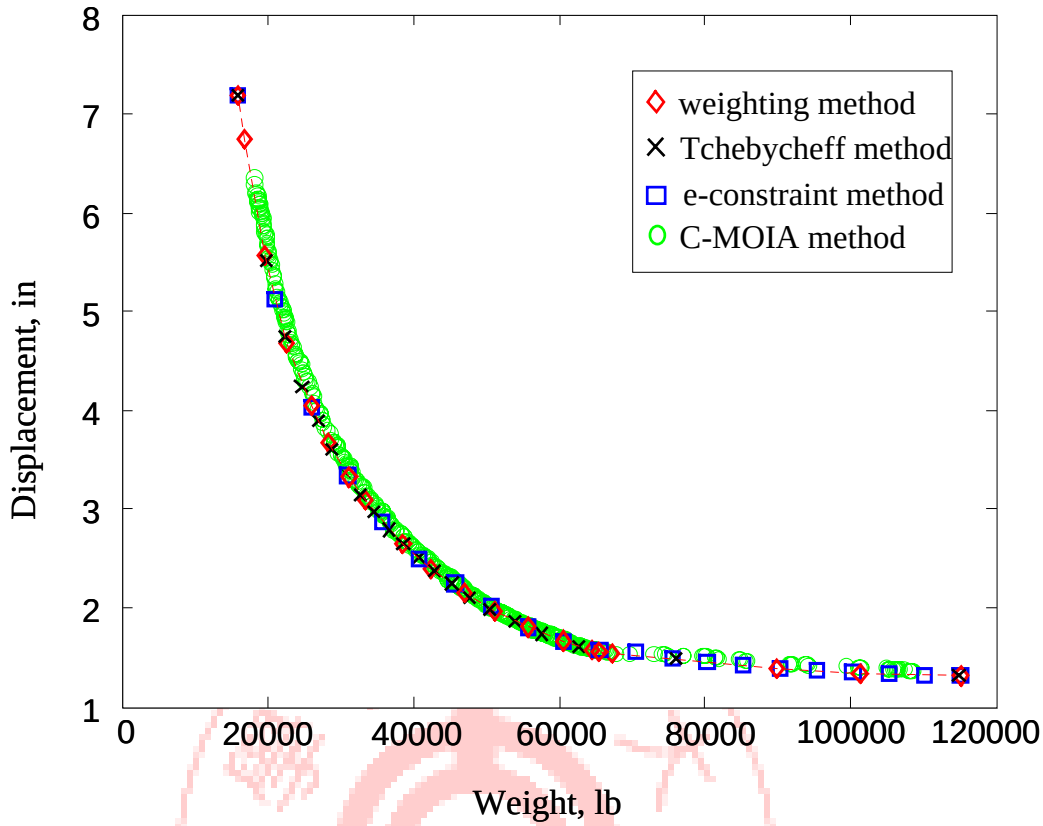


Fig. 25 feasible Pareto solutions and comparisons of 10-bar plane truss

25-bar space truss with discrete design variables

As to the 25-bar space truss problem, the two extreme objective values found are [977.39, 0.2363] and [99.87, 2.0281], respectively. Fig. 26 demonstrates the Pareto-optimal front of the 232 feasible non-dominated solutions derived employing C-MOIA. In addition, numerous simulation results utilizing single-objective GAs are adopted for comparison and depicted in Fig. 26. Clearly the results derived using C-MOIA dominate the solutions obtained from the literatures except the optimal solution attained by Wu (1995a) utilizing single objective GAs. In summary the proposed C-MOIA has

the ability to provide a good estimate of the Pareto front for the 25-bar space truss optimization problem as well.

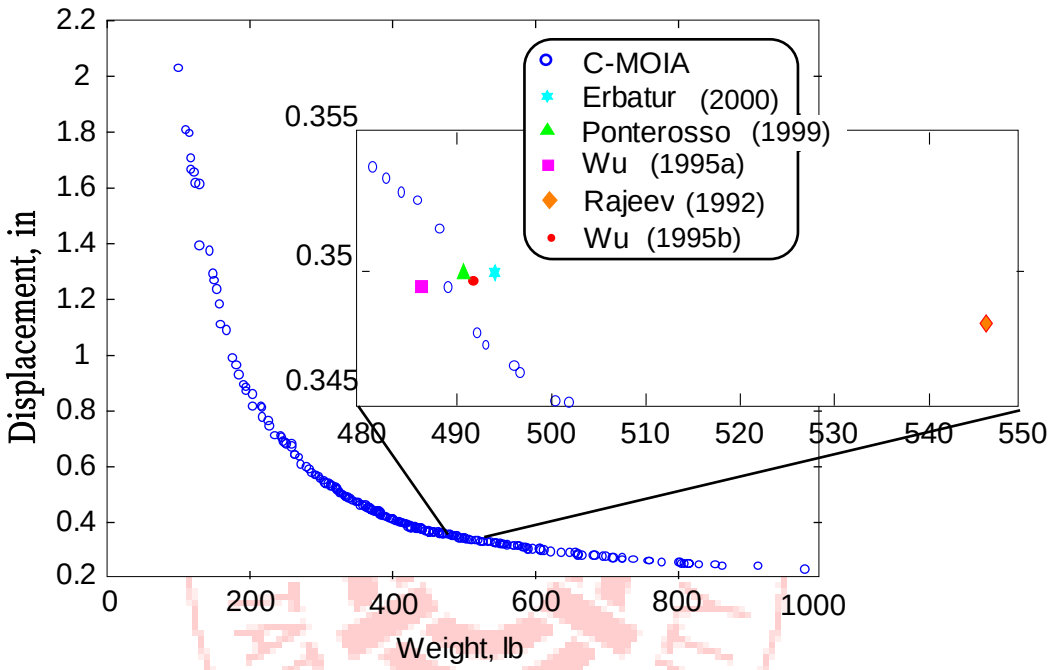


Fig. 26 Feasible Pareto solutions and comparisons of 25-bar space truss

5.5 Summary

In this chapter, the proposed immune algorithm was implemented to several test functions considering with/without constraints and two typical truss-structure sizing problems with a mix of discrete and continuous variables for the purpose of determining constrained Pareto-optimal solutions. Overall results indicate that the proposed immune algorithm is capable of quickly determining accurate and diverse Pareto-optimal solutions to multi-objective optimization problems (MOOPs). It is suggested that this capability

is due to the combination of diversification immune operators, the construction of germ-line library equivalents, and a process of gene fragment recombination, they are all the features in the immune system. The key natural selection components (gene fragments) are similar to the building blocks of genetic algorithms associated with stimulus antibodies and memory cell pools. In this particular immune algorithm, the antibodies (solutions) are the direct products of gene fragment combinations (schemata), rather than the antibody (analog to the role of individual in genetic algorithms) itself. This explains the need for several diversification schemes to prevent the premature effect of proposed immune algorithm.

CHAPTER 6

STRUCTURAL TOPOLOGY OPTIMIZATION

6.1 Introduction

In this chapter, the proposed immune algorithm will then apply to single-objective multi-modally optimization of structural topology. For applying to multi-modal optimization, the immune algorithm described in chapter 4 needs to be modified slightly. The modified immune algorithm used in the chapter is called multi-modal immune algorithm or MMIA. Analogous to the sharing and niching approaches in genetic algorithms employed by Goldberg and Richardson (1987), a similarity value (the relationship between antibodies) combined with antibody-to-antigen affinity is employed to explore the single-objective with multi-modally solutions. Two well-studied benchmark topological problems considering asymmetry are used for evaluating the effectiveness of proposed immune algorithm in the field of multi-modal optimization. The goal of this application is to maximize the structure's stiffness-to-weight ratio proposed by Chapman *et al.* (1994) subjected to maximum allowable stresses. Because of different antibody representation (two-dimensional binary-encoded matrix) from previous chapter, steps described in Chapter 4 in section 4.2 are needed to modify for applying to single-objective multi-modally optimization. The procedure of modification

is depicted in next section.

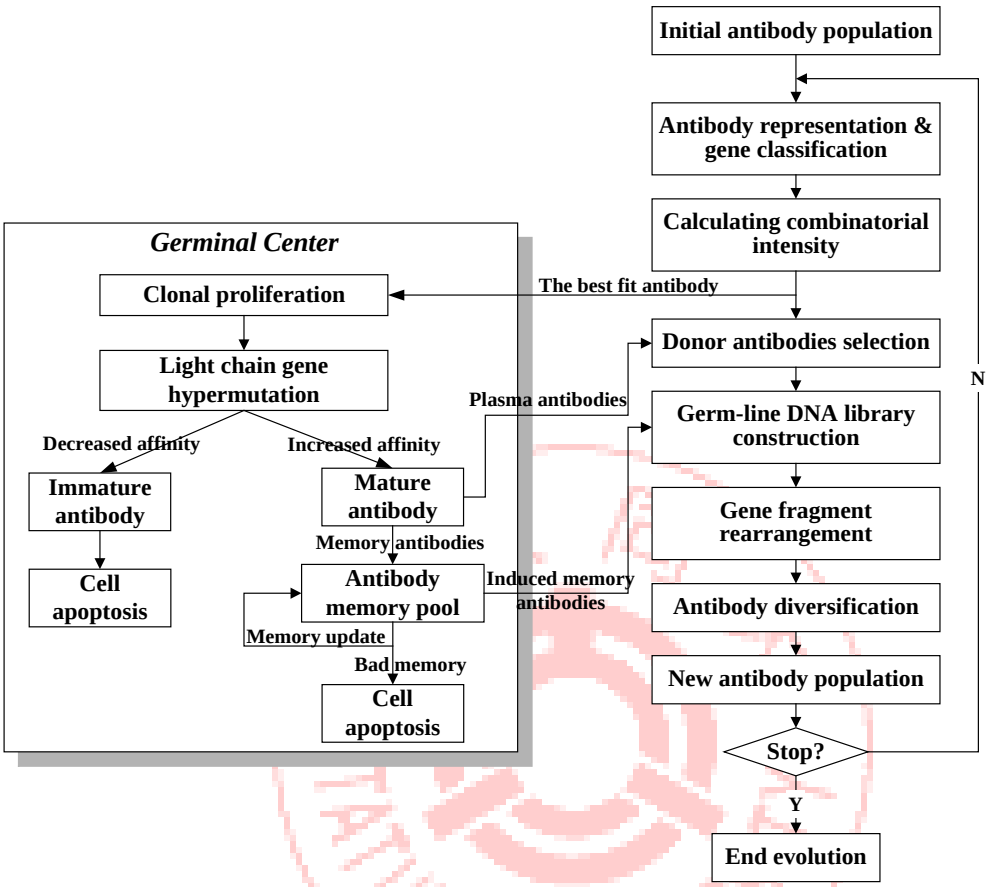


Fig. 27 Multi-modal immune algorithm (MMIA) flowchart

6.2 Immune Algorithm Revision for Topological Optimization

Corresponding to the topological optimization problems, the antigen (Ag) and antibodies (Ab_i) serve as objective (f) and associated solutions (*i.e.* topologies) in a computational model and are expressed as follows:

$$\begin{aligned} &Ab_i \equiv \mathbf{x}_i = \text{possible topologies},\ i = 1,2,...,N_{Ab} \\ &Ag \equiv f \end{aligned} \tag{6.1}$$

$$AgAb_i \equiv f(\mathbf{x}_i) \tag{6.2}$$

where Ab_i (or $\mathbf{x}_i$) indicates the *ith* antibody (or topology) while $AgAb_i$ indicates the

affinity (*i.e.* objective value, $f(\mathbf{x}_i)$) between an *ith* antibody and antigen, N_{Ab} is the number of antibodies. The antibodies/topologies continuously evolve until a match is found with the specific antigen/objective. In the rest of this section we will verbally describe the MMIA procedure represented by the flowchart in Fig. 27.

[Step 1] Random initialization of antibody population and connectivity analysis

Similar to the generation of population initially in MOIA, the initial binary-encoded antibody population is also generated randomly. Different antibody representation form MOIA (binary-encoded with one-dimensional array), the antibody used in MMIA is represented by a two-dimensional binary-encoded string (or matrix) with binary values of 1 refers to as the structure materials and 0 refer to as no material presented in the design domain. Fig. 28 shows how an antibody is mapped to a topology. Once an antibody is converted into a topology the resulting material elements with binary values of 1 are analyzed for connectivity. For any two elements in a topology to be considered as connected they must share at least one edge while element sharing only one corner are considered as disconnected. A topology contained disconnected elements will undergo a structure modification procedure. In this procedure, the removal of disconnected elements or the adding of elements to neighboring disconnected element will be done randomly until the discontinuous topology is compensated. The continuous topology

will be further analyzed via the finite element computation to obtain the required displacements and stresses. To reduce computation time, elements with a stress value lower than the user-defined level of average stress (which do not break connectivity requirements) will be removed from the topology, and the corresponding gene set to a binary value of 0.

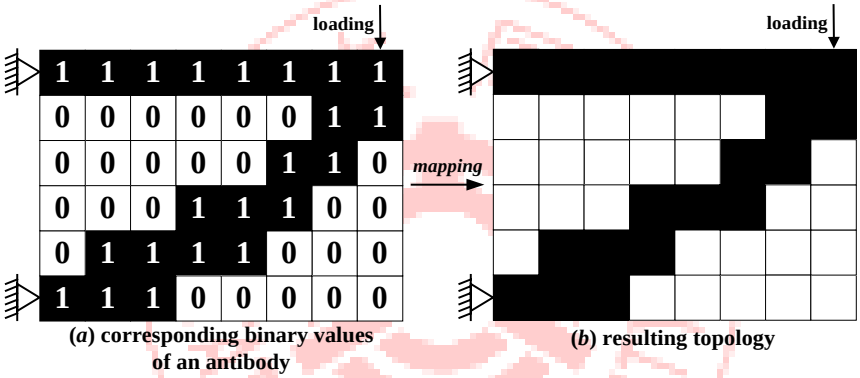


Fig. 28 Mapping from antibody into topology

[Step 2] Antibody representation and gene classification

In the same manner as in biological immune systems, each antibody/topology (as depicted in Fig. 29) is separated into a two-dimension matrix comprising four different kinds of genes/elements: a constant gene (**C**), a heavy-chain gene (**H**), a light-chain gene (**L**), and a pseudo gene (**0**). The genes are classified into heavy-chains or light-chains according to *i*) a default light/heavy chain-length ratio determined by the user and *ii*) the average stress of the mapped continuous topology calculated by fore-node plane stress

finite element analysis. A gene is categorized into a light-chain gene if its stress is either (*i*) large than the normalized average stress multiplied by light/heavy chain-length ratio or (*ii*) small than the normalized average stress multiplied by one minus the light/heavy chain-length ratio. In other words, the gene can be defined as light-chain gene if it receives exceeding large or small stress. Constant genes are those genes required to contain material where support conditions or loads are applied. These genes are fixed and cannot be changed during the whole evolution process. Pseudo genes are those genes which no contain material. The other genes will be defined as heavy-chain gene except light-chain, constant, and pseudo gene.

The distribution of these genes within an antibody thus establishes the topology. The number of genes in each antibody equals the number of elements in the topology domain. The antibody is then resolved into binary values where all constant, light-chain and heavy-chain genes are defined as 1, with all pseudo genes being defined as 0 (see Fig. 28*a*). The inbuilt ability for genes to mutate enables pseudo genes to evolve into light or heavy chain genes if they contain the material, conversely light and heavy chain genes may evolve into pseudo genes where no material is present. Due to the binary coding, the material/void design domain typically results in a discrete, non-convex search space [Anagnostou *et al.*, 1992] and serves as a good test of the capacity of MMIA in finding optimal solutions.

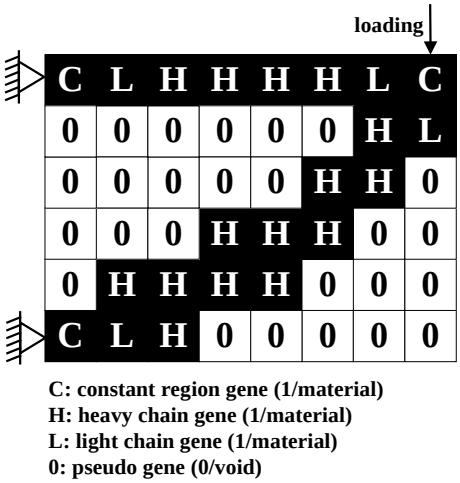


Fig. 29 Antibody representation for topological optimization

[Step 3] Calculating combinatorial intensity

The antibody-to-antigen affinity value ($AbAg_i$) of the topology is employed to illustrate the combinatorial intensity between an antigen and the *ith* antibody expressed as follows:

$$AbAg_i = \frac{Obj_i}{(1+CK_i)\cdot S_i}$$

$$\text{and} \qquad Obj_i = \frac{1}{d_i^{\max}\cdot Area_i} \tag{6.3}$$

where Obj_i indicates the *ith* topology's stiffness-to-weight ratio with the stiffness being represented by the inverse of the topology's maximum displacement ($\frac{1}{d_i^{\max}}$) at the point of loading application. The number of connected genes/elements of the topology is used as a qualitative measure of the topology's weight ($Area_i$). In addition, similar to the sharing or niching schemes implemented in the genetic algorithms, the relationship among antibodies is evaluated according to the similarity value S_i for the purpose of

multi-modally optimization, and expressed as:

$$S_i = \sum_{j=1}^{N_{Ab}} count_{ij}\ ,\quad i = 1,2,...,N_{Ab};\quad j = 1,2,...,N_{Ab}$$

$$\text{with}\quad count_{ij} = \begin{cases} 1, & if\ \ AbAb_{ij} \le \delta_{Ab} \\ 0, & else \end{cases}, \tag{6.4}$$

$$AbAb_{ij} = \sqrt{(avg_i^{stress} - avg_j^{stress})^2 - (std_i^{stress} - std_j^{stress})^2}$$

where δ_{Ab} is a user-defined threshold value illustrating the allowable difference between antibodies. $AbAb_{ij}$ is the affinity between *ith* and *jth* antibody and represents the distance between the *ith* and the *jth* antibodies in a coordinate system of average stress versus standard deviation stress, the larger the $AbAb_{ij}$, the larger the difference between *ith* and *jth* antibodies. Note that avg_i^{stress} and avg_j^{stress} are normalized averages stress values, and that std_i^{stress} and std_j^{stress} are normalized standard deviations stress values of the *ith* and the *jth* antibody/topology. In addition, cytokine value *CK* of the antibody described in Eq. (6.3) is treated as the penalty term for constraint violation and defined identical to Eq. (4.5) in the CHAPTER 4 in subsection 4.2.2.

Because of the single-objective with multi-modally problem, the avidity value in Fig. 4 in CHAPTER4 will be neglected in this procedure MMIA. A higher combinatorial intensity – affinity value means that an antibody has a higher activation with an antigen and a lower similarity with other antibodies. Therefore, the higher the affinity value, the higher the probability that the antibody may be selected as the donor to enter the germ-line DNA library for gene fragment rearrangement. After affinity values of all

antibodies are calculated, the best (*i.e.*, highest affinity) antibody will be placed into the germinal center for clonal proliferation with the remaining antibodies proceeding to [Step 5] awaiting donor selection.

[Step 4] Clonal proliferation

In the MMIA scheme, only most-matched antibody (*i.e.* highest affinity antibody) derived from [Step 3] is chosen for hypermutation during the clonal proliferation process, with a user-defined hypermutation rate and proliferation number. Similar to the MOIA scheme, hypermutation only takes place in light-chain genes. In this study, a gene is categorized into a light-chain gene if its stress is either (*i*) large than the normalized average stress multiplied by light/heavy chain-length ratio or (*ii*) small than the normalized average stress multiplied by one minus the light/heavy chain-length ratio. In the process of hypermutation, a light-chain gene is likely to be deleted if its stress is smaller than the value calculated in (*i*). On the contrary, a gene is added to the void neighborhood of the light-chain gene if its stress is larger than the value evaluated in (*ii*).

After the hypermutation process, mature antibody that have a better affinity than un-proliferated antibody is differentiated into plasma antibody and memory antibody preserved and updated in the memory pool. Further, the resulting bad memory antibodies are deleted as immature antibodies similar to the cell apoptosis process in

biological immune systems. Resulting plasma antibody combined with the remaining antibodies derived from Step 3 are then proceed to Step 5 for donor antibody selection according to their affinity value. In the memory pool, only the most diverse (determined by similarity value S_i) antibodies with high affinity survive. On the other hand, those antibodies with low affinity and high similarity will be removed from the memory pool. In this step, diversity is evaluated by checking the average stress and standard deviation stress of the elements/genes in the topology/antibody. In addition, a part of memory antibodies are induced into the germ-line DNA library (as per Step 6) according to a user-defined inducing rate.

[Step 5] Tournament selection for donor antibodies

Based on the affinity values, the tournament selection method is also employed here for donor antibody selection.

[Step 6] Germ-line DNA library construction

As described in **CHAPTER 3**, the genetic material used to produce antibody molecules is stored in germ-line DNA libraries, each one containing a fragment of an antibody gene. In the MMIA, components from the memory antibodies and the donor antibodies construct the germ-line DNA library.

[Step 7] Gene fragment rearrangement

In the MMIA, new antibodies are created via gene fragments (or blocks) rearrangement as illustrated in Fig. 30. Arbitrary gene blocks are selected randomly from randomly chosen sub-libraries and then integrated into a new antibody.

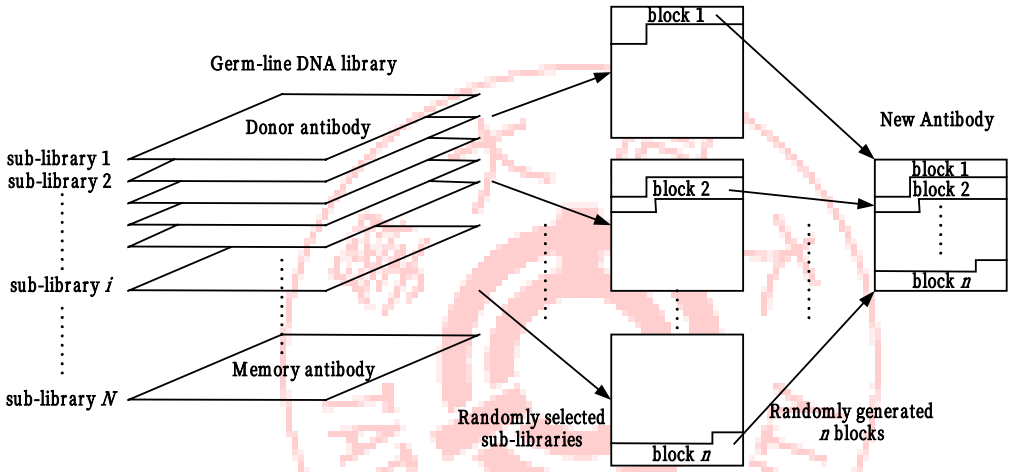


Fig. 30 Illustration of antibody rearrangement for topological optimization

[Step 8] Antibody diversification

In the proposed MMIA, this was achieved by mimicking the following six diversification mechanisms found in biological immune systems. All the schemes described below have the exploration effect in optimization search processes. Because of different antibody representation, a part of antibody diversification mechanisms depicted in MOIA (subsection 4.2.8) are needed to modify for applying in this diversification process.

1. *Somatic point mutation*. In terms of binary-encoded representation, this means reversing a bit from 1 to 0 or vice versa according to a pre-defined diversity probability. The result is a slight alteration of an antibody heavy chain gene for local search purposes.

2. *Somatic recombination*. As shown in Fig. 31, two same-size fragments/blocks are randomly selected from the same antibody, after which a partial exchange is performed between the two fragments according to a pre-defined diversity probability.

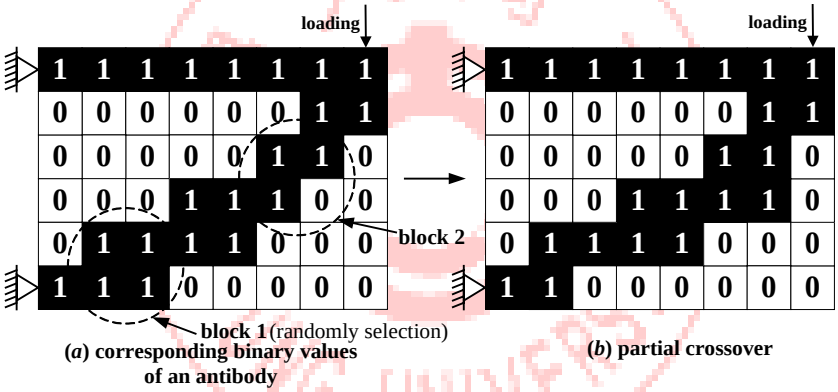


Fig. 31 Somatic recombination illustration for topological optimization

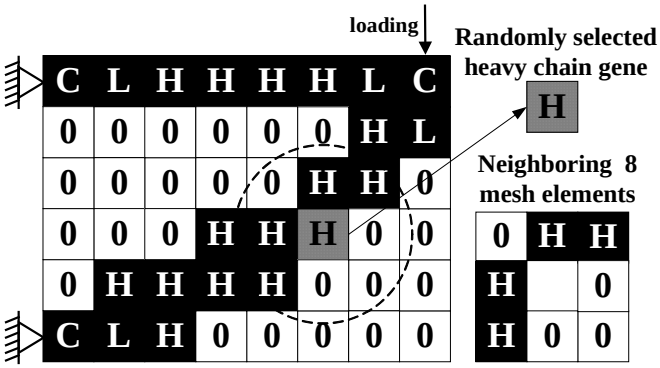


Fig. 32 Randomly selected heavy chain gene and associated neighborhoods

3. *Gene conversion*, *gene inversion*, and *gene shift*. Following predefined probabilities, gene conversion, gene inversion, and gene shift is performed using a randomly selected heavy-chain gene and its neighboring 8 mesh elements as depicted in Fig. 32. Fig. 33 illustrates the process of gene conversion where all elements within the selected 9 mesh elements have their binary values reversed from 1 to 0 and vice versa. In gene inversion as depicted in Fig. 34, each of the 8 periphery mesh elements are mirrored around the center element. Fig. 35 shows the illustration of gene shift where each of the 9 mesh elements shift a number of position from left to right and top to bottom. Note that the number of bit-shift genes is a predefined by the user. These diversification schemes create the desired global search effect.

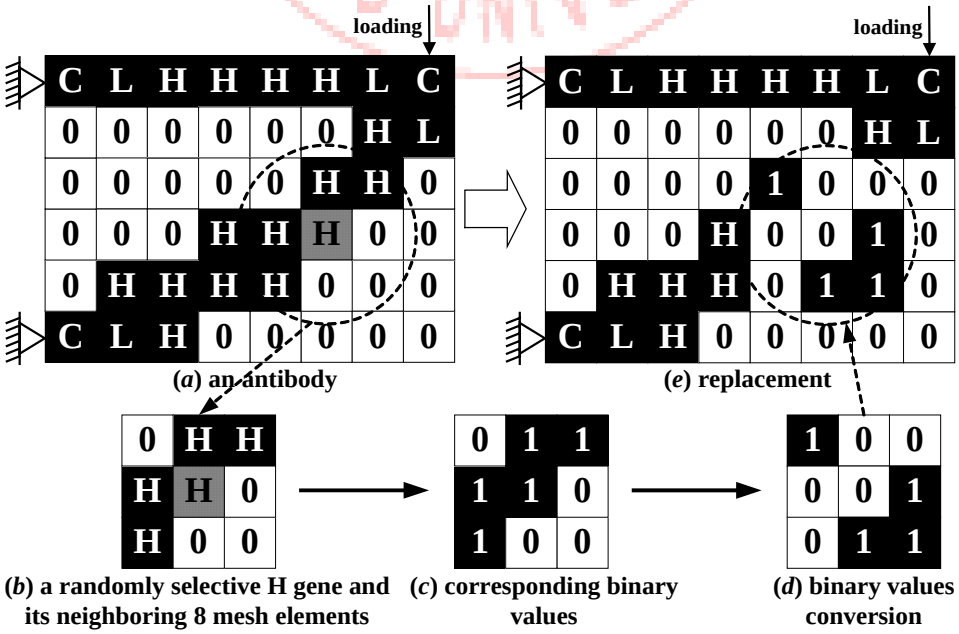


Fig. 33 Gene conversion illustration for topological optimization

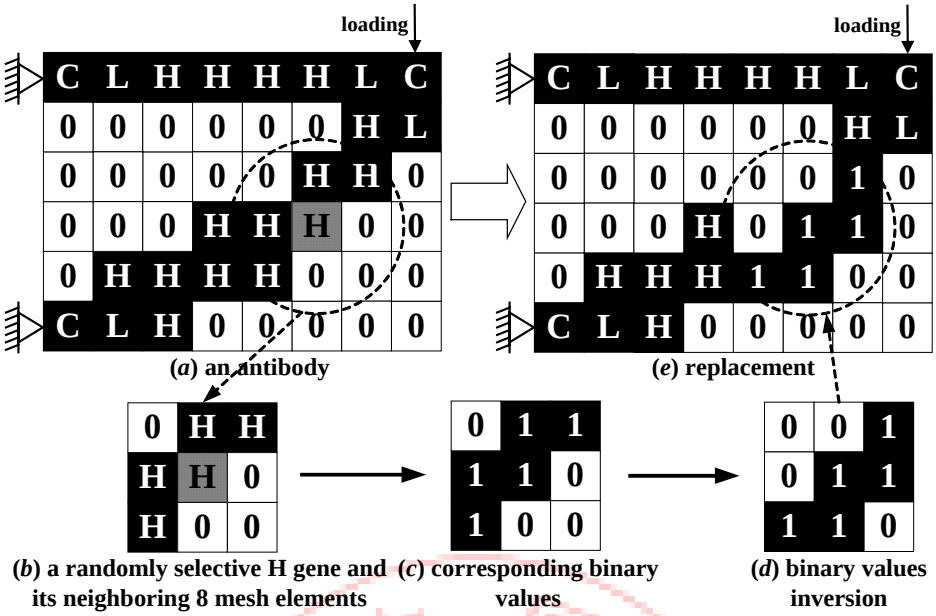


Fig. 34 Gene inversion illustration for topological optimization

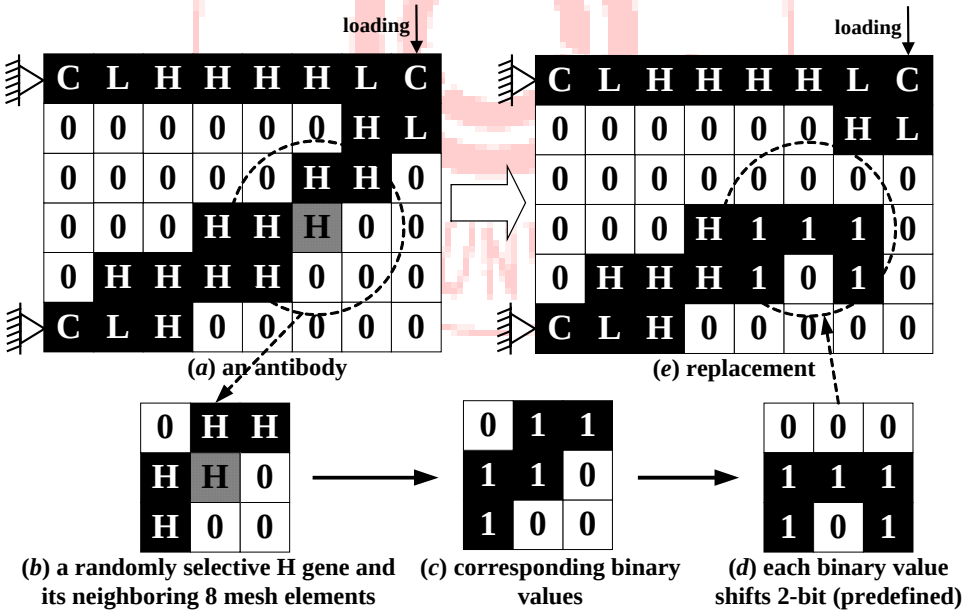


Fig. 35 Gene shift illustration for topological optimization

4. *Nucleotide addition.* As shown in Fig. 36, nucleotides insertion may be accomplished either in light- or heavy-chain genes. The nucleotide is a randomly

created binary array of predefined block size. In this study a 9 mesh elements is used to represent the nucleotide inserted at a randomly chosen point in the antibody locus. Displaced genes are then shifted to the right with excessive elements out with the antibody boundary being discarded.

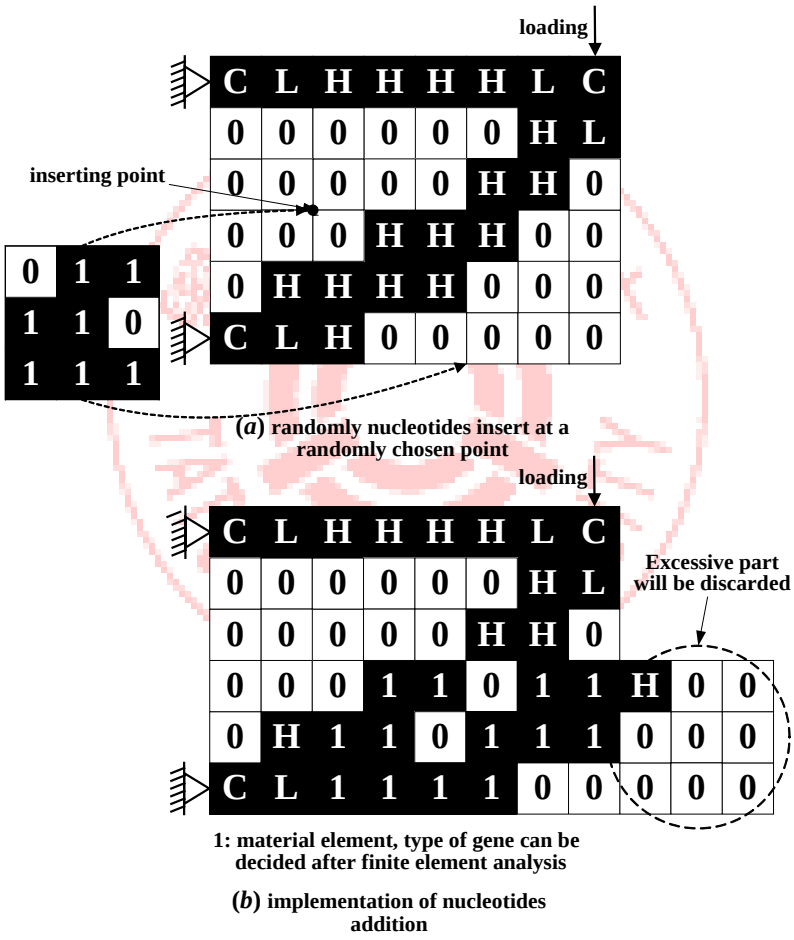


Fig. 36 Nucleotide addition illustration for topological optimization

It should be noted that the six diversification mechanisms described in this step are adopted randomly in the antibody diversity process.

[Step 9] Stopping criterion

The whole process will stop when the iteration equals a pre-defined number. Otherwise the process reverts to [Step 2] for another generation. In the final stage, the best and most diverse solutions are stored in the memory pool.

Fig. 37 demonstrates the results of the procedure of MMIA described above employing 4 antibodies/topologies. For simplicity, each topology contains 6×8 elements. Fig. 37(*a*) indicates the random generation of the 4 antibodies and the corresponding continuity analysis. Clearly, three antibodies/topologies are discontinuous except the second one. After structure modification procedure, for example removing one gray element from Ab_3 and adding several gray elements for Ab_1 and Ab_4, all the elements of the four topologies are continuous. Then, finite element calculation and gene classification (*i.e*. constant gene **C**, heavy-chain gene **H**, light-chain gene **L**, and pseudo gene **0**) for these topologies are implemented as Fig. 37(*b*) shown. Since the fourth antibody (Ab_4) has the highest affinity value, it is chosen for clonal proliferation through hypermutation as Fig. 37(*c*) illustrated. The best one matures as a plasma and memory antibody simultaneously. Subsequently germ-line DNA library is constructed by donor antibodies which selected from the antibody population and plasma antibody produced after clonal proliferation by using tournament selection. Note that fourth donor antibody

in this library is induced from memory antibodies. Fig. 37(*d*) expresses the procedure of gene fragment rearrangement. As step 7 described, new antibodies are created from the donor antibodies derived in the last step. Finally the diversification schemes chosen randomly can be employed to increase the exploration of the antibody population. Fig. 37(*e*) shows one of the possible results for antibody $NewAb_3$ since the selection of antibodies and their elements is a randomized procedure.

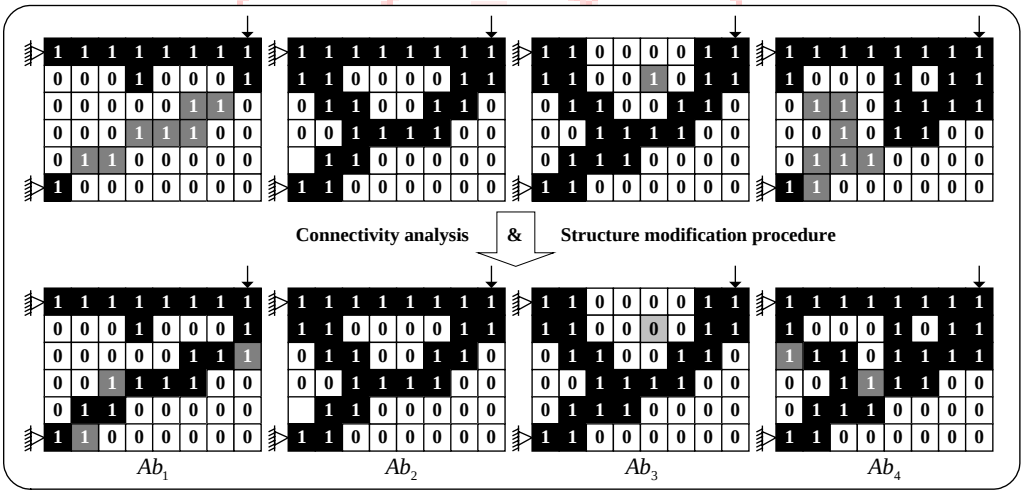


Fig. 37(*a*) Random initialization of antibody population & connectivity analysis

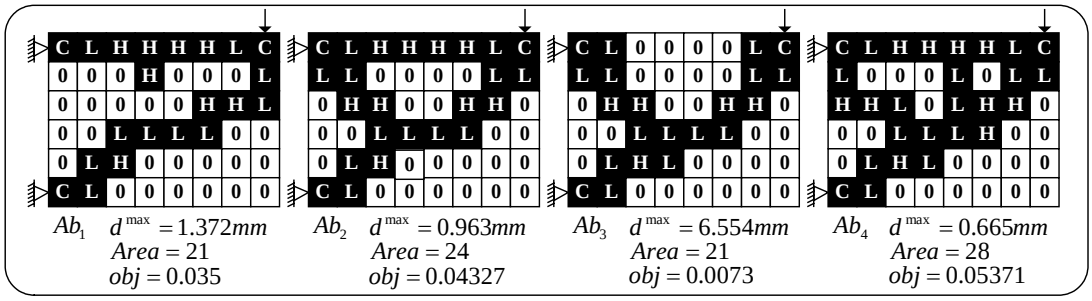


Fig. 37(*b*) Finite element calculation & gene classification

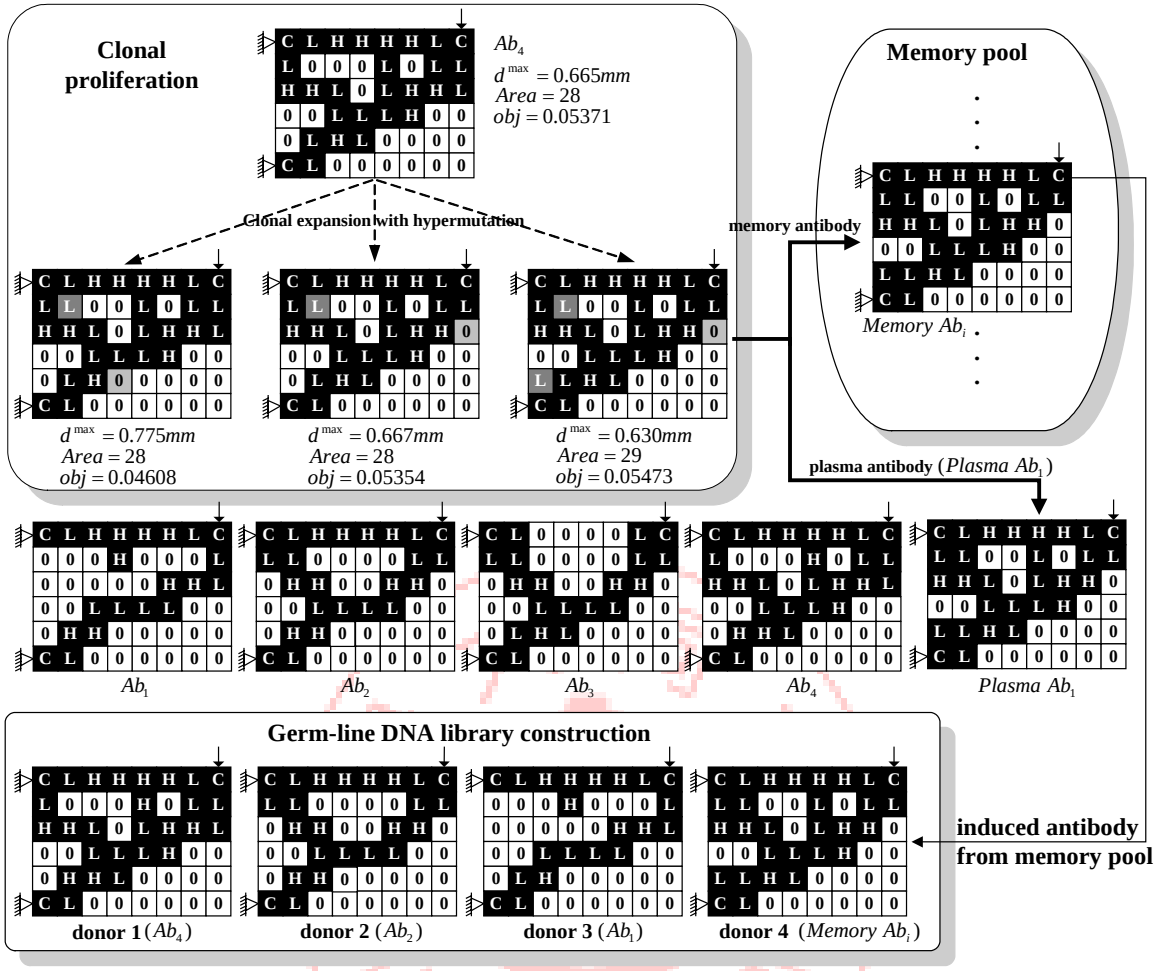


Fig. 37(*c*) Clonal proliferation, donor selection, and germ-line library construction

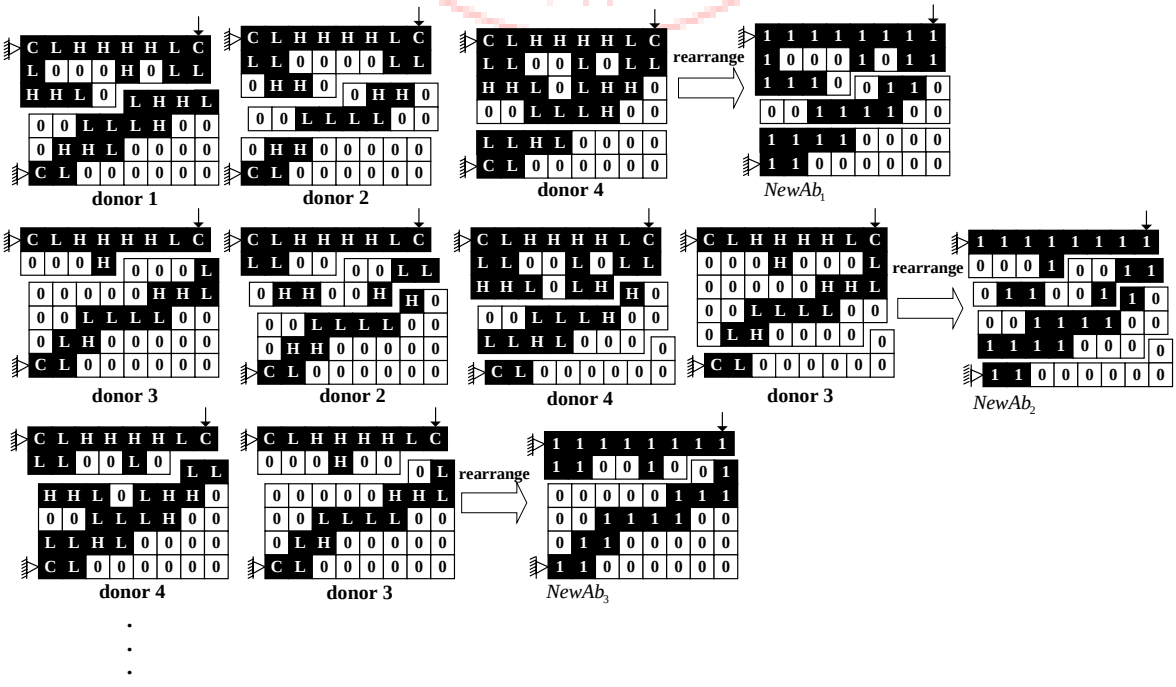


Fig. 37(*d*) Randomly gene fragment/block rearrangement

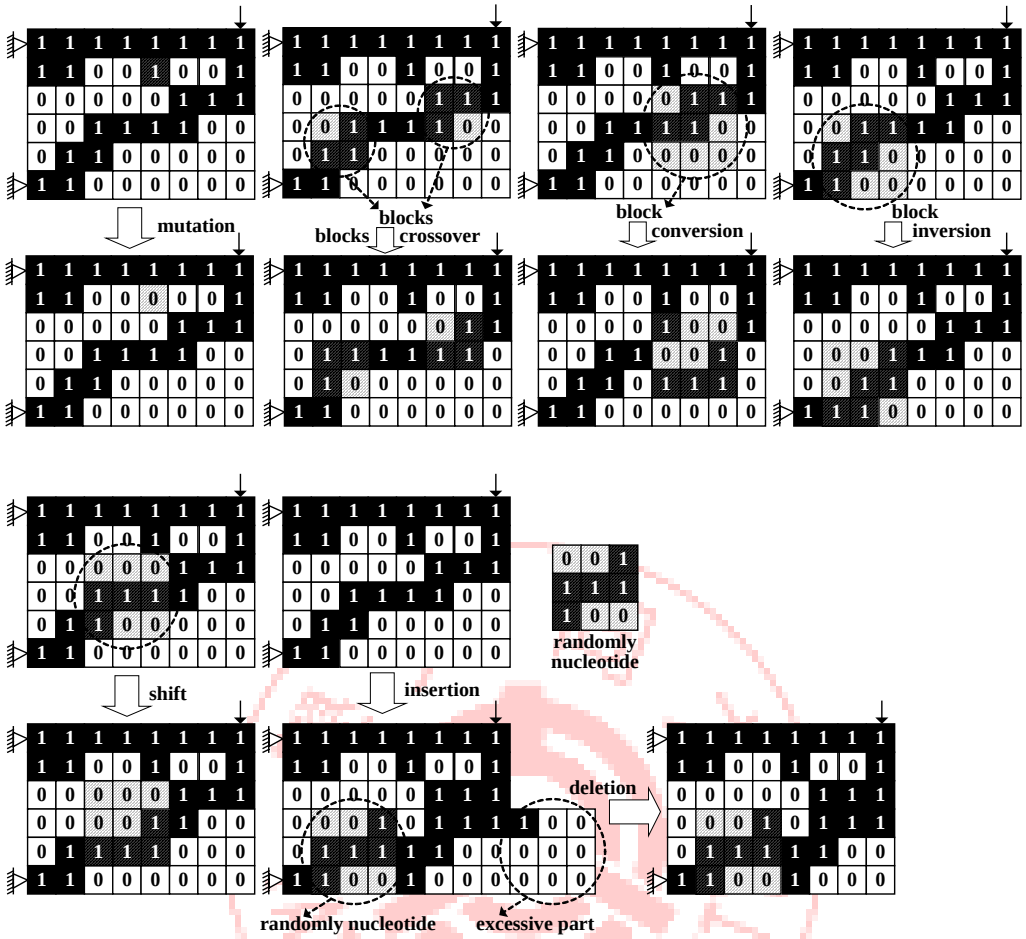


Fig. 37(*e*) Six antibody diversification schemes

Fig. 37 Illustration of the 8 steps in MMIA using four antibodies/topologies

6.3 Problems Description

In this study, two topological optimization examples were employed to evaluate the effectiveness of the proposed multi-modally immune algorithm. All the mechanical model and material properties are tabulated in Table 10. Table 11 lists the associated parameters used in the MMIA. These parameters were determined through numerical experiments after multiple simulation runs.

Table 10 Illustration of topological optimization examples

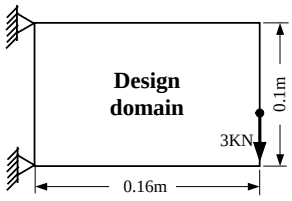
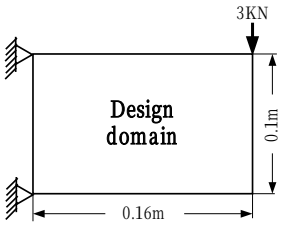
	Case 1	Case 2
Loading position	Right hand side 2/5ths from the bottom edge	Top right hand corner
Fixed position	Both end of left hand side	Both end of left hand side
Mechanical model		
Material properties	$E = 200GPa$ $\upsilon = 0.33$ allowable stress $200MPa$ thichness $t = 0.001m$	

Table 11 Immune algorithm parameters

Parameter	Case 1	Case 2
Grids of elements	20×32	20×32
Antibody length	640	640
Antibody size	50	50
Generations	500	500
Proliferation number	10	10
Tournament size	10	10
Light/heavy chain-length ratio	0.3	0.3
Hypermutation rate	0.4	0.4
Diversity rate	0.05	0.05
Inducing rate	0.2	0.2
Threshold (δ_{Ab})	0.1	0.1

Case 1

This example presents the optimization of a short cantilever plate (with aspect ratio 1.6) which is subjected to a downward concentrated loading applied at an finite element (FE) node on its right hand side 2/5ths from the bottom edge (non-symmetric structure) with its stress being constrained to $200MPa$. The support nodes on both end of left hand

side are defined to have zero displacement in the finite element (FE) analysis. In addition, the design domain is discretized according to a 20×32 plane stress element FE model. One execution of the computer model requires around 250,000 functional evaluations (500 generations by 50 antibodies/topologies by 10 clonal proliferations per generation), taking approximately 3 hours with a Pentium 4 processor running at 1.5GHz. Numerous memory antibodies/multi-modal topologies (local optimum solutions) with different configurations were derived from memory pool after 500 iterations. Fig. 38 demonstrates 16 significant topologies and their corresponding maximum displacements ($d^{\max}$), weights (*Area*) and objective values (*obj*, *i.e.* stiffness-to-weight ratio), respectively.

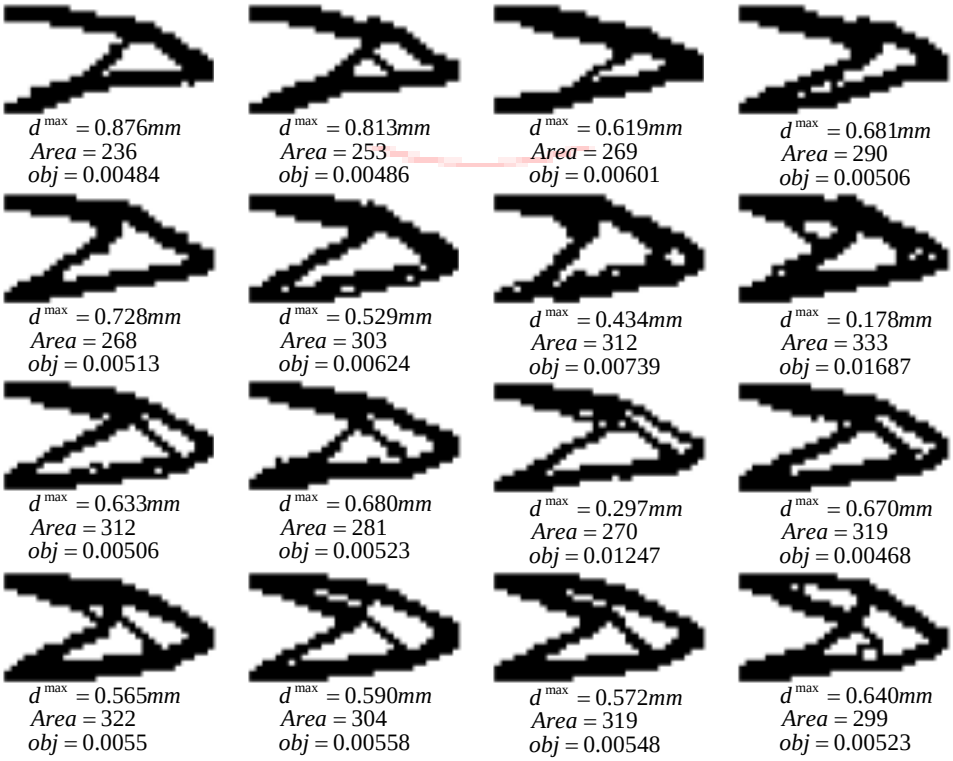


Fig. 38 Multi-modal results of case 1

Case 2

In Case 2 the downward concentrated load is applied at an FE node positioned at the top right hand corner of the design domain with all other conditions being as per those in Case 1. Again, the design domain is divided into a 20×32 plain stress element FE model and the required CPU time and functional evaluations being consistent with those in the previous case. After 500 iterations, the 9 significant memory antibodies/multi-modal topologies with their corresponding maximum displacements ($d^{\max}$), weights (*Area*) and objective values (*obj*) were illustrated in Fig. 39.

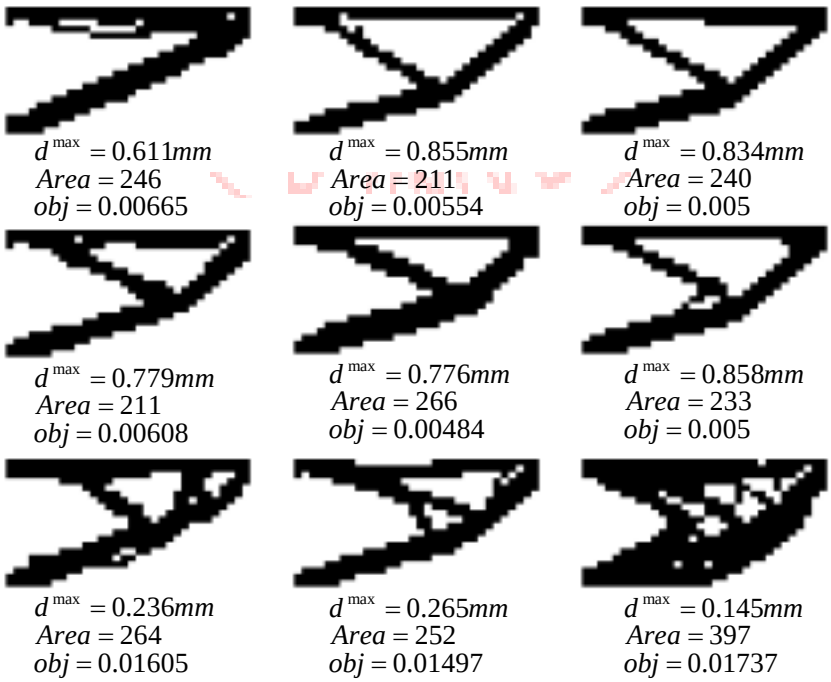


Fig. 39 Multi-modal results of case 2

6.4 Simulation Results and Discussions

As can be seen from the diverse range of resultant topologies illustrated in Fig.38 and 39, the structures show very well defined truss-like members of constant cross sectional area with large voids between members. A high proportion of these structural members have straight alignment between joints and exhibit low levels of porosity. The theoretical structures therefore provide the designer with a set of near-optimal solutions which can be easily developed into discrete truss systems. If manufacturability is the prime consideration (*i.e.* large voids between members), the designer may choose the topology shown in the left-top examples in Fig. 38 or the center-top examples in Fig. 39. On the contrary, if stiffness is the main consideration, the right-bottom examples in Fig. 38 and 39 (*i.e.* more material) are good chooses.

It should be noted the proposed MMIA is fully capable of evaluating both symmetric and asymmetric structures and is therefore more flexible than the other methods only handle symmetric structures. Moreover, the proposed method does not force a solution into a specific area of the search space, but automatically allows balanced evolution using features of the immune system to create diverse antibody/topology solutions. In addition, the inherent local search ability of the biological immune system employing clonal proliferation enhances the search speed and convergence accuracy of IA, with the substitution of increasing computation time.

6.5 Summary

In this chapter, a novel concept for applying to constrained multi-modal topological optimization has been presented by using an immune algorithm to imitate the features of a biological immune system. The proposed methodology enhances accuracy and diversity via the operation of clonal proliferation and schemata recombination implemented through the process of gene fragment rearrangement. Moreover, the potential of the proposed immune algorithm as a tool for investigating optimal topologies and for automatically creating innovative solutions to structural design problems has been illustrated in the examples presented.

CHAPTER 7

JOB-SHOP SCHEDULING OPTIMIZATION

7.1 Introduction

Scheduling problems exist almost ubiquitously in real-world applications including distribution, transportation, management, construction, engineering and manufacturing, especially in the industrial engineering world. Many scheduling problems on manufacturing industries are quite complex and very difficult to solve using conventional optimization techniques. Since the early 1950s it has been the subject of extensive research and captured the interest of researchers from several research communities including operation research and artificial intelligence, management science, as well as industrial engineering. Its main focus is concerned with the allocation of finite resources to tasks with the objective to optimize specific cost functions. An important issue is the improvement of resource utilization. It is well known that the job-shop scheduling problem (JSSP) is the most complicated and typical problem of all kinds of production scheduling problems. Scheduling for job shops is an important topic in production management. It is concerned with finding the operations and times of a set of jobs on the relevant machines subject to the processing constraints. The purpose is to improve the production efficiency and reduce the processing duration so as to gain as high profits

as possible. The JSSP may be described as follows: given j jobs, each composed of several operations that must be processed on m machines. Each operation uses one of the m machines with a deterministic processing time. Each machine can process at most one operation at a time and once an operation initiates processing on a given machine it must complete processing on that machine without interruption. Each job consists of a specific set of operations, which have to be processed according to a given technical precedence order. The operation sequences on the machines should be found to minimize the total time required to complete all jobs, *i.e.* makespan. A comprehensive survey of job shop scheduling techniques can be found in Jain & Meeran (1999). The total number of all possible schedules including feasible and infeasible solutions is $(j!)^m$. Apparently, it is impossible to exhaust all the alternatives for finding the optimal solution even though very small j and m values.

Different from previous studies, this chapter focuses on not-bit/integer string encoding optimization and applied the immune algorithm to the job-shop scheduling problems (JSSPs) with single objective. In this application, an antibody (analogous to the chromosome in GA) is encoded via operation-based representation. This representation encodes a schedule as a sequence of operations and each gene (integer number) stands for one operation. The goal of this optimization is to find the operation sequence on the machines in order to minimize the makespan, *i.e.*, the time required to

complete all jobs, and to compare with other heuristic methods for performance validation. Similar to structural topology optimization applied in the immune algorithm, some steps described in CHAPTER 4 were needed to revise by incorporating with some repairing procedures for applying to scheduling optimization problem. The procedure of revision is depicted in next section.

7.2 Immune Algorithm Revision for Scheduling Optimization

Corresponding to the JSSPs, the antigen and antibodies serve as objective (*i.e.*, makespan) and associated solutions (*i.e.*, schedules). The antibodies/schedules continuously evolve until a match is found with the specific antigen/objective (minimize the maximum makespan). The flowchart of this optimization is analogous to the topological optimization one as illustrated in Fig. 27.

[Step 1] Random initialization of antibody population

Similar to the genetic algorithms used in JSSP, the initial integer string encoding antibody population is randomly generated.

[Step 2] Antibody representation and gene classification

An operation-based representation [Gen, 1994] is used to represent the genes of an

antibody. This representation named all operations for a job with the same integer number and then interpreted it according to the order of occurrence in the given antibody. For a j jobs and m machines problem, an antibody contains $j \times m$ genes. Each job j appears in the antibody m times, and each repeating gene (*i.e.*, integer number) does not indicate a concrete operation of a job but refers to a unique operation. It is easy to see that any permutation of operations can correspond to a feasible schedule. For instance, consider a 3-job and 3-machine problem given in Table 12. As Fig. 40 shown, suppose the genes of an antibody is given randomly as [3 1 2 2 1 3 1 2 3], where numbers 1, 2, and 3 stand for jobs j1, j2, and j3, respectively. Because each job needs three operations/machines, it appears exactly three times in an antibody. Based on the machine sequence and processing time given in Table 12, the machine sequence for job j1is 1-2-3, for job j2 is 1-3-2, and for job j3 is 2-1-3, while the processing time for job j1is 3-3-2, for job j2 is 1-5-3, and for job j3 is 3-2-3. Therefore, the corresponding machine list and time list of given antibody shown in Fig. 40 are [2 1 1 3 2 1 3 2 3] and [3 3 1 5 3 2 2 3 3], respectively.

Table 12 Example data of 3-job and 3-machine JSSP

Processing time operations				Machine sequence			
Job	1	2	3	Job	1	2	3
j1	3	3	2	j1	m1	m2	m3
j2	1	5	3	j2	m1	m3	m2
j3	3	2	3	j3	m2	m1	m3

Antibody$_i$	**3**	**1**	**2**	**2**	**1**	**3**	**1**	**2**	**3**
Machine list:	**2**	**1**	**1**	**3**	**2**	**1**	**3**	**2**	**3**
Time list:	**3**	**3**	**1**	**5**	**3**	**2**	**2**	**3**	**3**
Chain type:	**H**	**H**	**H**	**H**	**H**	**L**	**H**	**L**	**L**

Fig. 40 Antibody representation for scheduling problem

In the same manner as in biological immune systems, each antibody/schedule is separated into two different kinds of genes: a heavy-chain gene (**H**) and a light-chain gene (**L**). These genes are classified into heavy-chains or light-chains according to *i*) a default light/heavy chain-length ratio determined by the user and *ii*) job/gene order of occurrence appearing in the identical machine. Take the 3×3 JSSP mentioned above for example, an antibody is given at random as [3 1 2 2 1 3 1 2 3] and its corresponding machine list is [2 1 1 3 2 1 3 2 3], assume that the user-defined light/heavy chain-length ratio is 0.3 and there are *j* jobs appearing in the identical machine (see from machine list). The number of light-chain gene is defined by the rounded of light/heavy chain-length ratio multiplied by number of jobs plus 0.5 (*i.e.*, rounded of $(0.3\times j)+0.5$) and assigned from later part of the jobs appearing in the same machine. The other genes except light-chain gene are defined as heavy-chain. Suppose there are 3 jobs in the JSSP, the number of light-chain gene is defined to 1 (integer of $(0.3\times 3)+0.5$) and the number of heavy-chain gene is 2 (jobs minus the number of light-chain gene). For machine number 1 in machine list, the corresponding jobs/genes are 1-2-3, therefore, its chain type is defined as H-H-L. And the same manner for machine number 2, the corresponding

jobs/genes and chain type are 3-1-2 and H-H-L respectively, and so on. Hence, the corresponding chain list of given antibody is [H H H H H L H L L] (see Fig. 40, chain list).

[Step 3] Calculating combinatorial intensity

The antibody-to-antigen affinity value ($AgAb_i$) is employed to illustrate the combinatorial intensity between antigen/objective and the *ith* antibody/schedule. In this chapter, the maximum makespan of a schedule is used as the affinity value, and it should be minimized. The makespan is obtained by following decoding procedure: The first gene/job is scheduled/decoded first, then the second gene, and so on. Each scheduling gene/job is allocated in the best available processing time for corresponding machine. The process is repeated until all genes are scheduled. Thus, the Gantt chart for this given antibody ([3 1 2 2 1 3 1 2 3]) is drawn in Fig. 41 with its maximum makespan of 14.

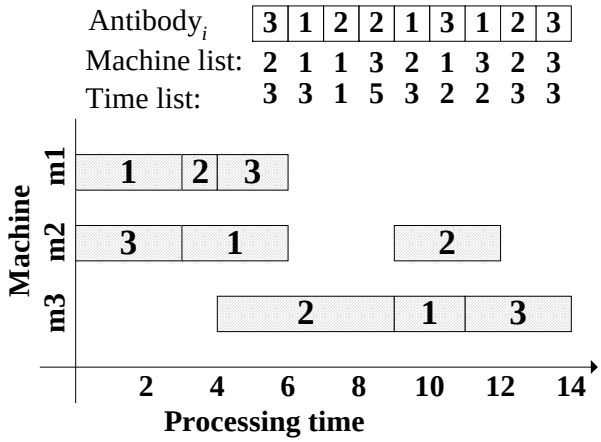


Fig. 41 Decoding for an antibody/schedule (Gantt chart)

[Step 4] Clonal proliferation

In the proposed scheme, the most-matched antibody which has minimal maximum makespan derived from [Step 3] is chosen for hypermutation during the clonal proliferation process, with a user-defined hypermutation rate and proliferation number. Again, hypermutation only takes place in light-chain genes (**L**). For a 3 jobs JSSP, if a light-chain gene mutated form job j3 to j1 (j1 is generated randomly from all jobs), the original job j1 which has the same machine number with j3 should also be repaired to j3 (reciprocal exchange within the same machine) in order to avoid yielding illegal or infeasible schedules (that is, some jobs are repeated more than once while other jobs get lost in the identical machine). After the hypermutation process, the proliferated antibodies which have better affinity than un-proliferated antibody are differentiated into plasma antibody and memory antibody preserved and updated in the memory pool. Further, the resulting poor proliferated antibodies are neglected. Resulting plasma antibodies combined with the remaining antibodies derived from [Step 3] are then proceed to Step 5 for donor antibody selection according to their affinity value. In the memory pool, only highest affinity antibody can be survived. On the other hand, those antibodies with low affinity and repeat will be removed from the memory pool. In addition, a part of memory antibodies are induced into the germ-line DNA library (as per Step 6) according to a user-defined inducing rate.

[Step 5] Tournament selection for donor antibodies

The proposed algorithm uses a tournament selection scheme to select donor antibodies exhibiting higher affinity values to assemble germ-line DNA libraries.

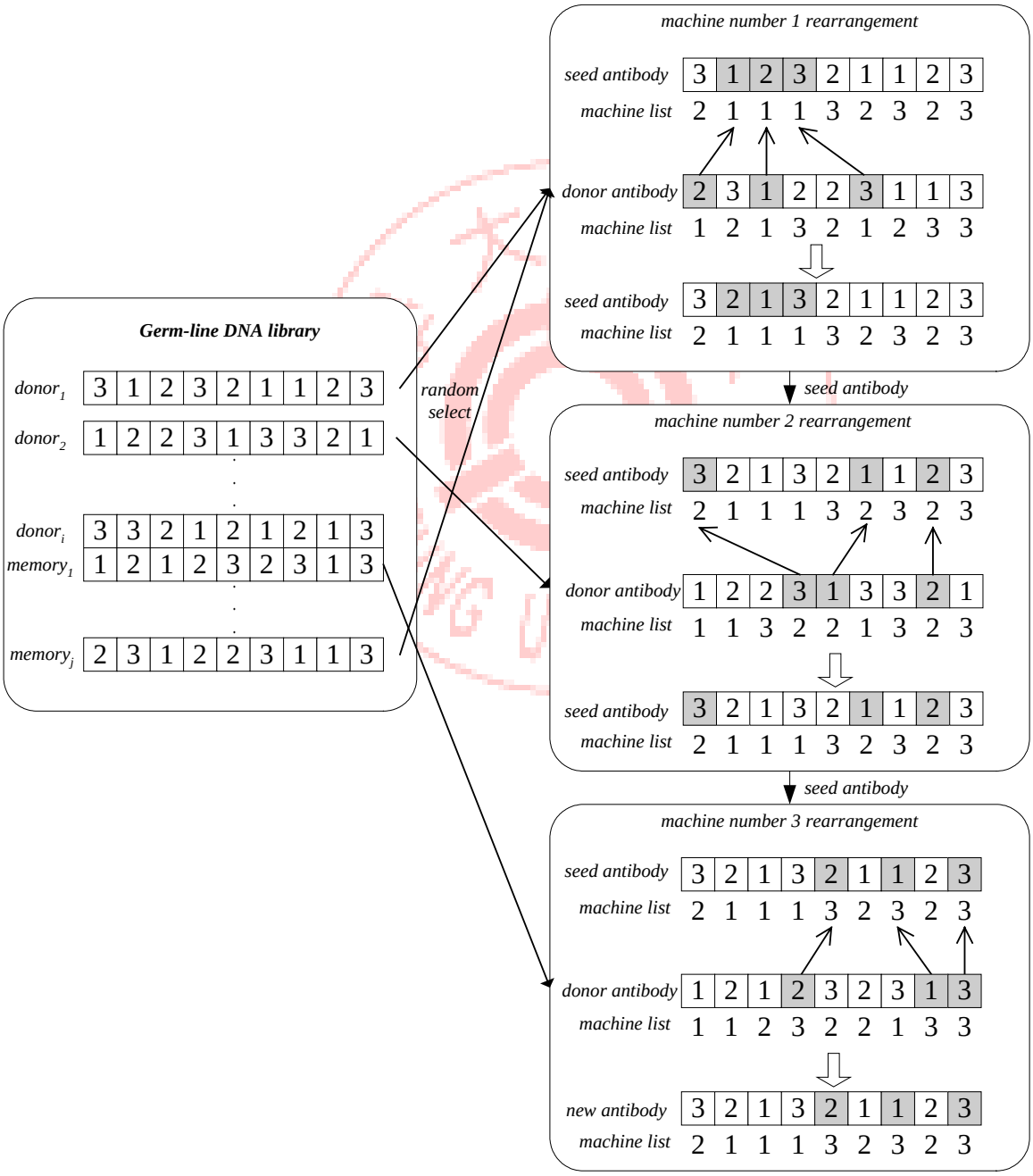


Fig. 42 Illustration of fragmental rearrangement for scheduling problem

[Step 6] Germ-line DNA library construction

As to MOIA, the components of germ-line DNA library are constructed from the memory antibodies and the donor antibodies.

[Step 7] Randomly gene fragment rearrangement

In this optimization, new antibodies/schedules are rearranged via gene fragments picked randomly from germ-line DNA library. A concept of machine-based rearrangement is used for producing a new antibody and as shown in Fig. 42 considering a 3-job × 3-machine job-shop scheduling problem. First, randomly chosen a seed antibody and a donor antibody from the library, and then assigned the genes/jobs of donor antibody with first machine number (number 1) to the seed in corresponding gene locus. Next, chosen a randomly donor antibody again for assigning the genes with second machine number (number 2) to the seed antibody in corresponding gene locus. Repeating processes till all machine numbers are assigned. The new antibody is produced once all machine numbers are assigned. It is easy to see that any assign of fragments will generate a feasible schedule without repairing procedures.

[Step 8] Antibody diversification

Because of different antibody representation, the antibody diversity mechanisms are

needed to revise for applying to this scheduling optimization, and some of these revised mechanisms need repairing procedures.

1. *Somatic point mutation.* As depicted in Fig. 43, in terms of not-binary/integer encoding representation, this means swapping two randomly selected heavy-chain genes according to a pre-defined diversity probability.

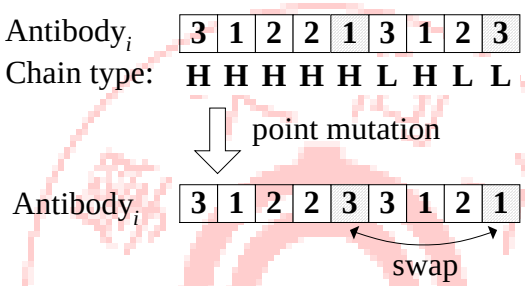


Fig. 43 Somatic point mutation illustration for scheduling problem

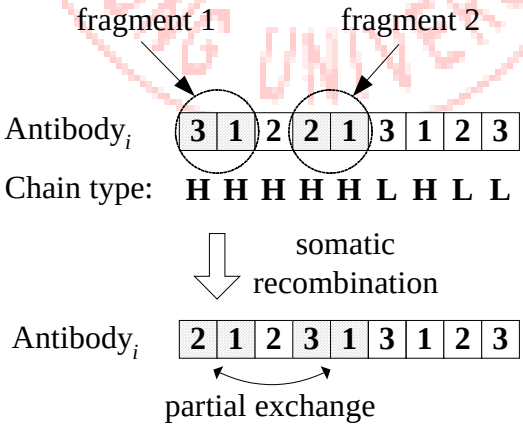


Fig. 44 Somatic recombination illustration for scheduling problem

2. *Somatic recombination*. As shown in Fig. 44, two heavy-chain gene fragments with the same length are randomly selected from an antibody. After which a partial

exchange is performed between the two fragments according to a pre-defined diversity probability.

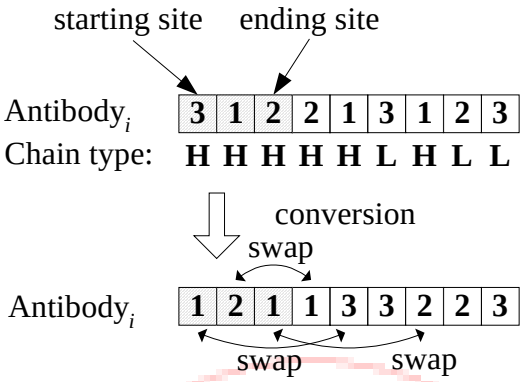


Fig. 45 Gene conversion illustration for scheduling problem

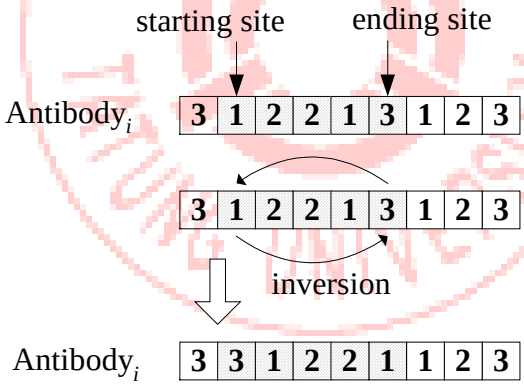


Fig. 46 Gene inversion illustration for scheduling problem

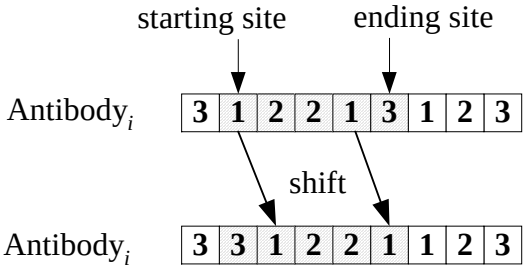


Fig.47 Gene shift illustration for scheduling problem

3. *Gene conversion*, *gene inversion*, and *gene shift*. Following predefined probabilities, gene conversion, gene inversion, and gene shift is performed using a randomly selected heavy-chain gene (see Fig. 45 to 47). Note that the starting and ending sites were randomly generated, and the number of shift genes was predefined. This type of diversification scheme results in a global search effect. In gene conversion (depicted in Fig. 45), those heavy-chain genes between the starting and ending site were swapped with the other heavy-chain gene chosen from antibody at random. As to the gene inversion operator shown in Fig. 46, randomly chosen gene/job fragment inverses sequentially its gene positions from front to rear and from rear to front. Fig. 47 illustrates gene shift operation. Following the predefined number of shift, the selected gene/job fragment right-shift their gene locations with excessive positions being reallocated from rear to front.

4. *Nucleotide addition.* Nucleotides insertion may be accomplished either in light- or heavy-chain genes. The nucleotide is a randomly created natural number of predefined size representing the genes/jobs and inserted at a randomly chosen inserting site in the antibody locus. Displaced genes are then shifted to the right with excessive genes out with the antibody boundary being repairing. In the repairing process, first *m* of each job will be preserved and excessive part will be discarded as depicted in Fig. 48.

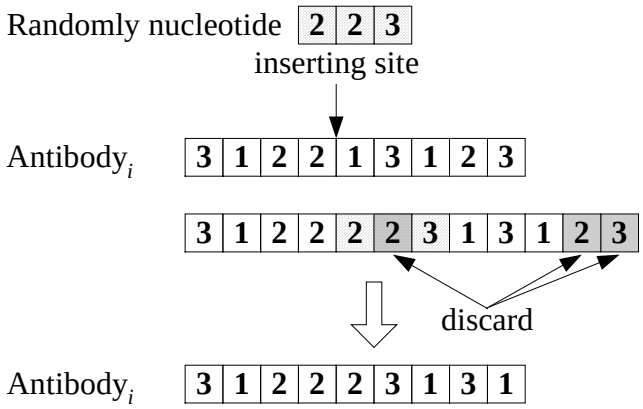


Fig. 48 Nucleotide addition illustration for scheduling problem

It should be noted that the six diversification mechanisms described in [Step 8] are adopted randomly in the antibody diversity process.

[Step 9] Stopping criterion

The whole process will stop when the iteration equals a pre-defined number. Otherwise the process reverts to [Step 2] for another generation. In the final stage, the best solutions are stored in the memory pool.

7.3 Experimental Results and Discussions

For carrying out the necessary computations and evaluating the performance of the proposed immune algorithm, the program for computing JSSP was developed using C++ language and running with a Pentium 3 processor at 1.0GHz. In this study, 27 benchmark instances of different size (operations) collected from the OR-Library

(http://www.mscmga.ms.ic.ac.uk) including two classes of standard JSSP test problems [Fischer and Thompson, 1963; Lawrence, 1984] are considered to illustrate the effectiveness of the proposed algorithm. These instances are widely used in the literatures, and each of instances is run randomly 10 times. The associated user-defined parameters utilized in proposed immune algorithm for scheduling optimization are tabulated in Table 13, each with the same parameter setting. Table 14 summaries the computational results, it lists the instance name and its size (job × machine), the best known solution, the solution obtained by proposed immune algorithm (IA), and the solution computed by other algorithms such as genetic algorithms (GA) [Dorndorf and Pesch, 1995; Wang and Zheng, 2002; Gonçalves *et al.*, 2002; Croce *et al.*, 1995], stimulated annealing (SA) [Kolonko, 1999; Van Laarhoven, 1992], and tabu search (TS) [Dell' Amico and Trubian, 1993]. Table 15 shows the corresponding best schedules obtained by proposed immune algorithm. In the random runs, the compared results (Table 15) shown that the average relate error of the proposed immune algorithm over 10 random runs compared to other for the best known so far is very small, and the optimal or near-optimal solutions is found for 16 of the 27 instances, and apart of them are found very quickly for middle-size instances such as the size/operations small than 15 × 5. For remaining large-size instances the results are also very close to that of other comparison algorithms.

Table 13 Immune algorithm parameters for scheduling problem

Instance size (job × machine)	6x6	10x5	15x5	20x5	10x10	15x10	20x10	30x10	15x15
Iteration number	100			500			1000		
Antibody population size	job × machine			2 × (job × machine)					
Antibody length	36	50	75	100	100	150	200	300	225
Diversity probability	0.1			0.1			0.1		
Hypermutation rate	0.5			0.5			0.5		
Light/heavy chain-length ratio	6:4			5:5			5:5		
Number of proliferation	6			8			10		
Inducing ratio	0.1			0.2			0.2		
Bit number in Gene shift	2-bit			2-bit					
Bit number of nucleotide	random between 1 to number of jobs								
Tournament size	(job × machine)/10			(2 × job × machine)/10					

Table 14 Computational results

Name	Size	Best known	IA	GA				SA		TS
				Dorndorf and Pesch (1995)	Wang and Zheng (2002)	Gonçalves *et al.* (2002)	Croce *et al.* (1995)	Kolonko (1999)	Van Laarhoven *et al.* (1992)	Dell' Amico *et al.* (1993)
FT06	6x6	55	55		55		55		55	55
FT10	10x10	930	955	960	930	936	946		930	930
FT20	20x5	1165	1201	1249	1165	1177	1178		1165	1165
LA01	10x5	666	666	666	666	666	666		666	666
LA02	10x5	655	659	681		666	666			655
LA03	10x5	597	597	620		597	666			597
LA04	10x5	590	593	620		590				590
LA05	10x5	593	593	593		593				593
LA06	15x5	926	926	926	926	926	926		926	926
LA07	15x5	890	890	890		890				890
LA08	15x5	863	863	863		863				863
LA09	15x5	951	951	951		951				951
LA10	15x5	958	958	958		958				958
LA11	20x5	1222	1222	1222	1222	1222	1222		1222	1222
LA12	20x5	1039	1039	1039		1039				1039
LA13	20x5	1150	1150	1150		1150				1150
LA14	20x5	1292	1292	1292		1292				1292
LA15	20x5	1207	1207	1237		1207				1207
LA16	10x10	945	946	1008	945	977	979	945	956	945
LA17	10x10	784	784	809		787				784
LA18	10x10	848	855	916		848				848
LA19	10x10	842	857	880		857		842		842
LA20	10x10	902	911	928		910				902
LA21	15x10	1046	1088	1139	1058	1047	1097	1046	1063	1047
LA26	20x10	1218	1257	1278	1218	1218	1231		1218	1218
LA31	30x10	1784	1784		1784	1784	1784		1784	1784
LA36	15x15	1268	1316	1373	1291	1305	1305	1268	1293	1268

Table 15 The corresponding best schedules

Name	Size	Best known	IA	Corresponding best schedule
FT06	6x6	55	55	[3 2 3 1 4 6 2 3 6 6 4 2 5 5 3 2 4 5 1 1 5 6 4 1 2 1 3 2 5 3 4 5 4 6 1 6]
FT10	10x10	930	955	[7 4 6 6 10 8 2 7 5 6 4 7 5 7 4 9 9 10 2 1 8 5 7 6 1 10 3 7 4 6 8 2 8 3 1 5 9 3 10 7 9 8 6 6 6 2 3 9 7 1 4 4 7 3 4 3 5 5 9 1 8 10 1 2 2 5 6 2 9 10 9 4 9 10 1 8 5 7 2 2 5 3 8 9 4 3 1 3 1 1 8 6 10 10 4 5 2 10 8 3]
FT20	20x5	1165	1201	[20 5 16 20 2 5 16 1 19 20 20 15 6 2 16 9 19 15 20 8 2 19 5 17 19 10 12 17 15 10 5 14 10 14 17 17 18 19 6 6 1 13 9 12 5 8 11 2 18 14 7 11 7 6 11 12 10 13 1 8 2 3 1 3 7 18 13 16 14 1 4 3 17 16 12 7 15 4 11 10 12 3 13 8 13 7 4 15 8 11 3 9 14 6 9 18 18 4 4 9]
LA01	10x5	666	666	[6 7 5 10 7 5 6 10 2 2 7 9 5 6 10 3 6 5 1 1 1 4 4 9 6 9 2 2 9 10 9 1 2 8 4 3 8 10 4 5 7 4 7 8 3 1 3 8 8 3]
LA02	10x5	655	656	[3 1 5 2 7 2 9 10 2 7 3 9 4 10 9 8 10 6 8 8 5 3 3 1 4 2 5 6 7 2 5 7 4 6 10 1 3 9 7 1 1 8 5 8 9 6 6 4 4 10]
LA03	10x5	597	597	[4 4 2 7 8 1 8 10 6 9 6 2 3 5 7 9 7 4 1 10 9 2 3 5 4 7 1 9 6 5 6 7 1 5 8 6 2 3 8 3 10 9 4 5 1 10 8 3 10 2]
LA04	10x5	590	593	[5 10 7 3 5 5 6 6 1 4 9 1 2 2 2 10 3 8 7 5 6 8 6 8 7 10 10 9 9 1 4 3 4 3 8 5 6 7 1 7 9 8 3 2 4 1 10 2 9 4]
LA05	10x5	593	593	[8 2 1 9 4 6 2 9 8 6 2 3 6 1 4 3 1 8 10 4 7 7 8 4 9 7 3 5 10 9 6 1 4 3 2 7 7 5 2 1 3 6 9 10 8 5 5 5 10 10]
LA06	15x5	926	926	[5 7 14 7 1 5 6 11 11 4 10 3 10 1 12 8 15 9 11 12 12 12 6 14 12 11 10 14 10 3 5 14 10 14 5 1 2 11 6 4 4 3 8 13 9 7 4 1 1 8 5 2 4 6 6 7 8 9 9 7 8 15 2 13 9 3 2 2 13 3 15 15 13 15 13]
LA07	15x5	890	890	[12 15 13 3 7 4 15 4 1 4 12 1 3 5 8 9 4 14 8 2 15 10 15 11 4 10 6 9 10 9 14 1 6 2 5 14 14 1 13 3 11 15 7 8 7 2 13 9 13 6 3 7 2 3 13 11 11 1 12 12 10 5 5 8 5 10 2 6 12 14 9 8 6 7 11]
LA08	15x5	863	863	[14 8 11 6 9 5 11 4 14 14 13 10 12 7 8 10 7 8 8 5 15 11 9 12 3 6 5 3 5 8 1 1 4 4 9 1 6 2 2 14 9 13 13 12 4 4 15 9 13 14 6 15 10 2 1 15 7 3 15 7 13 11 7 10 12 3 2 5 3 11 6 12 2 1 10]
LA09	15x5	951	951	[4 1 10 3 4 11 14 12 5 12 15 13 15 11 2 8 8 13 6 3 4 5 2 5 13 10 11 9 15 14 7 10 10 14 6 7 7 13 6 12 2 6 1 1 1 5 9 3 12 2 15 6 3 7 4 3 8 14 10 8 9 4 7 12 2 11 15 13 14 9 8 9 5 11 1]
LA10	15x5	958	958	[15 1 7 3 12 15 9 3 2 10 6 13 11 14 5 13 8 4 7 6 12 3 13 4 8 4 14 2 5 8 7 2 11 11 7 10 10 9 9 15 10 12 9 14 14 12 2 7 13 15 1 4 14 8 5 5 6 1 11 3 8 12 1 6 1 15 4 6 5 9 2 11 10 3 13]

LA11	20x5	1222	1222	[13 11 20 11 20 8 15 1 14 2 7 18 5 11 4 9 7 20 4 12 6 9 15 7 18 3 13 8 20 17 18 15 10 16 5 16 16 8 14 6 12 15 3 17 2 16 2 1 13 11 5 6 6 10 5 16 9 4 3 11 14 12 1 15 18 17 18 12 6 13 4 14 19 9 10 2 1 7 8 9 13 7 4 2 19 17 19 17 20 10 3 12 1 8 14 5 10 19 3 19]
LA12	20x5	1039	1039	[1 14 7 20 2 5 19 14 9 15 20 16 8 11 6 20 19 19 5 6 7 9 13 1 18 12 8 15 4 4 13 9 3 18 18 2 5 9 11 20 3 15 9 12 11 16 10 16 13 3 11 12 4 7 1 17 1 5 17 3 5 2 14 6 8 8 12 13 7 14 17 2 15 4 6 20 10 17 3 17 7 18 10 18 6 19 19 15 8 12 16 16 11 4 14 10 1 2 13 10]
LA13	20x5	1150	1150	[7 20 8 9 6 15 17 4 5 3 12 18 11 16 17 8 3 3 2 8 7 2 19 9 11 19 6 12 13 11 14 4 14 19 20 20 7 13 4 18 10 9 6 10 2 1 19 4 10 14 18 13 5 17 18 16 1 12 10 12 19 17 4 13 14 17 6 15 2 15 8 7 8 7 1 15 5 6 9 1 1 5 16 10 20 9 14 13 2 16 11 12 11 3 18 15 3 20 16 5]
LA14	20x5	1292	1292	[5 7 8 17 3 8 1 11 7 20 2 3 20 14 13 9 15 19 1 20 18 11 2 14 9 13 9 4 10 14 2 15 17 2 20 12 10 11 3 2 7 18 11 13 6 17 12 6 7 7 6 14 15 11 5 19 5 8 5 10 8 5 1 17 16 3 18 8 15 1 4 18 4 16 16 9 17 9 19 1 18 15 6 10 16 10 13 4 19 4 19 12 3 20 16 13 14 12 6 12]
LA15	20x5	1207	1207	[2 7 3 13 20 14 13 2 12 11 9 15 2 11 11 7 20 12 19 12 4 13 4 20 12 4 1 7 11 1 15 1 6 15 8 2 5 10 3 4 19 6 9 13 18 8 10 17 19 3 2 11 16 16 6 10 8 5 12 10 18 5 8 19 6 1 1 8 19 13 7 20 14 5 18 9 15 15 14 4 14 6 5 10 16 17 20 17 17 3 9 9 16 3 17 18 18 14 7 16]
LA16	10x10	945	946	[10 9 10 6 3 2 6 8 3 7 1 3 3 6 4 5 7 9 8 6 7 5 3 7 2 2 10 1 1 3 7 4 3 1 8 6 1 4 10 5 3 9 10 4 1 2 7 5 6 5 3 8 4 9 2 5 8 4 9 1 7 10 5 4 4 6 8 6 5 6 1 2 3 2 2 5 10 7 7 10 9 1 9 4 7 8 8 8 10 5 4 9 10 9 2 6 9 8 1 2]
LA17	10x10	784	784	[7 2 4 8 6 2 1 1 10 5 3 4 7 4 5 1 8 3 2 8 2 4 3 5 6 9 7 5 8 6 6 5 4 4 10 1 2 6 7 3 9 5 9 3 9 5 1 2 10 7 3 10 6 8 2 8 2 9 3 1 2 9 8 3 10 3 9 7 1 9 7 9 1 4 6 6 7 5 10 5 7 2 10 9 8 8 7 4 8 4 6 10 6 4 1 10 5 1 3 10]
LA18	10x10	848	855	[9 7 10 9 9 10 5 2 7 5 10 7 7 6 3 3 8 1 8 9 4 6 3 6 5 1 1 9 4 8 8 6 3 8 3 10 1 6 5 10 3 9 8 9 6 1 3 3 10 3 4 3 4 10 6 7 5 1 1 2 2 1 1 4 1 4 2 5 6 2 2 5 4 8 4 5 9 5 7 6 6 5 4 2 2 7 8 7 7 9 8 2 10 10 2 10 8 7 9 4]
LA19	10x10	842	857	[8 9 6 10 5 1 3 2 2 7 4 10 1 6 1 3 10 4 5 10 3 2 9 5 9 9 1 1 4 6 5 7 5 7 8 6 2 8 1 9 4 4 4 3 4 9 2 4 5 8 5 9 10 2 9 7 1 4 7 4 7 5 6 2 7 3 8 6 5 3 7 1 3 7 3 1 6 7 2 6 5 9 8 8 9 6 2 1 8 2 8 6 10 10 10 8 10 10 3 3]
LA20	10x10	902	911	[1 9 6 6 10 2 5 8 8 6 9 10 5 7 10 5 2 6 5 9 5 7 1 8 8 7 10 3 2 6 1 2 7 10 1 9 10 1 6 3 5 9 6 6 8 9 1 10 5 2 4 3 6 4 5 9 7 3 3 7 4 4 3 8 4 1 1 6 1 10 4 1 4 9 9 4 9 3 8 2 5 5 7 10 2 2 7 7 3 3 2 2 8 10 3 7 4 8 4 8]

LA21	15x10	1046	1088	[6 7 1 2 12 7 8 5 10 7 11 3 11 11 12 15 11 8 7 3 11 6 6 13 10 9 15 15 14 12 8 9 5 2 11 1 2 1 3 12 2 8 15 12 14 12 12 1 13 4 8 4 10 8 5 8 3 4 7 15 4 11 2 3 12 13 7 9 14 7 5 2 4 2 13 1 14 14 2 10 5 11 6 7 10 5 12 2 15 9 13 6 3 4 10 14 3 13 9 13 10 6 1 3 3 6 9 6 6 14 13 12 1 9 3 5 15 4 4 10 13 14 8 11 7 15 10 4 5 4 10 2 7 6 8 15 14 5 8 9 13 15 14 1 1 11 1 5 9 9]
LA26	20x10	1218	1257	[9 15 11 19 2 7 1 4 5 19 8 1 17 17 1 16 8 10 6 4 9 17 13 7 12 14 18 3 14 4 11 5 15 5 16 3 18 19 11 10 10 18 4 10 10 19 9 7 7 4 18 4 1 2 1 1 12 15 11 19 8 2 12 13 4 16 14 8 2 14 2 20 1 9 20 1 4 18 16 5 8 2 13 4 1 14 18 13 8 7 19 5 17 7 16 15 10 12 15 6 17 10 3 14 3 6 4 18 12 16 11 12 20 3 11 17 15 20 2 9 17 2 12 2 19 5 9 13 3 15 17 6 7 8 10 18 12 13 14 7 3 6 15 18 9 3 11 15 13 10 16 5 14 6 16 6 8 3 9 11 15 14 2 20 20 18 19 9 20 14 20 6 20 5 5 17 13 20 6 3 13 1 6 12 7 17 10 13 19 11 12 7 16 5 8 16 11 8 9 19]
LA31	30x10	1784	1784	[15 25 5 4 13 25 19 26 3 19 23 29 12 20 8 14 30 21 28 4 20 2 8 27 18 29 11 15 27 20 12 30 3 7 26 10 22 22 18 7 20 8 25 21 4 29 15 1 29 5 24 16 28 12 4 1 3 2 16 1 4 22 25 2 9 12 15 8 18 26 6 27 29 30 10 4 9 30 25 14 28 11 26 25 25 3 7 3 17 25 28 8 14 13 2 30 9 5 21 8 10 22 2 14 11 14 6 18 17 20 11 12 21 15 2 21 21 29 24 10 11 16 16 18 6 10 5 9 30 2 20 17 5 23 24 21 26 14 28 9 23 28 12 26 20 4 11 22 18 16 24 10 29 23 6 20 15 13 7 19 3 17 19 14 22 25 12 24 10 5 23 19 5 11 17 16 16 13 1 28 13 30 1 5 22 18 1 24 17 24 13 12 8 6 17 19 17 4 11 3 13 24 11 30 24 4 27 26 29 7 16 27 20 13 1 9 23 20 30 6 17 5 27 23 6 29 21 6 30 22 15 2 14 28 6 16 28 24 6 2 18 10 11 3 7 26 15 19 28 19 7 1 3 4 19 8 10 13 29 26 27 9 15 14 22 3 5 23 1 22 7 27 12 18 1 27 27 14 13 18 9 2 8 26 19 12 8 9 15 7 21 10 16 17 21 7 23 25 23 9]
LA36	15x15	1268	1316	[4 11 8 5 8 2 8 12 13 14 7 1 7 11 6 9 9 15 12 5 13 2 2 7 9 15 15 11 9 7 11 8 13 1 7 13 1 13 6 6 5 10 8 8 5 4 12 11 4 2 12 4 15 1 10 3 5 1 10 15 3 7 5 3 14 2 11 15 12 14 6 15 6 5 8 14 4 1 9 15 2 7 12 7 5 10 4 9 8 10 5 3 7 10 11 15 12 11 1 10 14 9 3 15 12 13 3 3 3 1 13 15 10 4 15 12 12 7 5 11 8 13 2 2 1 2 14 3 7 4 10 7 4 14 9 9 11 6 5 3 15 11 14 11 6 13 4 8 6 4 14 3 2 12 12 9 14 12 15 9 8 13 1 7 4 6 8 1 6 1 6 6 10 9 10 10 5 1 8 11 2 6 3 12 2 8 13 13 10 13 2 13 5 9 4 5 12 14 13 5 1 7 14 14 8 15 14 11 3 10 2 11 4 2 9 3 4 6 1 3 7 10 6 9 14]

7.4 Summary

In this chapter, a novel immune algorithm emulating the features of a biological immune system is proposed for solving the job-shop scheduling problem. The antibody/solution representation of a scheduling is based on the operation (operation-based representation), and the goal is to minimize the maximum makespan time of a scheduling. During the optimal search of scheduling, inherent local search ability in immune system offered by clone process enhances the search speed and accuracy in large-size scheduling problem. In addition, by integrating the features of biological immune system such as antibody memory, fragmental rearrangement, and diversity, the proposed immune algorithm provides a balance between exploring search space and finding optimum solutions. Finally, numerical simulation based on the benchmark instances demonstrated the effectiveness of the proposed immune algorithm which produces optimal or near-optimal solutions on all instances tested, and has better performance to part of comparison methods.

CHAPTER 8

CONCLUSIONS

In this dissertation, a novel immune algorithm emulating the biological immune system fully has been proposed by the author for solving the single-objective, single-objective with multi-modally, and multi-objective optimization problems considering different solution encoding system *e.g.* one- & two-dimensional binary-encoded string and integer-encoded string. The proposed algorithm differs from the other hybrid algorithms which are combined immune algorithm with evolutionary algorithm (especially genetic algorithm) not only used the characteristics of the clonal selection principle and the immune diversity, but the concepts of the cytokine, the germ-line DNA libraries, the antibody fragment rearrangement, the antibody memory, and the more antibody diversity mechanisms are also employed for finding the non-dominated solutions and maintaining diversity in obtained non-dominated front, these are two remarkable things concerned when adopted the evolutionary algorithm in the optimization search. Moreover, the proposed methodology enhances convergent accuracy in solutions via the function of clonal proliferation and schemata recombination implemented through the process of gene fragmental rearrangement. The key natural selection components (gene fragments) are similar to the building blocks of genetic

algorithms associated with stimulus donor antibodies and memory antibodies. The rearrangement of antibody genes involved in the production of antibodies differs somewhat from the recombination of parental genes in genetic algorithms. In the former, antibodies (solutions) are the direct products of gene fragment (*e.g.* light- and heavy-chain gene fragments) combinations (schemata), rather than the antibody itself, while the latter involves the crossing-over (or chromosome mixing) from parental genetic material to create an offspring. Meanwhile, the two main drawbacks of the genetic algorithms – the lack of local search ability and the premature convergence pointed out by Tazawa (1996), have also been improved in this dissertation through the use of clonal proliferation and antibody diversification schemes. The inherent local search and memory abilities of the biological immune system employing clonal proliferation enhance the search speed and convergence accuracy of solutions in the proposed algorithm, with the substitution of increasing computation time. In the other hand, the innate capabilities of specificity, distinction, and diversity using affinity, cytokine, and diversification mechanisms further improve the premature convergence and diversity of solutions. Therefore, the balance between exploration and exploitation of non-dominated solutions within a search space are realized in this dissertation through the integration of clonal proliferation, germ-line gene libraries, cytokine, gene fragment rearrangement, and memory antibodies, further assisted by several diversification schemes.

The effectiveness and adaptability of proposed immune algorithm have been proved by several optimization problems including multi-objective optimizations in the unconstrained/constrained test functions and the sizing of truss structure, single-objective with multi-modal optimizations in the structural topology, and single-objective optimizations in the job-shop scheduling problems. In the multi-objective optimization of unconstrained/constrained test functions, numerous test functions were performed to determine the effectiveness (accuracy as well as spread of global non-dominated solutions or Pareto-optimal solutions) of the proposed immune algorithm, with Pareto-optimal solution performances quantitatively measured by five performance metrics. The compared results of these tests shown that the proposed immune algorithm generally performs better than SPEA and NSGA-II, and by extension also better than MOGA, NPGA, and NSGA in several areas. For multi-objective tress-structure sizing optimization considering the constraint of maximum allowable stress, the compared figures shown that the proposed algorithm is capable of finding acceptable feasible Pareto-optimal solutions in 10-bar plane truss and 25-bar space truss optimization problems. In addition, during the single-objective multi- modal topological optimization considering the asymmetry structures and the constraint of maximum allowable stress, the resulting figures indicated that the potential of the proposed immune algorithm as a tool for investigating optimal topologies and for automatically creating innovative solutions to

structural design problems has been illustrated in the examples presented. In the single-objective job-shop scheduling optimization, 27 benchmark instances were used for demonstrating the optimal search ability in such not-bit encoded system. The scheduling results show that the proposed immune algorithm has ability to produce optimal or near-optimal solutions on all instances tested, and has better performance than simple methods.

Finally, numerous compared results from various applications confirmed that the immune algorithm proposed in this dissertation is capable of fining acceptable Pareto-optimal solutions quickly meanwhile maintaining diversity among Pareto-optimal front and can be applied well in widely field of engineering optimal design.

Bibliography

Aisu, H. and **Mizutani, H.** (1996), "A Rule Acquisition for Image Processing using Immune Mechanism", *Proceeding of the 12th Fuzzy System Symposium*, pp. 75-78.

Allaire, G. and **Kohn, R.V.** (1993), "Optimal design for minimum weight and compliance in plane stress using external microstructures", *European Journal of Mechanics*, A/Solid **12** (6), pp. 839-878.

Anagnostou, G., **Rønquist, E.**, and **Patera, A.** (1992), "A computational procedure for part design", *Computer Methods in Applied Mechanics Engineering*, **97**, pp. 33-48.

Applegate, D. and **Cook, W.** (1991), "A computational study of the job shop scheduling problem", *ORSA Journal on computing*, **3**(2), pp. 149-152.

Balicki, J., **Kitowski, Z.**, and **abd Stateczny, A.** (1998), "Extended Hopfield models of neural networks for combinatorial multi-objective optimization problems", *The IEEE International Joint Conference on Neural Networks, 1998. IEEE World Congress on Computational Intelligence*, pp. 1646 –1651.

Bendsøe, M. P. and **Kikuchi, N.** (1988), "Generating optimal topologies in structural design using a homogenization method", *Computer Methods in Applied Mechanics Engineering*, **71**, pp. 197-224.

Bersini, H. and **Varela, F. J.** (1990), "Hints for adaptive problem solving gleaned

from immune network", *Parallel Problem Solving from Nature*, pp. 343-354.

Bersini, H. and **Varela, F. J.** (1991), "The immune recruitment mechanism: A selective evolutionary strategy", *Proc. 4th Int. Conf. On Genetic Algorithms*, pp. 520-526.

Bersini, H. and **Varela, F. J.** (1994), "The Immune Learning Mechanisms: Reinforcement, Recruitment and Their Applications", In Paton, R. (ed.), *Computing with Biological Metaphors*, Chapman and Hall, pp. 166-192.

Blazewicz, J., **Domschke, W.**, and **Pesch, E.** (1996), "The job shop scheduling problem: Conventional and new solution techniques," *European Journal of Operational Research*, **93**, pp. 1-33.

Brucker, P., **Jurish, B.**, and **Sievers, B.** (1994), "A branch & bound algorithm for the job shop scheduling problem", *Discrete Applied Mathematics*, **49**, pp. 07-127.

Carlier, J. and **Pinson, E.** (1989), "A algorithm for solving the job-shop problem", *Management Science*, 35(2), pp. 164-176.

Carter, J. H. (2000), "The Immune System as a Model for Pattern Recognition and Classification", *Journal of the American Medical Information Association*, **7**(1), pp. 28-41.

Chapman, C., **Saitou, K.**, and **Jakiela, M.** (1994), "Genetic algorithms as an approach to configuration and topology design", *ASME Journal of Mechanical design*,

116, pp. 1005-1012.

Cheng, R., **Gen, M.**, and **Tsujimura, Y.** (1996), "A tutorial survey of job-shop scheduling problems using genetic algorithm – I. Representation", *Computers and Industrial Engineering*, **30**(4), pp. 983-997.

Cheng, R., **Gen, M.**, and **Tsujimura, Y.** (1999), "A tutorial survey of job-shop scheduling problems using genetic algorithm – II. Hybrid genetic search strategies", *Computers and Industrial Engineering*, **30**(4), pp. 51-55.

Chun J. S., **Lim, J. P.**, **Jung, H. K.**, and **Yoon, J. S.** (1999a), " Optimal design of synchronous motor with parameter correction using immune algorithm", *IEEE Trans. On Energy Conversion*, **14**(3), pp. 610-615.

Chun, J. S., **Lim, J. P.**, **Jung, H. K.**, and **Jung, H. K.** (1999b), "Multisolution optimization of permanent magnet linear synchronous motor for high thrust and acceleration operation", *Electric Machines and Drives*, 1999. International Conference IEMD '99, pp. 57-59.

Chun, J. S., **Jung, H. K.**, and **Hahn, S. Y.** (1998), "A study of optimization performances between immune algorithm and other heuristic algorithms", *IEEE Trans. on Magnetics*, **34**(5), pp. 2972-2975.

Chun, J. S., **Kim, M. K.**, **Jung, H. K.**, and **Hong, S. K.** (1997), "Shape optimization of electromagnetic device using immune algorithm", *IEEE Trans. on Magnetics*, **33**(2),

pp. 1876-1879.

Coello Coello, C. A., **Van Veldhuizen, D. A.**, and **Lamont G. B.** (2002), "Evolutionary algorithms for solving multi-objective problems", New York: Kluwer Academic Publishers.

Coello Coello, C. A. and **Christiansen, A. D.** (2000), "Multiobjetive optimization of trusses using genetic algorithms", Computers and Structures, **75**(6), pp. 647-660.

Coren, D., **Dorigo, M.**, and **Glover, F.** (eds.), "New idea in Optimization, Part Three: Immune System Methods", McGraw Hill, New York, pp. 203-215.

Croce, F. D., **Tadei, R.**, and **Volta, G.** (1995), "A genetic algorithm for job shop scheduling problem", *Computers and Operations Research*, **22**, pp. 15-24.

Dasgupta, D. and **Forrest, S.** (1999), "An Anomaly Detection Algorithm Inspired by the Immune System", In Dasgupta, D. (ed.), *Artificial Immune System and Their Applications*, Springer-Verlag, pp. 262-277.

Davis, J. (1985), "Job shop scheduling with genetic algorithm", *proceeding of First international Conference on Genetic Algorithm*, pp. 136-140.

Deb, K. (1999), "Multi-objective genetic algorithms: Problem difficulties and construction of test problems. *Evolutionary Computation*, **7**(3), pp. 205-230.

Deb, K. (2001), "Multi-objective optimization using evolutionary algorithms", New York: JOHN WILEY & SONS.

Deb, K. and **Gulati, S.** (2001), "Design of truss-structures for minimum weight using genetic algorithms", *Finite Elements in Analysis and Design*, **37**, pp. 447-465.

Deb, K., **Agrawal, S.**, **Pratap, A.**, and **Meyarivan, T.** (2000), "*A fast and elitist multi-objective genetic algorithm: NSGA-II*", Technical Report 200001, Indian Institute of Technology, Kanpur: Kanpur Genetic algorithms Laboratory (KanGAL).

Deb, K., **Pratap, A.**, and **Meyarivan, T.** (2001), "Constrained test problems for multi-objective evolutionary optimization", *First International Conference*, EMO, pp. 284-298.

de Castro, L. N. and **Jonathan, T.** (1999), "Artificial immune systems: A new Computational Intelligence Approach", Springer-Verlag.

Dell' Amico, M. and **Trubian, M.** (1993), "Applying tabu search to the job shop scheduling problem", *Ann. Ops. Res.*, **40**, pp. 231-252.

Dorndorf, U. and Pesch, E. (1995), "Evolution based learning in a job shop environment", *Computers and Operations Research*, **22**, pp. 25-40.

Endoh, S., **Toma, N.**, and **Yamada, K.** (1998), "Immune Algorithm for n-TSP", *Proceeding of IEEE Systems, Man, and Cybernetics Conference*, pp. 3844-3849.

Fourie, P. C. and **Groenwold, A. A.** (2002), "The particle swarm optimization algorithm in size and shape optimization", *Structural Multidiscipline Optimization*, **23**, pp.259-267.

Erbatur, F., **Ohasancebi, O.**, **Tütüncü, I.**, and **Kilic, H.** (2000), "Optimal design of planar and space structures with genetic algorithms", *Computers and Structures*, **75**, pp. 209-224.

Fadel, G. and **Li, Y.** (2002), "Approximating the Pareto curve to help solve biobjective design problems", *Struct Multidisc Optim*, **23**, pp. 280-296.

Fisher, H. and **Thompson, G. L.** (1963), "Probabilistic learning combinations of local job-shop scheduling rules", In: Industrial Scheduling, Muth, J. F. and Thompson, G. L. (eds.), Prentice-Hall, Englewood Cliffs, NJ, pp. 225-251.

Fonseca, C. M. and **Fleming, P. J.** (1993), "Genetic algorithm for multiobjective optimization: Formulation, discussion, and generalization", *In Proc. of the Fifth Int. Conf. on Genetic Algorithms*, pp. 416-423.

Foo, S. Y., **Takefuji, Y.**, and **Szu, H.** (1995), "Scaling properties of neural networks for job shop scheduling", *Neurocomputing*, **8**(1), pp. 79-91.

Fukuda, T., **Mori, M.**, and **Tsukiyama M.** (1993), "Immune Network Genetic Algorithm for Adaptive Production Scheduling", *Proceeding of 15th IFAC World Congress*, **3**, pp.57-60.

Fukuda, T., **Mori, K.**, and **Tsukiyama, M.** (1998), "Parallel search for multi-modal function optimization with diversity and learning of immune algorithm", In Dasgupta, D. (ed,), *Artificial Immune Systems and Their Applications*, pp. 211-219.

Gandibleux, X., **Mezdaoui, N.**, and **Freville, A.** (1996), "A Tabu Search Procedure to Solve Multiobjective Combinatorial Optimization Problems", In Caballero, R. and Steuer, R. (eds.), *Proceedings Volume of Multiple Objective Programming and Goal Programming '96*, Springer-Verlag.

Gen, M., **Tsujimura, Y.**, and **Kubota, E.** (1994), "Solving job-shop scheduling problem using genetic algorithms", *Proc. of the 16th Int. Conf. on Computer and Industrial Engineering*, Ashikaga, Japan, pp. 576-579.

Goldberg, D. E. (1989), "Genetic algorithms in search optimization & learning", Addison-Wesley.

Goldberg, D. and **Richardson, J.** (1987), "Genetic algorithms with sharing for multimodal function optimization, *Proceedings of the Second International Conference on Genetic algorithms*, pp. 41-49.

Goldberg, D. and **Samtani, M.** (1986), "Engineering optimization via genetic algorithm", In: Electronic Computation – Proceedings of the Ninth Conference on Electronic Computation, American Society of Civil Engineers, Alabama at Birmingham, February, pp. 471-482.

Gonçalves, J. F., **Mendes, J. M.**, and **Resende, M. G. C.** (2002), "A hybrid genetic algorithm for the job shop scheduling problem", AT&T Labs Research Technical Report TD-5EAL6J.

Hajela, P. and **Lee, J.** (1996), "Constrained Genetic Search via Schema Adaptation: An Immune Network Solution", *Structural Optimization*, **12**(1), pp. 11-15.

Hajela, P. and **Lin, C.-Y.** (1992), "Genetic search strategies in multicriterion optimal design", *Structural Optimization*, **4**, pp.99-107.

Hajela, P., **Yoo, J.**, and **Lee, J.** (1997), "GA based Simulation of Immune Networks – Application in Structural Optimization", *Engineering Optimization*, **29**, pp. 131-149.

Hajela, P. and **Yoo, J.** (1999), "Immune Network Modeling in Design Optimization", In

Hansen, M. P. (1997), "Tabu Search for Multiobjective Optimizataion: MOTS", *MCDM'97*, Cape Town, South Africa, January, pp. 6-10.

Harris, R.S., **Kong, Q.**, and **Maizels, N.** (1999), "Somatic hypermutation and the three R's: repair, replication and recombination", *Mutation Research*. **436**, pp. 157-178.

Hasanc, Àebi O. and **Erbatur F.** (2001), "Evaluation of crossover techniques in genetic algorithm based optimum structural design", *Computers and Structures*, **78**, pp. 435-448.

Horn, J., **Nafpliotis, N.**, and **Goldberg, D. E.** (1994), "A niched pareto genetic algorithm for multi-objective optimization", *Proceedings of the First IEEE Conference on Evolutionary Computation*, pp. 82-87.

Hunt, J. E. and **Cooke, D. E.** (1996), "Learning using an Artificial Immune System", *Journal of Network and Computer Applications*, **19**, pp. 189-212.

Ishida, R., **Sato, T.** and **Sugiyama, Y.** (1995), "Optimum design of truss structure by genetic immune recruitment mechanism", *Japan Society of Mechanical Engineers*, **61**(581), pp. 205-210.

Jakiela, M. J., **Chapman, C.**, **Duda, J.**, **Adewuya, A.**, and **Saitou, K.** (2000), "Continuum structural topology design with genetic algorithms", *Computer Methods in Applied Mechanics Engineering*, **186**, pp. 339-356.

Jenkins, W. (1991), "Structural optimization with the genetic algorithm", *Structural engineering*, **69**, pp. 418-422.

Jerne, N. K. (1974), "Towards a network theory of immune system", *Ann. Immunol.*, **125**C, pp. 373-389.

Jiao L. and **Wang L.** (2000), "A novel genetic algorithm based on immunity", *IEEE Transactions on Systems, Man, and Cybernetics – Part A: Systems and Humans*, **30**(5), pp. 552-561.

Jun, J.-H., **Lee, D.-W.**, and **Sim, K.-B.** (1999), "Realization of Cooperative and Swarm Behavior in Distributed Autonomous Robotics Systems using Artificial Immune System", *Proceeding of IEEE Systems, Man, and Cybernetics Conference*, **4**, pp. 614-619.

Kane, C. and **Schoenauer, M.** (1996), "Topological optimum using genetic algorithm", *Control and Cybernetics*, **25**, pp. 1059-1088.

Kephart, J. O. (1994), "A Biological Inspired Immune System for Computers", In Brooks, R. A. and Maes, P. (eds.), *Artificial Life IV Proceeding of the Fourth International Workshop on the Synthesis and Simulation of Living Systems*, MIT press, pp. 130-139.

Kibsgaard, S. (1992), "Sensitivity analysis - the basis for optimization", *International Journal of Numerical Methods in Engineering*, **34**, pp. 901-932.

Kim, J. and **Bentley, P.** (1999), "Negative Selection and Niching by an Artificial Immune System for Network Instrusion Detection", *Proceeding of Genetic and Evolutionary Computation Conference*, pp. 149-158.

Knight, T. and **Timmis, J.** (2001), "AINE: An Immunological Approach to Data Mining", *Proceeding of the IEEE International Conference on Data Mining*, pp.297-304.

Knowles, J. D. and **Corne, D. W.** (1999), "Approximating the Nondominated Front Using the Pareto Archived Evolution Strategy", *Evolutionary Computation*, **7**(3), pp. 1-26

Kolonko, M. (1999), "Some new results on simulated annealing applied to the job shop scheduling problem", *European Journal of Operational Research*, **113**, pp.

123-136.

Koppen, M. and **Rudlof, S.** (1997), "Multi-objective optimization by NESSY algorithm", *Proceedings of the Second International ICSC Symposium on Soft Computing*, SOCO'97, pp. 243-248.

Krawinkel, U., **Zoebelein, G.**, **Bruggemann, M.**, **Radburch, A.**, and **Rajewsky, K.** (1983), "Recombination between antibody heavy chain variable-region genes: Evidence for gene conversion", *Proc. Natl. Acad. Sci. USA*, **80**, pp. 4997-5001.

Krishnakumar, K. and **Neidhoefer, J.** (1997), "Immunized Adaptive Critics for Level 2 Intelligent Control", *Proceeding of IEEE Systems, Man, and Cybernetics Conference*, **1**, pp. 856-860.

Laumanns, M., **Rudolph, G.**, and **Schwefel, H. P.** (1998), "A spatial predator-prey approach to multi-objective optimization: A preliminary study", *Proceedings of the Parallel Problem Solving from Nature V (PPSN-V)*, pp. 241-249

Lawrence, S. (1984), "Resource constrained project scheduling: An experimental investigation of heuristic scheduling techniques", GSIA, Carnegie Mellon University, Pittsburgh, PA.

Lin, C. Y. and **Chou, J. N.** (1999), "A two-stage approach for structural topology optimization", *Advances in Engineering Software*, **30**, pp. 261-271.

Lis, J. and **Eiben, A. E.**(1997), "Multi-sexual genetic algorithm for multiobjective

optimization", *Proceedings of the IEEE Conference on Evolutionary Computation*, Indianapolis, IN, USA, pp. 59-64

Liu, J. S., **Parks, G. T.**, and **Clarkson, P. J.** (2000), "Metamorphic development: a new topology optimization method for continuum structures", *Struct. Multidisc. Optim.*, **20**, pp. 288-300.

Lo, C.-C. and **Chang, W.-H.** (2000), "Multiobjective hybrid genetic algorithm for the capacitated multipoint network design problem", *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, **30**(3), pp. 461-470

Loh, H. T. and **Papalambros, P. Y.** (1991), "A sequential linearization approach for solving mixed-discrete nonlinear design optimization problems", *ASME Transactions on Journal of Mechanical Design*, **113**(3), pp. 325-334.

Luh, G.-C. and **Cheng, W.-C.** (2001), "Non-linear System Identification using an Artificial Immune System", *Proc. Instn. Mech. Engrs, Part I, Journal of Systems and Control Engineering*, **215**, pp. 569-585.

Luh, G.-C. and **Cheng, W.-C.** (2002), "Behavior-based Intelligent Mobile Robot using an Immunized reinforcement adaptive learning Mechanism", *Advanced Engineering Informatics*, **16**(2), pp. 85-98.

Luh, G.-C., **Chueh, C.-H.**, and **Liu, W.-W.** (2003), "MOIA: Multi-objective Immune Algorithm", *Engineering Optimization*, **35**(2), pp. 143-164.

Luh, G.-C. and **Chueh, C.-H.** (2004), "Multi-objective Optimal Design of Truss Structure with Immune Algorithm", *Computers and Structures*, **82**, pp. 829-844.

Manser, T., **Wysocki, L. J.**, **Margolies, M. N.**, and **Gefter, M.L.** (1987), "Evolution of Antibody Variable Region Structure during the Immune Response", *Immunological Reviews* **96**, pp. 141-162.

Mariano C. E. and **Morales, E.** (1999), "A Multiple Objective Ant-Q Algorithm for the Design of Water Distribution Irrigation Networks", *Technical Report HC-9904*, Instituto Mexicano de Tecnología del Agua.

Maturana, F., **Gu, F.**, **Naumann, A.**, and **Norrie, D. H.** (1997), "Object-oriented job-shop scheduling using genetic algorithms", *Computers in Industry*, **32**, pp. 281-294.

Michalewicz, Z., **Dasgupta, D.**, and **Le Riche, R. G.** (1996), "Schoenauer M. Evolutionary algorithms for constrained engineering problems", *Computers Industrial Engineering*, *30*(4), pp. 851-870.

Moh, J. S. and **Chiang, D. Y.** (2000), "Improved simulated annealing search for structural optimization", *AIAA Journal*, **38**(10), pp. 1965-1973.

Mori, M., **Tsukiyama, M.**, and **Fukuda, T.** (1993), "Immune Algorithm with Searching Diversity and its application to Allocation Problem", *Trans. of the institute of Electrical Engineering of Japan*, **113-C**(10), pp.872-878.

Murata, T. and **Ishibuchi, H.** (1995), "MOGA: Multi-objective genetic algorithms", *Proceedings of the Second IEEE International Conference on Evolutionary Computation*, pp. 289-294.

Murata, T. and **Ishibuchi, H.**, and **Tanaka, H.** (1995), "Multi-objective genetic algorithm and its applications to flowshop scheduling", *Computers industrial Engineering*, **30**(4), pp. 957-968.

Narayanan, S. (1998), "On improving multiobjective genetic algorithms for design optimization", Structural Optimization, **18**(2-3), pp. 146-155.

Nowicki, E. and **Smutnicki, C.** (1996), "A fast taboo search algorithm for the job-shop problem", *Management Science*, **42**(6), pp. 797-813.

Osyczka, A. (2002), "Evolutionary algorithms for single and multicriteria Design optimization", Germany: Physica Verlag.

Osyczka, A. and **Kundu, S.** (1995), "A new method to solve generalized multicriteria optimization problems using the simple genetic algorithm", *Structural Optimization*, **10**(2), pp. 94-99.

Perelson, A. S., **Mirmirani, M.**, and **Oster, G.F.** (1978), "Optimal strategies in immunology II. B Memory cell Production", *J. Math. Biol.*, **3**, pp. 325-367.

Ponnambalam, S. G., **Aravindan, P.**, and **Rajesh, S. V.**, (2000), "A tabu search algorithm for jod shop scheduling", *International Journal of Advanced Manufacturing*

Technology, **16**, pp. 765-771.

Ponterosso, P. and **Fox, D. S. J.** (1999), "Heuristically Seeded Genetic Algorithms Applied to Truss Optimization", *Engineering with Computers*, **15**, pp. 345–355.

Querin, O. M., **Steven, G. P.**, and **Xie, Y. M.** (1998), "Evolutionary structural optimization (ESO) using a bidirectional algorithm", *Engineering Computations*, **15**, pp. 1031-1048.

Rajeev, S. and **Krishnamoorthy, C. S.** (1992), "Discrete optimization of structures using genetic algorithms", *Structural engineering*. **118**, pp. 1233-1250.

Rao, S. S., **Sundararaju, K.**, **Prakash, B. G.**, and **Balakrishna, C.** (1992), "Multi-objective fuzzy optimization techniques for engineering design", *Computers & Structures*, **42**(1), pp. 37-44.

Rao Vemuri, V. and **Cedeno, W.** (1995), "New genetic algorithm for multi-objective optimization in water resource management", *IEEE International Conference on Evolutionary Computation*, pp. 495-500.

Roitt, I. and **Brostoff, J.** (1998), "Immunology 5/e", Mosby International Ltd., 1998.

Schaffer, J. D. (1985), "Multiple objective optimization with vector evaluated genetic algorithms", *Proceedings of an international Conference on Genetic Algorithms and Their Applications*, Pittsburgh, PA, pp. 93-100.

Schott, J. R. (1995), "*Fault Tolerant Design using Single and Multi-criteria Genetic*

Algorithms", Master's Thesis, Boston, MA: Department of Aeronautics and Astronautics, Massachusetts Institute of Technology

Shih, C. J. and **Yu, K. C.** (1995), "Weighting objectives strategy in multicriterion fuzzy mechanical and structural optimization", Structural Engineering and Mechanics, 394, pp. 373-382.

Srinivas, N. and **Deb, K.** (1994), "Multi-objective optimization using non-dominated sorting in genetic algorithms", *Evolutionary Computation*, **2**(3), pp. 221-248.

Suppapitnarm, A., **Seffen K. A.**, **Parks G. T.**, and **Clarnkson P. J.** (2000), "Simulated annealing algorithm for multiobjective optimization", *Engineering Optimization*, **33**(1), pp. 59-85.

Suzuki, K. and **Kikuchim, N.** (1991), "A homogenization method for shape and topology optimization", *Computer Methods in Applied Mechanics Engineering*, **93**, pp. 291-318.

Tazawa, I., **Koakutsu, S.**, and **Hirata, H.** (1996), "An immunity based genetic algorithm and its application to the VLSI floorplan design problem", *Proceedings of IEEE International Conference on Evolutionary Computation*, pp. 417-421.

Templeman, A. B. (1988), "Discrete optimum structural design", *Computers and Structures*, 30(3), pp. 511-518.

Tenek, L. H. and **Hagiwara, I.** (1993), "Static and vibrational shape and topology

optimization using homogenization and mathematical programming", **Computer Methods in Applied Mechanics Engineering**, **109**, pp. 143-154.

Toma, N., **Endo, S.**, and **Yamada, K.** (1999), "Immune algorithm with immune network and MHC for adaptive problem solving", *Proc. of the IEEE System, Man, and Cybernetics*, **IV**, pp. 271-276.

Tomoyuki, M. (2003), "An application of Immune Algorithms for Job-Shop Scheduling Problems", *Proc. of the 5th International Symposium on Assembly and Task Planning*, pp. 146-150.

Van Laarhoven, P. Aarts, E., and Lenstra, J. (1992), "Job shop scheduling by simulated annealing", *Ann. Ops. Res.*, **40**, pp. 113-125.

Viennet, R., **Fonteix, C.** and **Marc, I.** (1996), "New multicriteria optimization method based on the use of a diploid genetic algorithm: Example of an industrial problem", *Lecture Notes in Computer Science*, v1063, pp. 200.

Wang, L. and **Zheng, D. Z.** (2002), "A modified genetic algorithm for job shop scheduling", *Int J Adv Manuf Technol*, **20**, pp. 72-76.

Woon, S. Y., **Querin, O. M.**, and **Steven, G. P.** (2001), "Structural application of a shape optimization method based on a genetic algorithm", *Struct. Multidisc. Optim.*, 22, pp. 57-64.

Wu, S. J. and **Chow, P. T.** (1995a), "Steady-state genetic algorithms for discrete

optimization of trusses", *Computers & structures*, **56**, pp. 979-991.

Wu, S. J. and **Chow, P. T.** (1995b), "Integrated discrete and configuration optimization of trusses using genetic algorithms", *Computers and Structures*, **55**(4), pp. 695-702.

Xie, Y. M. and **Steven, G. P.** (1993). "A simple evolutionary procedure for structural optimization", *Computers & structures*, **49**, pp. 885-896.

Yoo, J. and **Hajela, P.** (1999), "Immune network simulations in multicriterion design", *Structural Optimization*, **18**, pp. 85-94.

Zitzler, E., **Deb, K.**, **Thiele, L.**, **Coello Coello C. A.**, and **Corne D.** (2001), "Evolutionary multi-criterion optimization", Springer.

Zitzler, E., **Deb, K.**, and **Thiele, L.** (2000), "Comparison of Multiobjective Evolutionary Algorithms: Empirical Results", *Evolutionary Computation*, **8**(2), pp. 173-195.

Zitzler, E. (1999), "*Evolutionary Algorithms for multi-objective optimization: Methods and Application*", Ph. D. Thesis, Zurich, Switzerland: Swiss Federal Institute of Technology (ETH) (Dissertation ETH No. 13398).

Zitzler, E. and **Thiele, L.** (1998), "An evolutionary algorithm for multi-objective optimization: The strength Pareto approach", *Technical Report 43*, Computer Engineering and Communication Network Lab (TIK), Swiss Federal Institute of

Technology (EIH) Zurich, Gloriastrasse 35, CH-8092 Zurich, Switzerland.

Zitzler, E. and **Marco, L.**, “Test problems and test data for multiobjective optimizers”, Test Problem Suite, http://www.tik.ee.ethz.ch/~zitzler/testdata.html