Neural Comput & Applic (2012) 21:1465–1475
DOI 10.1007/s00521-012-0852-2

ORIGINAL ARTICLE

Applying a hybrid artificial immune systems to the job shop scheduling problem

Gary Weckman · Akshata A. Bondal · Magda M. Rinder · William A. Young II

Received: 5 August 2009 / Accepted: 17 January 2012 / Published online: 28 January 2012

Abstract In today's economy, manufacturing sectors are challenged by high costs, low revenues. As part of the managerial activities, scheduling plays an important role in optimizing cost, revenue, profit, time, and efficiency by optimization of available resources. The objective of this research is to evaluate the existing artificial immune system (AIS) principles, models, and applications, and to develop an algorithm applicable to job shop scheduling problems. The developed algorithm was based on the theories of the positive selection algorithm and the clonal selection principle. To test the algorithm, ten job shop scheduling problems were evaluated using the new AIS model. To validate the results, the same job scheduling problems were evaluated using a genetic algorithm (GA) model. The results of the two evaluations were compared against each other using the dimensions of optimality and robustness. The testing revealed that the AIS model was slightly less competitive than the GA model in the optimality test but beat the GA in robustness. Another key finding was that the robustness of the model increased as the best solutions produced by the model were closer to the known optimal.

Keywords Artificial immune systems · Genetic algorithms · Job shop · Scheduling

G. Weckman (✉) · A. A. Bondal · M. M. Rinder
Industrial Systems Engineering, College of Engineering, Ohio University, Athens, OH 45701, USA
e-mail: weckmang@ohio.edu

W. A. Young II
Department of Operations Management Systems, Ohio University, Athens, OH 45701, USA

1 Introduction

Scheduling is the process of optimizing the assignment of a set of tasks to a group of finite resources. The final aim of scheduling is, generally, to optimize one particular dimension or more, such as cost, revenue, profit, time, or efficiency. These dynamics imply that tasks, as well as resources, have different constraints, which make scheduling a complex process. Thus, a schedule has to be developed so that an optimum result is obtained by taking limitations and constraints into consideration.

Since scheduling is essentially an optimization-under-constraints problem, it comes under the broad scope of operations research. For each of the different objectives, many traditional and non-traditional techniques and rules have been developed [15]. All of the approaches can be broadly classified under two categories: *exact methods* and *approximation techniques*. The most significant exact method used for the job shop scheduling problem is called the *branch-and-bound method*. This method was developed by Land and Doig [22] for the primary purpose of 'optimization of problems which could be formulated as linear programming problems with additional constraints'. This method first finds a feasible region in which the solution exists and then tries to narrow down the search to find the exact value within that region. Though not guaranteed, exact methods are more likely to determine optimal solutions, but require considerable more time than approximation techniques.

Approximation techniques are also used to determine solutions for non-deterministic polynomial (NP) problems. These techniques do not always reach the optimal solution, but often determine solutions within 5% of the optimal in less time than exact methods. Some of the approximation techniques used on job shop problems include *priority*

dispatch rules, *bottleneck based heuristics*, *opportunistic scheduling*, and *artificial intelligence*.

The *Job Shop Scheduling Problem* (JSSP) is a special type of scheduling problem. The JSSP pertains to the allocation of jobs to machines in a job shop environment. A job shop is a production center where all of the machines possessed by the center are placed in the same area and all jobs entering the center share the same machines. It is possible that a particular job entering the system may not need all of the machines that are in the center, and it is also possible to have more than one of the same type machine. This makes the process of scheduling the most difficult as all jobs that enter the system use most of the machines. Therefore, there is a high probability of jobs waiting for busy machines (or vice versa if improper scheduling procedures are followed), resulting in poor productivity within the production system.

Artificial intelligence (AI) refers to evolutionary algorithms like *genetic algorithms* (GA), *artificial immune systems* (AIS), *artificial neural networks* (ANN), *Tabu search techniques* (TS), and *simulated annealing* (SA), and others are generally based on the working of some natural occurrence. For instance, GAs are based on how chromosomes are built from genes using the process of selection, mutation, and crossover. Computer programs imitate these natural processes and can be applied to different optimization problems.

The JSSP is a good representation, or generalization, of many types of scheduling problems. Even the most simplified version of the JSSP is considered to be NP, making it difficult to find an optimal solution. In some situations, it is unknown whether or not a NP problem even has an optimal solution. GAs, TSs, and ANNs are arguably among the most popular of all AI algorithms and have the advantage of requiring less computation time than traditional scheduling techniques. However, their performance cannot be guaranteed for a particular problem, but are often used in practice due to their high-quality solution at low computation times. The uses of AIS have received little to no attention for the JSSP.

1.1 Objective

Although some of the past approaches have proved to be adept at solving complicated JSSP, shortcomings have been noted. For example, some approaches may be capable of finding optimal solutions, but take a very long computational time, while others reach near-optimal solutions in a shorter computational time frame. Overcoming these shortcomings and at the same time testing a novel algorithm in the field of JSSP made this topic an attractive proposition. The basic approach used in this research was to garner an understanding of the theory behind AIS and devise an algorithm that would not only best suit the needs of the scheduling problem but also introduce unique features as compared to methods already applied. The approach can be summarized as follows:

- Understanding the underlying principles of AISs, analyzing their potential to be applied to JSSPs, and developing a suitable algorithm;
- Testing the performance of this algorithm on a set of JSSPs and evaluating its performance by comparing it to that of a GA. The GA was used as a good basis for comparison since it is routinely used to generate optimal solutions to JSSP problems.

2 Job shop scheduling problem

The JSSP is a special type of scheduling problem where it pertains to the allocation of jobs to machines in a job shop environment. A *Job Shop* is a production center where all of the machines possessed by the center are collocated and all jobs entering the center sequentially compete for the machines. It is possible that a particular job entering the system may not need all of the machines in the center, and it is also possible to have more than one of the same type machine. The example in Fig. 1 gives an idea of what a Job Shop looks like and how jobs entering the system move between machines in their respective processing orders for specified processing time periods. This example considers a 6 Machine 3 Job problem. The jobs enter the system, and the illustration describes the operation sequence flow for each job. Job 1, for instance, starts with Machine 1, goes on to Machine 4 then Machine 5, Machine 2, Machine 3, and finally Machine 6. Similarly, the sequence of the other jobs is also illustrated. The complications involved in any job shop problem would address the following issues:

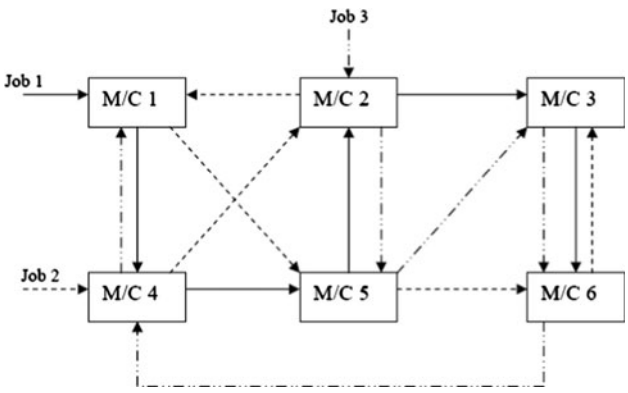

Fig. 1 An example of a job shop

- Reducing conflicts—assigning two jobs to the same machine at the same time
- In case of conflicts, determining which job gets the machine first and which job waits
- Final aim, in our case, is reducing time required to process all of the jobs in the system.

In general, job shops can have any number of machines, machine types, and number of jobs entering the system. Also, as mentioned earlier, each job may or may not utilize all of the machines in the system producing of the product. However, the type of job shop problem considered in this paper is a particular type of the general problem that has a few unique characteristics:

- All of the jobs have to be processed using all of the machines. Hence, if there are 10 machines and 4 jobs, each of the jobs will be processed on all 10 machines.
- Although each job has to be processed on all of the machines, the order in which each job is routed on the machines may vary according to the processing specifications for each job. For example, Job 1 may go to Machine 1 first and then continue to Machine 2, but Job 2 may go to Machine 2 first and then go to Machine 1.
- There are no constraints on the processing times of each job on each machine.
- All jobs arrive at the same time period ($t = 0$).
- There is no consideration for any job's due date. All of the jobs are simply required to complete all of their assigned operations.
- The main objective is to minimize makespan, which is defined as the time difference between the start and finish of a sequence of jobs. Thus, the aim of developing the schedule is to organize jobs on machines so as to minimize the total time that all of the jobs take in the job shop system.
- The only constraint in this problem is that of the precedence order of the operations of each job. A job has to follow the order of operations assigned to it and cannot violate that one constraint. For example, consider an example of a 6×6 Job Shop Scheduling problem. A 6×6 problem refers to a problem of 6 jobs being scheduled on 6 machines where each job needs to be machined on each of the 6 machines only once for the job to transform raw material to the final product. The sequence in which the job is scheduled on the 6 machines and its operation time on that machine is also needed. This information is presented in Table 1 for a common problem called the FT06.

The reason a Job Shop Problem was chosen to analyze this problem is that it is a good representation of all scheduling problems in general. Also, the problem is considered to be an NP problem, making it highly difficult to be solved optimally.

Table 1 The routing and machining times of the FT06 problem

Jobs	Operations					
	1	2	3	4	5	6
1	3.1	1.3	2.6	4.7	6.3	5.6
2	2.8	3.5	5.10	6.10	1.10	4.4
3	3.5	4.4	6.8	1.9	2.1	5.7
4	2.5	1.5	3.5	4.3	5.8	6.9
5	3.9	2.3	5.5	6.4	1.3	4.1
6	2.3	4.3	6.9	1.10	5.4	3.1

3 Artificial immune systems

A basic understanding of the human immune system is essential to understand the AIS model. The human immune system is characterized by its adaptive and robust nature. This can be illustrated by considering a simple example of an infection attacking the body. The infection (or *antigen*) attacking the body is countered by the defense mechanism called the *antibody*. The antibody consists of a varied combination of *T-cells* and *B-cells*, which can adapt themselves to counter the antigen.

For example, if an antigen has a particular configuration, the T-cells and B-cells will attack the antigen with different configurations until they determine the one match that is capable of destroying the antigen. Further, B-cells have the property of cloning the configuration that kills the antigen so that a healthy population of antibody will be created, which destroys the infection.

Theoretical immunology has been adapted by others into an artificial system by developing a number of problem-solving algorithms such as the *Positive Selection*, *Negative Selection*, and the *Clonal Selection Algorithms*. Much work continues in seeking useful aspects of the human immune system and developing them into artificial immune algorithms. The areas of practical applications of AIS have been used in diverse areas such as computer security, data mining, machine learning, and scheduling.

3.1 AIS application

AISs originated with the work by Farmer et al., and Hoffman. Farmer et al. [10] proposed a dynamical model based on the theory of the immune network, showing how immune system can be applied to the field of artificial intelligence. The proposed model became very popular and became the basis for applying it to different applications.

Hoffman [19] proposed a comparison of the nervous and immune system and built a model incorporating some aspects of the immune system into a neural network model. However, the scope of this model in terms of applications was limited and not as useful as that proposed by Farmer et al. [10]. However, the unique idea of combining the two promoted more research into the area, and numerous models have since been generated. Ishida [20, 21] developed models of the immune system and applying these models to areas like process diagnosis. His work deals with a model built on the recognition capabilities of the immune system and how it learns through recognition. Further, this model was used to diagnosing and eliminating computer viruses. His research focused on the fields of process diagnosis, robotics, and computer intrusion detection. Additional application-oriented research was presented by Bersini and Varela [3, 4] and Bersini [5] where they developed new models on the immune system to address machine learning, optimization, and adaptive control problems.

Research attempts were made to use GAs for pattern recognition in binary immune system models. This research was reported in Forrest and Perelson [12] and Forrest et al. [13] and mainly applied to protection of computers from viruses and network intrusions. These models became successful enough for computer companies like IBM to develop their own immune-based anti-virus software. This is still the most popular applications of AIS research.

After 1995, AISs were applied to diverse areas such as pattern recognition, anomaly detection, optimization, and scheduling. In one attempt to give a general overview of artificial immune systems, Dasgupta [7] states that the immune system's qualities of learning, memory, and adaptation make it ideal in modeling recognition and classification tasks. Timmis [27] developed a data analysis system based on the principles of artificial immune systems where he used an antibody network to represent the diverse nature of a data set and then used cloning and mutation. De Castro and Von Zuben [8, 9] proposed an artificial immune network named AiNet for data analysis, especially clustering and filtering of unlabeled numerical data sets. This method reduced data redundancy and successfully described data structures in terms of their spatial distribution and cluster interrelations. AiNet is used for automating knowledge discovery, mining of redundant data and automatic data clustering.

Nasaroui et al. [24] addressed the shortcomings of previous methods by developing fuzzy algorithms to identify patterns in observed data to make predictions in unseen data. The main area of improvement was in the matching of antibodies to antigens. This fuzzy AIS helps websites process requests. The experimentation done revealed that the system performs more efficiently using the new algorithm. Neal [25] also developed an artificial immune system used for data clustering. The model is stable, adaptive, and dynamic and can handle a very large number of data presentations. The approach uses artificial recognition balls and is self-stabilizing. The algorithm was tested on two data sets. First, a 2D data set consisting of 50 dimensions each was tested for clustering. Next, it was tested on the well-known Fisher Iris data set for ease comparison to other algorithms. The long-term stability of the model was comprehensively demonstrates by analyzing over 1,000,000 data items. The results obtained from this comprehensive experimentation revealed that the model was effectively applied to the analysis of continuously changing data sets without hindering its learning process.

3.2 Scheduling

The first attempt at solving a scheduling problem using an AIS was by Fukuda et al. [14] where an expression in terms of *multi-agent nets* was developed. The immune agents follow a three-step procedure starting with a sensing mode, progressing to a decision mode and ending with a controlling mode. The authors employed this procedure to dispatch material with the aim of minimizing work in progress (WIP) as well as the product cycle time. The framework of the AIS is the multi-agents, which are comprised of detector agents, mediator agents, inhibitor agents, and restoration.

Hart et al. [18] also studied the JSSP using AISs, where they produced robust schedules for dynamic environment. With the aim of minimizing maximum lateness, the authors used their system to develop antigen, antibody, and *gene libraries*. The lateness of a job is defined as its completion time minus its due date. Hart and Ross [17] developed an algorithm capable of scheduling jobs using historical environments. Their algorithm was tested extensively with 10 test scenarios generated from a base problem of 15 jobs. The results demonstrated that their method of recreating a schedule from previous patterns is effective. Hart [16] applied AIS theories to the areas of job shop scheduling and data clustering. She justified the suitability of applying AIS to these areas by arguing that the problem characteristics are very similar to those of the human immune system. In total, she developed four models, two for each area of application and tested them on relevant benchmark artificial data sets, showing the relevance and utility of the models.

In 2004, Aickelin et al. [2] stated that schedules developed using AIS are more robust than those obtained by GAs. Previous results show that increasing the number of antigens in AISs improves the optimality of schedules obtained, but also decreases fitness. The fitness function is

defined as a particular type of objective function that defines the optimality of a solution. Thus, it is suggested that rescheduling the same problem is preferable to increasing the number of antigen. As suggested by previous research, this method exposes the schedules to more antigens, while at the same time, not adversely affecting fitness. Consequently, the system becomes more robust.

Coello et al. [6] combined the Clonal Selection Algorithm with *hypermutation principles* into a unique mechanism in order to solve the JSSP. The features of their method include exploring the vicinity of the reference solution, a search technique that eliminates gaps in the schedule, and incorporating this into the decoding strategy. The authors take up a 6×4 (i.e. 6 jobs and 4 machines) JSSP that has previously been solved. Gaps are then sought in the current schedule with the aim of reducing *makespan* (the time difference between the start and finish of a sequence of jobs) and increasing the optimality of the solution.

4 Hybrid AIS methodology

The algorithm developed in this research makes use of two theories of theoretical immunology, namely, the clonal selection algorithm and the positive selection principle. Based primarily on these theories, the algorithm aims to build schedules for jobs being processed on certain machines while minimizing the system makespan.

Schedules are built randomly by combining different genes into one complete antibody. The antibodies are first evaluated to determine the system makespan and then mutated, in an attempt to improve it. There are two types of mutations involved—those within a particular chromosome and those within the antibody. When this process repeats itself up to a specified number of iterations, the best solution is the minimum makespan observed. Figure 2 explains the algorithm in flowchart form.

4.1 Building an antibody

The libraries, components, and genes of the components are used to build an antibody representing an ordered schedule of the jobs. In order to build such an antibody, the procedure has to:

1. Randomly select a library
2. Randomly select a component with that library
3. Copy the genes from that component into the antibody
4. Randomly select another library (a library once selected cannot be selected again)
5. Randomly select a component and add the genes from that component into the schedule starting from a position after the end of the previous gene string
6. Go to Step 4 and continue the process until the entire antibody is filled.

4.2 Deciphering an antibody

After building an antigen by combining components from different libraries, the next step is to decipher an antibody and build a schedule by assigning jobs onto machines in the specific order of precedence. The objective of building the schedule is to obtain the makespan of that schedule. The makespan is the characteristic that is chosen to evaluate the optimality of a schedule and comparing it to other schedules. The Positive Selection Algorithm, as derived from the human immune system, selects only those antibodies that are useful in fighting the infection. In this case, the infection is a solution that is considered a 'bad' solution. Similarly, the algorithm seeks out those antibodies that show potential in finding good solutions to the JSSP. In order to recognize potentially good solutions, a threshold is specified for makespan values, and those solutions that are below the defined threshold are retained in the system, while the others are completely discarded from the system. Thus, a threshold is a specified value that forms the line of separation between 'good and bad' solutions. This value should be carefully specified to preserve as many good solutions as possible, but at the same time, not unnecessarily retain solutions that ultimately have no chance to reaching optimality.

4.3 Antigen library

Those schedules that are below the threshold create a new set of solutions called the antigen library, while the others are discarded. These solutions are now converted to antigens because future solutions have to better these solutions in order to be retained in the system. It is called a library since it contains all the solutions that pass the threshold barrier. Thus, the Antigen Library is a collection of good solutions, which set a standard for future solutions to attain and overcome, similar to how antibodies overcome the infections that attack the human body. There is no constraint on the total number of antibodies that can fit into the Antigen Library as all of those that qualify will be selected. Figure 3 presents a theoretical representation of what an Antigen Library would look like. Figure 3 shows a relatively small problem of three jobs being scheduled on two machines. The strings of numbers are those antibodies that were below the threshold and are now considered to be antigens. The labels A1 through A7 are the identifiers of each antigen. The total numbers of antigen that can be stored in an Antigen Library are fixed by definition. If new antibodies

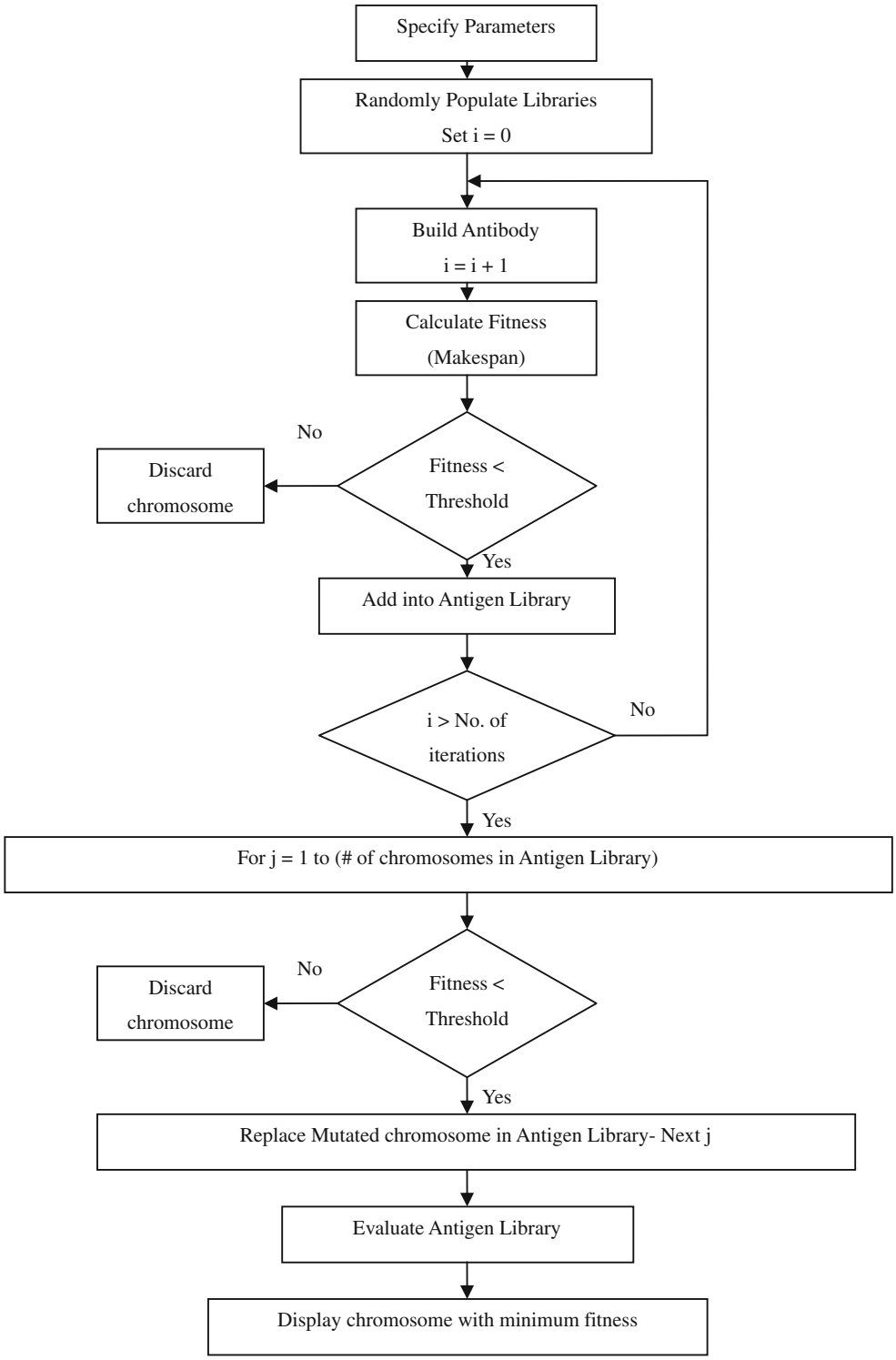

Fig. 2 Flowchart of the algorithm

with makespan less than any antigen in the library are created, they are inducted into the library as an antigen, and it will replace that antigen that has the highest makespan (or worst value) in the library.

The process of generating antibodies, evaluating them, and including selected ones, which are below the threshold, into the antigen library is performed as many times as the specified number of iterations.

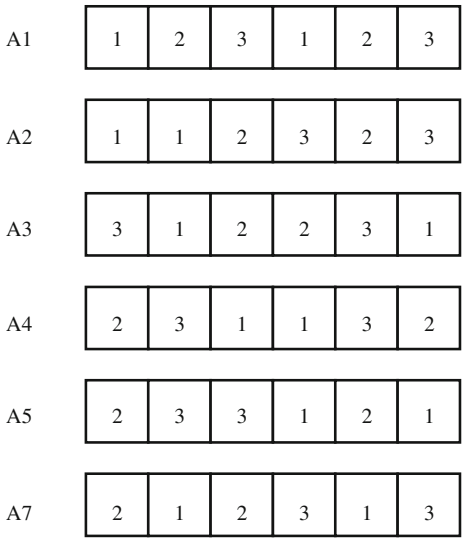

Fig. 3 The antigen library

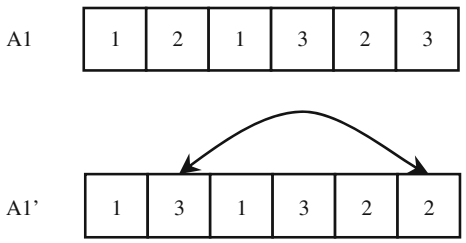

Fig. 4 Mutation of an antigen

4.4 Mutation

Once an absolute antigen library is formed, the newly created antigens undergo the process of mutation. In AIS, the process of mutation is emulated by randomly swapping the position of two genes of an antigen as shown in Fig. 4. The process of mutation is carried out only on those solutions, which show potential, (i.e. on those antibodies that are converted into antigen and included in the antigen library). The function of the mutation operator is to modify the solutions with the aim of identifying the optimal solution and to make the entire process more logical, less time-consuming, and incorporate a high level of efficiency. The mutation operator is carried out only on good solutions, which have high probabilities of reaching optimal or near-optimal.

The process of mutation is implemented by randomly choosing two genes and interchanging their respective values. The number of times this process is executed depends upon the number of mutations specified. The use of mutation is to exploit a solution, that is, make a good solution even better. The process is illustrated in Fig. 4 where A1 is an antigen, which will undergo mutation. Now, two random genes are selected. In this case, Gene 2 and Gene 6, and their values are exchanged. Thus, the value 2 in Gene 2 is exchanged with value 3 in Gene 6. A1 is the new mutated antigen.

All of the antigens in the Antigen Library undergo mutation. After each antigen is mutated, the new antigen is deciphered, evaluated, and compared to the original antigen. If the makespan produced by the new antigen is better than the original, this new antigen replaces the original one in the system. The original antigen is discarded from the Antigen Library. The process of mutation is an adaptation of the Clonal Selection Algorithm with the slight modification in that the clones are not exactly the same but slightly mutated in nature. The purpose of this is to improve the optimality of solutions by taking good ones and slightly modifying them in a random manner with the hope of further minimizing the fitness value.

4.5 Final evaluation

After the entire set of antigen in the Antigen Library undergo mutation, the antigen now present in the library are a group of the best solutions that can be obtained by the algorithm developed. The final step in this process is to now analyze the antigen library and display it as the final result with one solution that has the least value of makespan among all antigens within the antigen library. The solution may not be the absolute optimal solution to the problem, but is just the optimal solution reached by this algorithm. Further, taking another run of the same problem might drastically change the results due to the random nature of the algorithm. Generally, it takes multiple runs of the problem to get to the best results.

5 Foundation for comparison

The hybrid algorithm was tested on a total of basic 10 JSSPs that were obtained from the literature. To test and prove the feasibility of the proposed algorithm, it was developed into a program using the Visual Basic programing language. The VB 6.0 version was used since it provided all of the features required to successfully build, run, and test the algorithm. The tested problems were FT06 and FT10 [11], ABZ5 and ABZ6 [1], LA01–LA05, and LA16 [23]. A summary of the problems being tested and the best solution posted (optimal to date) of each problem are listed in Table 2. A general methodology was applied to quickly disembark on the most feasible solution. A backward approach starting with the knowledge of the known optimal allowed for greater ease and flexibility in the choice of input parameters such as the number of mutations, the number of iterations, and others. With the

Table 2 List of problems being tested

Problem #	Name of problem (jobs × machines)	Optimal solution
1	FT10 (10 × 10)	930
2	FT6 (6 × 6)	55
3	ABZ5 (10 × 10)	1,234
4	ABZ6 (10 × 10)	943
5	LA01 (10 × 5)	666
6	LA02 (10 × 5)	655
7	LA03 (10 × 5)	597
8	LA04 (10 × 5)	590
9	LA05 (10 × 5)	593
10	LA16 (10 × 10)	945

approach for this analysis being based on a trial and error method, this approach will help reduce the number of program runs required to get to the best solution. The methodology followed for the testing of each problem is described below:

- For each problem being tested, the known optimal solution was obtained from the literature (as noted above)
- With the aim of attaining this known optimal, the fitness threshold was set only slightly higher than the known optimal solution. This had no advantage in solving the solutions, only aided in reducing the computation time for the computer.
- Based on the results obtained, the parameters were then adjusted
    - Case 1: If the known optimal solution were to be obtained in the first run itself, the number of mutations and number of iterations would be reduced with the aim of determining the lowest number of total iterations required to get to the optimal solution
    - Case 2: If a solution is obtained but it is higher than the known solution, then the number of mutations and the number of iterations would be raised slowly until the known optimal is reached; if the known optimal is not reached, the best solution chosen is the solution that is closest to the best solution
    - Case 3: If a solution is not obtained (i.e. no schedule can be generated within the specified fitness threshold), the fitness threshold is raised, while the number of mutations and the number of iterations remains unchanged
    - Based on the results of each run, one of the above 3 cases were chosen to fine tune the solution obtained
- The lowest minimum makespan obtained through the testing process for each problem is chosen as the best solution; so far, the optimality as well as efficiency of the program will have been tested
- The next step is to test the reliability of the solution obtained; in order to do this, the parameters with which the best solution was obtained will be run 20 times to test the number of times the optimal solution is reached

Finally, the results were compared to GA solutions. The same problems were run on the GA Software developed in JAVA for 20 times in order to test robustness. The same parameters that are optimality and robustness were determined and then used for comparison. The purpose of comparing the solutions obtained with the AIS algorithm to the GA is to evaluate the results obtained with the AIS against a known benchmark.

6 AIS results and discussions

There are many ways of judging the optimality of solutions for a JSSP such as minimum makespan, lateness, tardiness, and number of late jobs. However, the problems considered are of the minimum makespan type, and thus, the primary concern in testing the viability of the algorithm is to verify whether the algorithm attains the known values of optimality. In this research, optimality is defined as *closeness of the best known solution* with the AIS algorithm to the known optimal solution for each of the problems. Alongside with optimality, the *robustness* is another key parameter that will test the performance of the algorithm. Robustness is the number of times that the best solution will be reached out of the total number of iterations for each of the problems being tested. Other areas of consideration of analyzing the results may be how the algorithm performs at solving problems of different sizes and how the results obtained in each case vary.

Further, if optimal solutions are obtained, then the secondary areas of investigation are the time and number of iterations taken to get to the solution. Each test followed the methodology listed in the previous section. The results from testing the AIS algorithm are listed in Table 3 for each of the problems tested. For example, for problem #1: The FT 10 × 10 job shop scheduling problem the known optimal makespan is 930. The known optimal solution of 930 was not obtained. The best solution reached with the AIS algorithm was 1,208 that was reached with 96,100 iterations.

7 Comparison conclusions

Based on this comparison, some meaningful conclusions can be drawn in terms of the quality of the results obtained,

Table 3 Summary of results with the AIS and GA

Problem #	Name of problem (jobs × machines)	Optimal solution	AIS		GA	
			Best solution	No. of best solutions (%)	Best solution	No. of best solutions (%)
1	ft10 (10 × 10)	930	1,208	10	1,099	5
2	ft06 (6 × 6)	55	55	90	55	15
3	abz5 (10 × 10)	1,234	1,434	15	1,339	5
4	abz6 (10 × 10)	943	1,084	15	1,043	5
5	la01 (10 × 5)	666	702	65	666	25
6	la02 (10 × 5)	655	708	55	716	5
7	la03 (10 × 5)	597	672	40	638	5
8	la04 (10 × 5)	590	644	35	619	10
9	la05 (10 × 5)	593	593	100	593	100
10	la16 (10 × 10)	945	1,124	20	1,033	5

the strengths and weaknesses of the AIS algorithm and some opportunities for improvement can be scoped out.

The results obtained with the AIS were compared with those obtained with the GA. The same ten job shop problems that were used for testing AIS were also run on a GA. There were performed 20 simulations for each problem to test robustness. The GA used for this comparison was devised by Shah [26] that created a distributed type GA model where the algorithm applied an order-based crossover methodology so that the offspring chromosome receives a random substring from the parent chromosome. The selection strategy used was the *Roulette Wheel*-based selection strategy.

In general, the GA had more solutions closest to the optimal solution, and the AIS algorithm had more robust solution than the GA. A summary of the results is shown in Table 3. The two attributes compared are the best solution based on lowest makespan and the *number of best solutions* that is an indicator of robustness. The best solution may or may not be equal to the optimal solution. The optimal solution of each problem is also listed for reference. The number of best solutions refers to the number of times the best solution was reached as a percent of the total 20 iterations that were run for each problem for each algorithm.

Overall, the GA produced better solutions than AIS algorithm. Not only did the GA get more optimal solutions that the AIS, it also, in almost all non-optimal solutions, got closer to the optimal solution than the AIS did. The GA obtained the optimal solution in 30% of the cases, while the AIS algorithm reached the optimal makespan in 20% of the cases. The genetic algorithm also obtained the optimal solution for both the problems for which the AIS reached optimality (i.e. FT06 and LA05). However, the AIS obtained the same solutions for 90 and 100% of the iterations, respectively, where the GA obtained the same solution only for 15 and 100%, respectively. The LA01 problem was the only other problem for which the GA got the optimal solution of 666 makespan units with a robustness of 25%.

For the rest of the problems that were tested, the results generated by the GA were closer to the optimal solution than those generated by the AIS algorithm. While comparing the 7 problems that did not arrive at the optimal solution, it should be noted that the GA produced better results than the AIS in 6 cases (85.7%). The only case where the AIS had a better result than the GA was with the LA02 (10 × 5) problem. In addition, the closest that the AIS reached to the GA in the case on non-optimal results was within 25 makespan units for the LA04 (10 × 5) job shop problem, while the farthest was a difference of 109 makespan units for the FT10 (10 × 10) job shop problem. It should be noted that the difference in makespan between AIS and GA solutions increases with the number of machines (Table 4).

The robustness of the results produced by the two algorithms was estimated by the number of times the best solution was reached out of the total 20 times that each problem was run by each algorithm. While it was observed in the previous section that the GA produced better optimality results, in this case, the AIS was observed to have more robust solutions. Figure 5 compares the degree of robustness between the two techniques. The only exception was with the LA05 (10 × 5) problem where both AIS and GA obtained the optimal solution 100% of the time. For the FT06 problem, even though both techniques reached the optimal solution, the AIS was more robust as it reached optimality 90% of the time compared to the GA reaching optimality 15% of the time.

Another measure of comparison was looking at the interaction between the degree of optimality of the solution and the robustness as shown in Fig. 6. This Figure depicts this relationship for AIS and GA, respectively. Linear

Table 4 Comparison of results between AIS and GA

	FT10			FR06			ABZ5			ABZ6			LA01		
	OS	AIS	GA	OS	AIS	GA	OS	AIS	GA	OS	AIS	GA	OS	AIS	GA
Optimal solution	930			55			1,234			934			666		
Best solution		1,208	1,099	55	55	55	1,234	1,434	1,339	943	1,084	1,043	666	702	666
Robustness	–	2	1	–	18	3	–	3	1	–	3	1	–	13	5
Optimality (difference)		−278	−169		0	0		−200	−105		−141	−100		−36	0
Robustness (%)		10	5		90	15		15	5		15	5		65	25

	LA02			LA03			LA04			LA05			LA16		
	OS	AIS	GA	OS	AIS	GA	OS	AIS	GA	OS	AIS	GA	OS	AIS	GA
Optimal solution	655			597			590			593			945		
Best solution	655	708	716	597	672	638	590	644	619	593	593	593	945	1,124	1,033
Robustness	–	11	1	–	8	1	–	7	2	–	20	20	–	4	1
Optimality (difference)		−53	−61		−75	−41		−54	−29		0	0		−179	−88
Robustness (%)		55	5		40	5		35	10		100	100		20	5

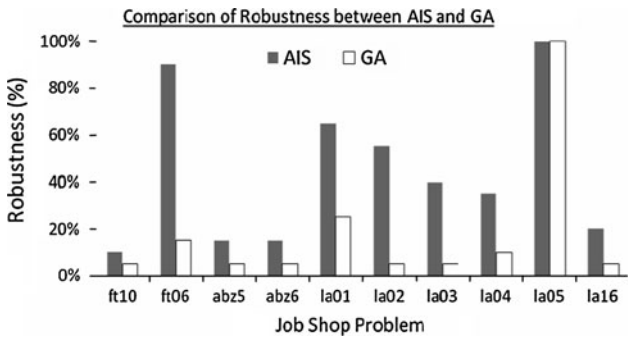

Fig. 5 Relationship between optimality and robustness for GA and AIS

regression lines were calculated for both the AIS and GA. For both algorithms, the two variables have an inverse relationship so that the robustness of the algorithm increases as the variation of the best solution from the optimal solution decreases. These linear lines are used to illustrate that both the algorithms have the same general trend. Even though these lines may not be the best fit by using a higher degree polynomial function, it does illustrate that the AIS has a higher slope. This further supports the conclusion that the AIS algorithm is more robust than the GA.

In conclusion, the comparison of the AIS algorithm with the GA revealed that while the AIS matched up well to the GA on the relatively simple or smaller problems, the GA provided solutions that were closer to the optimal. So, there is still scope for fine-tuning the AIS algorithm. On the positive side, the AIS solutions were more robust and reliable. Although this would require further testing, the implication of getting robust solutions could mean that lesser iterations would be required to determine the best

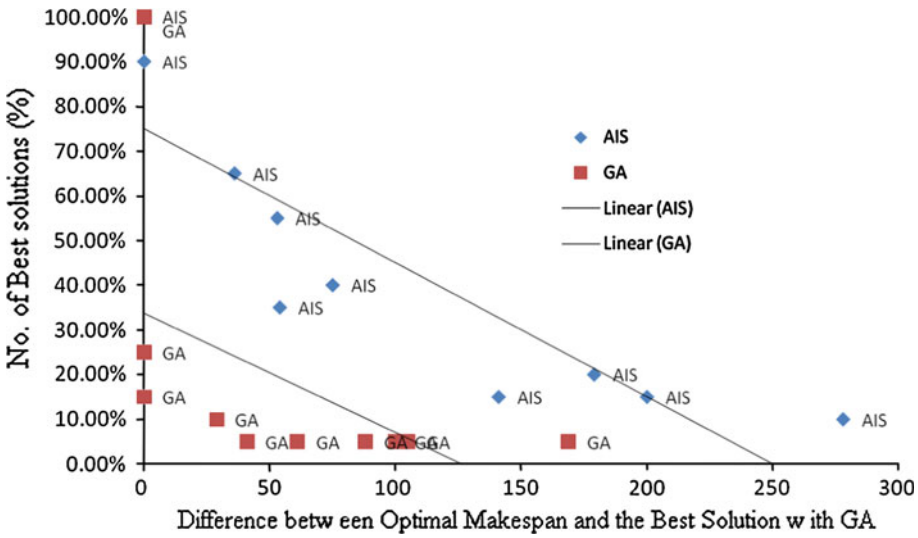

Fig. 6 Interaction between quality of solution and robustness for GA and AIS

solution with the AIS algorithm under the hypothesis that as the best solution is reached, that particular solution will appear more frequently. On the other hand, it could take more scenario runs, and hence, more time to reach the best solution with the GA.

References

1. Adams J, Balas E, Zawack D (1986) The shifting bottleneck procedure for job shop scheduling. Manag Sci 34:391–401
2. Aickelin U, Burke E, Mohamed Din A (2004) Investigating artificial immune systems for job shop rescheduling in changing environments. In: 6th international conference in adaptive computing in design and manufacture, Bristol, UK
3. Bersini H, Varela FJ (1990) Hints for adaptive problem solving gleaned from immune networks. Parallel problem solving from nature. pp 343–354
4. Bersini H, Varela FJ (1991) The immune recruitment mechanism: a selective evolutionary strategy. In: Proceedings of the international conference on genetic algorithm, pp 520–526
5. Bersini H (1991) Immune network and adaptive control. Proceedings of the first European conference on artificial life. Towards a practice of autonomous systems. MIT Press, Cambridge MA, pp 217–226
6. Coello CA, Rivas DC, Cruz-Cortés N (2004) Job shop scheduling using the clonal selection principle. ACDM'2004. Springer, Bristol
7. Dasgupta D (1999) An overview of artificial immune systems and their applications. In: Dasgupta D (ed) Artificial immune systems and their applications. Springer, Berlin, pp 3–21
8. De Castro LN, Von Zuben FJ (2000) An evolutionary immune network for data clustering. In: Proceedings of the IEEE SBRN'00 (Brazilian symposium on artificial neural networks), pp 84–89
9. De Castro LN, Von Zuben FJ (2001) aiNet: an artificial immune network for data analysis. In: Abbass HA, Sarker RA, Newton CS (eds) Book chapter in data mining: a heuristic approach, Chap. XII. Idea Group Publishing, USA, pp 231–259
10. Farmer JD, Packard NH, Perelson AS (1986) The immune system, adaptation and machine learning. Physica 22D:187–204
11. Fisher H, Thompson GL (1963) Probabilistic learning combinations of local job-shop scheduling rules. In: Muth JF, Thompson GL (eds) Industrial scheduling. Prentice Hall, Englewood Cliffs, NJ, pp 225–251
12. Forrest S, Perelson AS (1991) Genetic algorithms and the immune system. Proceedings of the parallel problem solving from nature. Springer, Berlin
13. Forrest S, Javornik B, Smith RE, Perelson AS (1993) Using genetic algorithms to explore pattern recognition in the immune system. Evol Comput 1(3):191–211
14. Fukuda T, Mori M, Tsukiyama M (1999) Immunity-based management system for a semiconductor production line. In: Dasgupta D (ed) Artificial immune systems and their applications. Springer, Berlin, pp 278–288
15. Giffler B, Thompson G (1960) Algorithms for solving production scheduling problems. Oper Res VIII:487–503
16. Hart E (2002) Immunology as a metaphor for computational information processing: fact or fiction? PhD thesis. Artificial Intelligence Applications Institute, Division of Informatics, University of Edinburgh
17. Hart E, Ross P (1999) An immune system approach to scheduling in changing environments. Genetic and evolutionary computation conference—GECCO 1999, Orlando, Florida, USA. pp 1559–1565
18. Hart E, Ross P, Nelson J (1998) Producing robust schedules via an artificial immune system. International conference on evolutionary computing, ICEC '98, Anchorage, AK. IEEE Press, New York, pp 464–469
19. Hoffman GW (1986) A neural network model based on the analogy with the immune system. J Theor Biol 122:33–67
20. Ishida Y (1990) Fully distributed diagnosis by PDP learning algorithm: towards immune network PDP model. In: Proceedings of the international joint conference on neural networks, pp 777–782
21. Ishida Y (1993) An immune network model and its applications to process diagnosis. Syst Comput Jpn 24(6):646–651
22. Land AH, Doig AG (1960) An automatic method of solving discrete programming problems. Econometricca 28(3):497–520
23. Lawrence S (1984) Resource constrained project scheduling: an experimental investigation of heuristic scheduling techniques (Supplement), Graduate School of Industrial Administration. Carnegie-Mellon University, Pittsburgh, PA
24. Nasaroui O, Gonzalez F, Dasgupta D (2002) The fuzzy artificial immune system: motivations, basic concepts, and application to clustering and web profiling. Published at IEEE international conference on fuzzy systems. In: Proceedings of the IEEE world congress on computational intelligence
25. Neal MJ (2002) An artificial immune system for continuous analysis of time-varying data. In: Proceedings of ICARIS 2002, Canterbury, UK
26. Shah N (2004) Using distributed computing to improve the performance of genetic algorithms for job shop scheduling problems. Master's Thesis, Department of Industrial and Manufacturing Systems Engineering, Ohio University, Athens, OH
27. Timmis J (2000) Artificial immune systems: a novel data analysis technique inspired by the immune network theory. PhD thesis, Department of Computer Science, University of Wales, Aberystwyth, Ceredigion, Wales