Artificial Intelligence Review **19:** 123–152, 2003.

Sexual Selection for Genetic Algorithms

KAI SONG GOH[1], ANDREW LIM[1] and BRIAN RODRIGUES[2]
[1]*Department of Computer Science, School of Computing, National University of Singapore, 3 Science Drive 2, Singapore 117543 (E-mail: alim@comp.nus.edu.sg);* [2]*School of Business, Singapore Management University, 469 Bukit Timah Road, Singapore 259756*

Abstract. Genetic Algorithms (GA) have been widely used in operations research and optimization since first proposed. A typical GA comprises three stages, the encoding, the selection and the recombination stages. In this work, we focus our attention on the selection stage of GA, and review a few commonly employed selection schemes and their associated scaling functions. We also examine common problems and solution methods for such selection schemes.

We then propose a new selection scheme inspired by sexual selection principles through female choice selection, and compare the performance of this new scheme with commonly used selection methods in solving some well-known problems including the Royal Road Problem, the Open Shop Scheduling Problem and the Job Shop Scheduling Problem.

Keywords: genetic algorithm, scheduling, selection

1. Introduction

The ideas behind Genetic Algorithms (GA) (Holland, 1975) are derived from the theory of natural selection originally proposed by Charles Darwin (Darwin, 1888). In GA, potential solutions to a target problem are viewed as individuals in a single population where the fitter individuals in any generation are allowed to reproduce and, in the process, bring forth a new generation of individuals (solutions). As the population of solutions evolves, fitter solutions are produced and, eventually, optimal or near-optimal solutions are reached.

There are three main stages in the implementation of GA. The *encoding stage*, the *selection stage*, and the *recombination stage*.

The **encoding stage** is where solutions to the target problem are expressed in a chromosome-like data structure. Usually at the initial encoding stage, solutions are generated randomly; however, there are cases where problem-specific heuristics are used. The encoding stage will produce the initial population (or generation 0) of individuals on which evolution will be based. Since problems differ from one another, the encoding stage is usually problem-specific.

The **selection stage** is where individuals in the current generation are chosen from the population and allowed to reproduce. Most selection mechanisms are based on the fitness of individuals in the population; hence, during this stage, it is common to have a measurement of the fitness of the solutions. The fitness of a solution is often how well the solution addresses the objective function of the target problem. Selection usually favors the fitter individuals.

The third stage is the **recombination stage**. In this stage, the selected individuals (parents) are used to produce child solutions. Two forms of reproduction are usually used: (1)The single-parent reproduction technique, or mutation technique, which involves some modification to the chromosome of the parent solution, with the resulting chromosome becoming the child solution, and (2)The multi-parent technique or crossover technique. In this method, i ($i \geq 2$), parents are chosen from the current generation and part of the chromosomes of each parent is then combined to form a child. It is not uncommon for the crossover process to produce infeasible child solutions; hence, certain implementations of GA include repair algorithms that will modify child solutions once crossovers are executed. Once a child is produced, it is integrated into the population of the next generation.

2. The Selection Stage

In his study of selective mechanisms, Thomas Back (Back, 1994) showed that the selection process can control the level of exploration or exploitation by varying the level of emphasis the process assigns to fitter individuals. A more stringent selection process that is more biased towards the fitter solutions will push a search towards exploitation while a less stringent selection process will push a search towards exploration.

If a selection is biased toward the fitter solutions, it exploits the knowledge that the current arrangement of the chromosome-like structure of the individual is good and hence when mated with another solution will probably retain certain traits and generate fitter offsprings. However, most search spaces are undulating, and there may exist several local optima in which searches can get trapped if there is an over-emphasis on exploitation. On the other hand, if a selection is biased towards unfit solutions (which is rarely the case since this violates the principle of survival of the fittest), it favors the exploration of the search space as the weaker solutions may allow escape routes from local optima. Likewise, over-emphasis on exploration has its ill-effects; a selection that is too much biased towards exploration can result in a longer search and will probably require a longer evolutionary process.

To provide a balance between exploration and exploitation, most selection mechanisms introduce a degree of randomness which ensures that the fittest

solution will not always be chosen, and the weakest solution will not always be neglected. However, this alone is not sufficient in most cases and usually leads to *premature convergence* where the search converges on a trait of a fit solution too early in the evolutionary process and eventually all solutions in the population exhibit this trait. Usually, this happens because the difference between fit solutions and unfit solutions is too large.

To counter this problem, several scaling mechanisms were introduced. Scaling mechanisms use functions that will convert the raw fitness of solutions to a scaled fitness with the aim of normalizing or reducing the difference between them. Scaling is not a simple process as there are usually parameters to be tuned and which differ according to the different search spaces encountered for various problems.

In this work, we propose a new selection method that will provide a clear separation between exploration and exploitation without the inconvenience of parameter tuning in scaling. The main idea is to separate the population into two sexes, males and females. The selection of females will handle the exploration of the search space while the selection of males will handle the exploitation of the search space. This is based on the idea of sexual selection operating through female choice. Hence, all the females will get to mate while only the more attractive males will be selected for mating, with unattractive males eventually being eliminated.

3. Common Selection Schemes and Scaling

The significant impact of the selection process in GA and, more generally, evolutionary algorithms has prompted several studies in this area, see (Kolarov, 1995), (Muhlenbein and Schlierkamp-Voosen, 1993) and (Blickle and Thiele, 1995). We will now take a look at the more commonly utilized selection mechanisms.

Proportional Selection – Proportional Selection is also known as Roulette Wheel Selection or Spinning Wheel Selection because the probability of an individual's selection corresponds to a portion of a Roulette Wheel. Each individual is assigned a selection probability based on its fitness over the total fitness of the entire population. Once this is done, individuals are concatenated with each other to form what can be visualized as a spinning wheel, with each portion of the wheel representing an individual and the size of each portion representing the corresponding selection probabilities. Back and Hoffmeister (Back and Hoffmeister, 1991) classify Proportional Selection as a *preservative* selection scheme, where every individual has a non-zero chance of being selected.

Sigma scaling is commonly used in conjunction with proportional selection. Sigma scaling produces a scaled fitness, f_{scaled}, from the raw fitness, f_{raw}, using the formula:

$$f_{scaled} = S + \frac{f_{raw} - f_{mean}}{2\sigma}$$

where,
S is the scaling factor, $1 \leq S \leq 5$,
f_{mean} is the mean fitness of all individuals in the population, and σ is the standard deviation.

Tournament Selection – Tournament Selection is one of the more commonly used selection schemes, perhaps because of its simplicity. The basic idea of tournament selection scheme is quite straightforward. A group of individuals is selected randomly from the population. The individuals in this group are then compared with each other, with the fittest among the group becoming the selected individual. Typical implementation of Tournament Selection involves picking only two individuals for comparison. Scaling in Tournament Selection works by introducing an acceptance threshold, t, into the selection. Each time, a random number, r ($0 \leq r \leq 1$), is generated. If $r < t$ then the fitter individual is selected, else the weaker individual is selected. Usually, t is set to be in the range $0.75 \leq t \leq 1$.

Rank-based Selection – Rank-based selection schemes first sort individuals in the population according to certain criteria (usually according to their fitness). A function is then used to map the indices of individuals in the sorted list to their selection probabilities. Although this mapping function can be linear (linear ranking) or a non-linear (non-linear ranking), the idea of rank-based selection remains unchanged. We note, however, that the performance of the selection scheme depends greatly on this mapping function. Rank-based selection schemes belong to the group of *static* selection schemes. These are schemes where the nth fittest individual in generation G_1 will have the same selection probability as the nth fittest individual in generation G_2 even if they differ in fitness.

Scaling in Ranked-based Selection involves preserving the top $p\%$ of the population and then ranking them according their fitness. The scaled fitness of these individuals are then set to be their rank. Consider the case where p = 80%, where we will eliminate the weakest 20% of the population based on raw fitness. The remaining 80% of the population are then ranked according to their raw fitness. Next the scaled fitness is allocated to individuals, with the top individual scoring 80% of the population size, and next scoring 79% and so on, ending with the weakest individual being allocated a scaled fitness of 1% of the population size. Recommended values of p are in the range $80\% \leq p \leq 100\%$.

We will compare the performance of these selection schemes in our experiments against that of our newly proposed selection scheme.

4. Sexual Selection Scheme

The Sexual Selection scheme was inspired by the concept of sexual selection proposed by Darwin (Darwin, 1888) which suggests that mate choice in some species operates through *female choice*. Our implementation comprises two main stages. The first stage involves determining the sex of individuals in the current generation. This can be done either randomly or based on some problem-specific knowledge where individuals with certain traits are chosen to be of one sex while the rest are chosen to be of another sex.

The second stage involves the actual selection of a pair of individuals (one male and one female). The selection of a female is quite straightforward. We propose that *all* females will get to reproduce regardless of their fitness level; hence, females are selected in a sequential fashion without replacement which means that each female will only get to be selected for mating once. The male selection stage is fitness biased. Currently, we use tournament selection, with tournament size two, and without any scaling. A different approach is to base male selection on some compatibility function with the selected female. However, it should be noted that by doing this, the selection scheme would become problem dependent. Hence we choose to use random separation of males and females and base our male selection purely on fitness.

The Sexual Selection scheme is implemented as follows:

1. Separate the current generation into males and females
2. Select the next unmated female
3. Select a male
4. Mate the female and male to produce offsprings
5. Return to Step 2 until all females have mated

This design ensures that all the females get to reproduce regardless of their fitness and aims to facilitate *exploration* of the search space. On the other hand, bias towards the better performing males *exploits* the knowledge that the current chromosome-like structure of the individual is good.

We have proposed this selection scheme with the intent of having a clear and balanced separation of functions between exploration and exploitation without the inconvenience of tuning parameters. Figure 1 to Figure 3 illustrate the Sexual Selection process.

Kai Song Goh and Andrew Lim and Brian Rodrigues

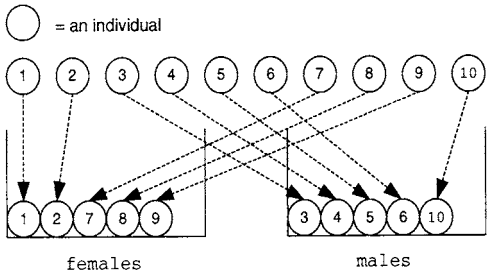

Figure 1. Separate current generation into females and males.

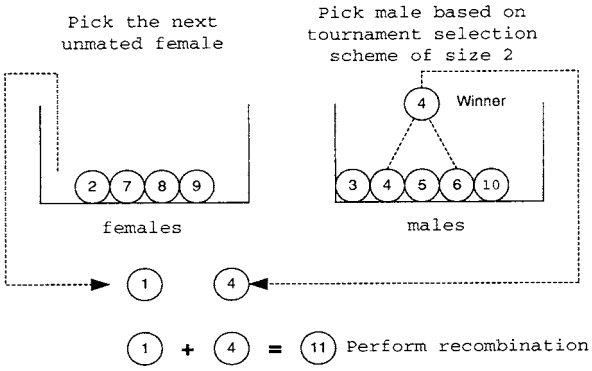

Figure 2. Pick the next unmated female and select a male based on tournament selection.

Sexual Selection for Genetic Algorithm

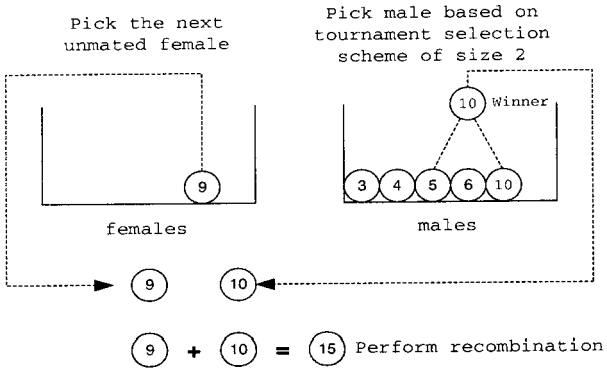

Figure 3. Process is repeated until all females have mated.

4.1. *Related works*

Gender in a GA process is not a new idea. However, sexes are defined for the purpose of solving multi-objective optimization where there will be one gender for each optimization criterion and the GA will select one individual from each gender to perform recombination so as to produce offspring (Lis and Eiben, 1996) Similarly, the idea of mate choice has also been incorperated in GA. Ratford (Ratford et al., 1997a), (Ratford et al., 1997b) and Ronald (Ronald, 1995) both proposed selection schemes in which the first mate is selected using a traditional selection method, with the second mate being selected based on some *seduction function* between itself and the first mate. However in their work, there is no explicit notion of separate sexes and the seduction function is problem dependent.

5. Experiment Setup

The main objective of this study is to show that the performance of the Sexual Selection scheme is as good, if not superior, to other selection schemes, while having the advantage of removing the need to determine scaling parameters. We show that the new scheme provides a good balance between exploration and exploitation of the search space and, hence, will produce good results. We compare the performance of the following four selection schemes in our experiments.

1. Proportional Selection with Sigma Scaling
2. Tournament Selection with Acceptance Threshold
3. Rank-based Selection with Preservation Percentage
4. Sexual Selection

Theoretically, generation 0 in GA is produced randomly, since we do not know the impact of using different generation 0's for running GA with each of the selection schemes. We have chosen to produce a single generation 0 and start off each GA evolution (with different selection schemes) using the same generation 0 each time. To further ensure that performance variation can only arise from the different selection schemes employed, we employ consistently the same encoding scheme, fitness measure, recombination method and crossover points.

We are comparing the performance of the new selection scheme against that of the traditional selection schemes which employ fitness scaling. Our experiments were conducted using different parameter settings for each of the scaling functions. However, since it would take an extremely long time to empirically determine the best parameter setting for each of the scaling functions (indeed, a motivating factor for our new selection scheme in the first

Table 1. Selection schemes, scaling parameters and values experimented.

Selection Method/Parameter	Recommended Range	Values Experimented
Roulette/Sigma	1.0–5.0	1.0, 2.0, 3.0, 4.0, 5.0
Tournament/Accpt Threshold	0.75–1.0	0.75, 0.85, 0.9, 0.95, 1.0
Rank-based/Presv %	0.8–1.0	0.8, 0.85, 0.9, 0.95, 1.0

place) we have experimented with values that are within the "recommended" ranges.

To do this, a significantly large number of evolutions of the GA is run each time, testing different settings of the parameters which are increased at fixed intervals within the commonly-used ranges. The tested parameters for each selection scheme are presented in Table 1.

We have experimented with two different settings of the GA, setting 1 involves 30 individuals evolving for 300 generations while setting 2 involves 50 individuals evolving for 500 generations. Each selection scheme (with different scaling setting) had 30 runs of the GA and the best solution was captured. The best solution and the average of all the best solutions over the 30 experiments are then compared against the proposed selection scheme. Results of this comparison are presented in the figures at the end of each section featuring the test problem.

6. The Royal Road Problem

The Royal Road Problem (RRP) for Genetic Algorithms was studied in (Mitchell et al., 1992), (Mitchell et al., 1994) and (van Nimwegen et al., 1999) and was used as a benchmarking problem in (Matsui, 1999). It was originally proposed to aid the study of the relationship between the performance of GA with different features of a fitness landscape (see (Mitchell et al., 1992) for more details).

The Royal Road Problem involves a set of schemas $S = S_1, S_2, \ldots, S_n$ with which fitness of an individual is determined according the score allocated to each match between an individual and a schema.

The objective of the Royal Road problem is to obtain a bit sequence consisting of all $1's$ which is the optimal solution. The schema is represented by a series of $1's$ and $*'s$, where $*'s$ can be either 1 or 0. Consider a schema S_{eg} represented by $^{**}11$. In this case, individuals 1011, 0011, 0111 and 1111 are all instances of schema S_{eg}.

s1 = 11111111**********; c1 = 8
s2 = ********11111111**********; c2 = 8
s3 = ****************11111111**********; c3 = 8
s4 = ************************11111111**********; c4 = 8
s5 = ********************************11111111**********; c5 = 8
s6 = **11111111**********; c6 = 8
s7 = **11111111********; c7 = 8
s8 = **11111111; c8 = 8
s9 = 1111111111111111**********; c9 = 16
s10 = ****************1111111111111111**********; c10 = 16
s11 = ********************************1111111111111111**********; c11 = 16
s12 = **1111111111111111; c12 = 16
s13 = 11111111111111111111111111111111**********; c13 = 32
s14 = ********************************1111111111111111111111111111111111; c14 = 32
s15 = 1111111111111111111111111111111111111111111111111111111111111111; c15 = 64

Figure 4. The Royal Road fitness function.

An example of the schema is shown in Figure 4. In the example shown, we consider 64-bit sequences. S_1 to S_{15} represent the scoring schema and C_1 to C_{15} represent the scores awarded for each match. Hence an individual having 64 $1's$ would match schema S_1 and score 8 points; the same individual would also match schema S_2 and score another 8 points. Similarly, this individual would match schemas $S_3 \ldots S_{15}$. The total score for this individual would be $8 \times 8 + 4 \times 16 + 2 \times 32 + 1 \times 64 = 256$.

Individuals in the Royal Road problem are represented by a string of 0's and 1's. Individuals are of fixed length L (usually $L = 8^n$, where n is a positive integer) for the purpose of our experiments, we have chosen to work on individuals of length ranging from 16 to 2048 bits.

The formal definition of the fitness measure for the RRP is as follows:

Fitness of individual x, $F(x) = \sum_{s \in S} C_s \sigma_s(x)$, where
S = the set of scoring schema,
C_s = score allocated for schema S_s,
$\sigma_s = 0$ if x is *not* an instance of schema S_s, and
$\sigma_s = 1$ if x is an instance of schema S_s.

Since there is no need to maintain precedence among the bits or to avoid repetition of bits within the chromosome representation of the solution, the simple traditional two-point crossover recombination technique is employed in this experiment.

We experimented with 8 test cases with increasing magnitude, ranging from 16 to 2048 bits. The head-to-head comparison of the proposed Sexual

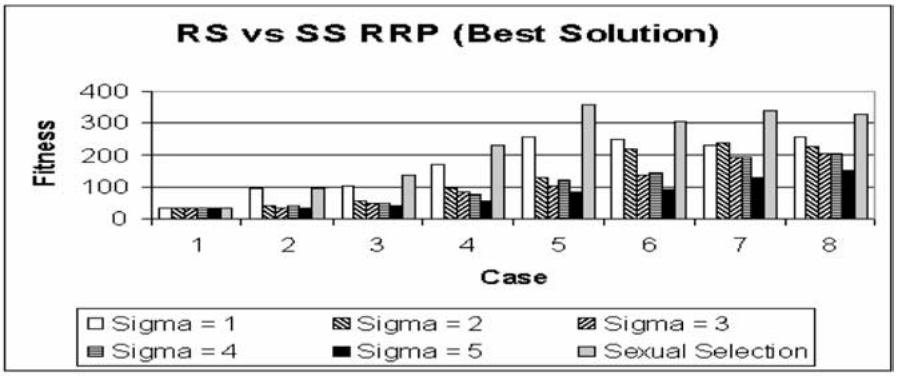

Figure 5. Best Results obtained by Roulette Selection compared against Sexual Selection (population = 30, generation = 300).

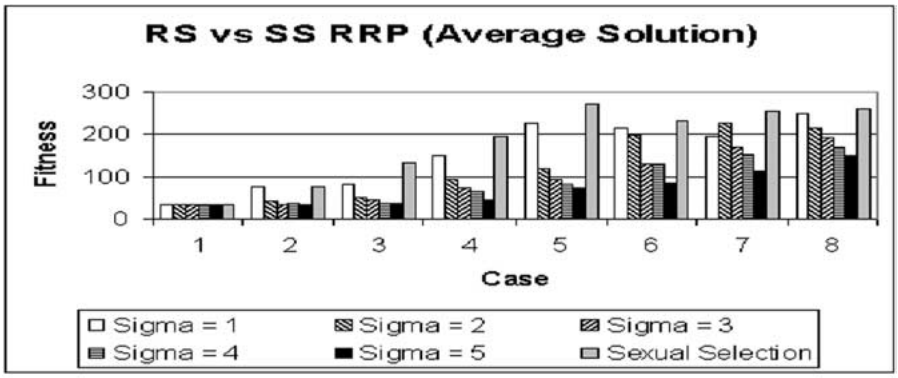

Figure 6. Average Results obtained by Roulette Selection compared against Sexual Selection (population = 30, generation = 300).

Selection scheme against the 3 other selection schemes for experiments conducted on 30 individuals over 300 generations are presented in Figures 5 to 10 to and results for experiments conducted on 50 individuals over 500 generation are presented in Figures 11 to 16.

7. The Open Shop Scheduling Problem

The Open Shop Scheduling Problem (OSSP) is another commonly selected problem for GA benchmarking purposes. (Khuri and Miryala, 1999) gives a fairly good introduction of the problem, which is also studied in (Fang et al., 1994) and (Louis and Xu, 1996).

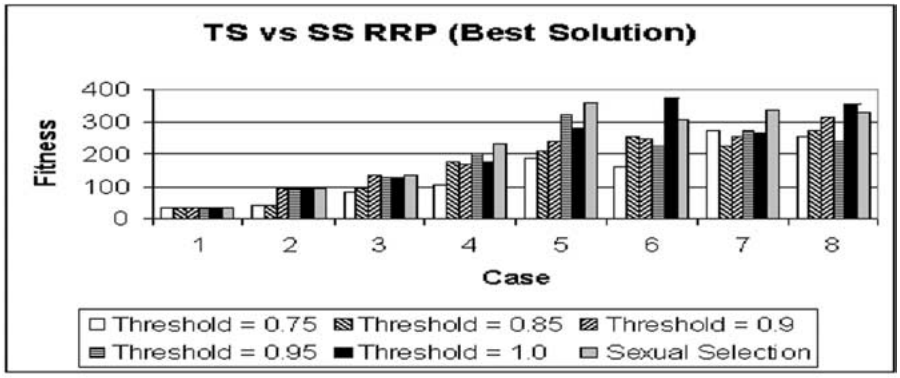

Figure 7. Best Results obtained by Tournament Selection compared against Sexual Selection (population = 30, generation = 300).

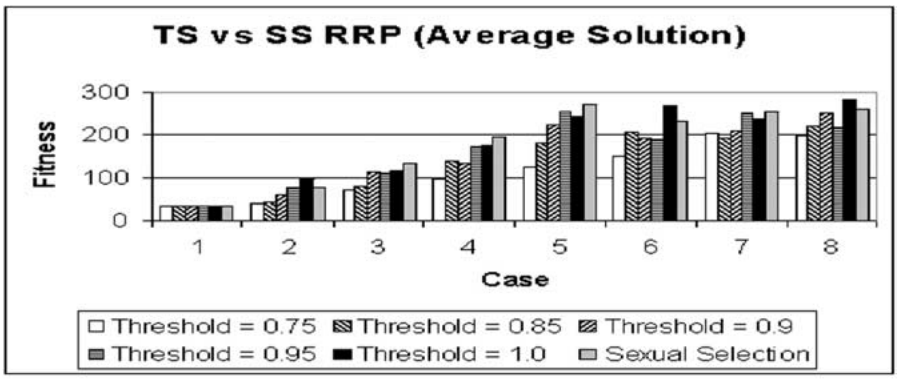

Figure 8. Average Results obtained by Tournament Selection compared against Sexual Selection (population = 30, generation = 300).

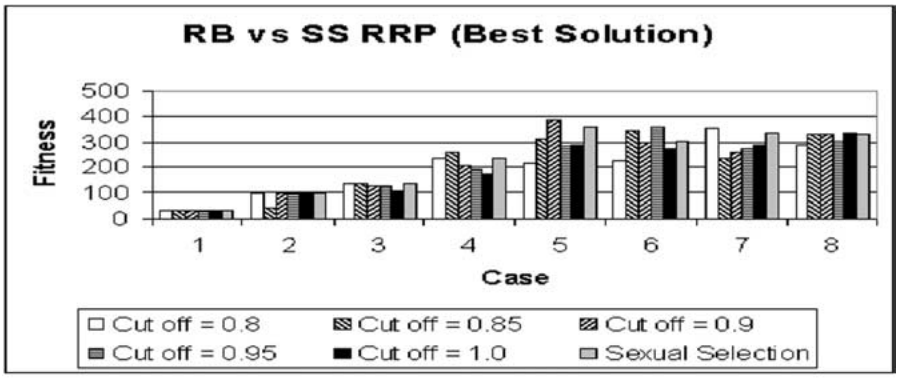

Figure 9. Best Results obtained by Rank-based Selection compared against Sexual Selection (population = 30, generation = 300).

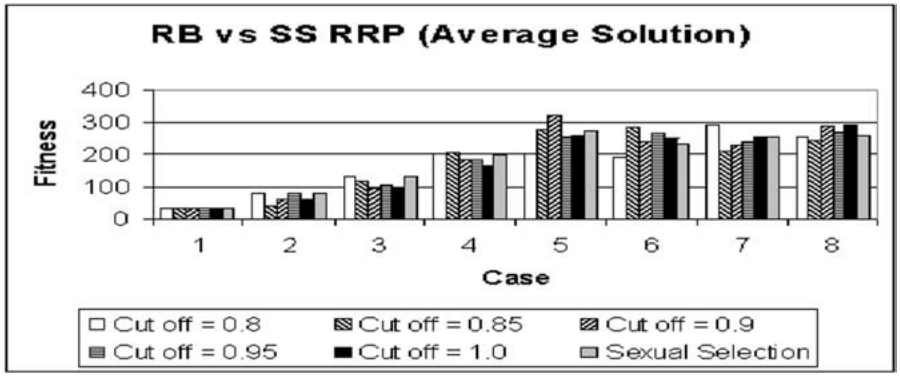

Figure 10. Average Results obtained by Rank-based Selection compared against Sexual Selection (population = 30, generation = 300).

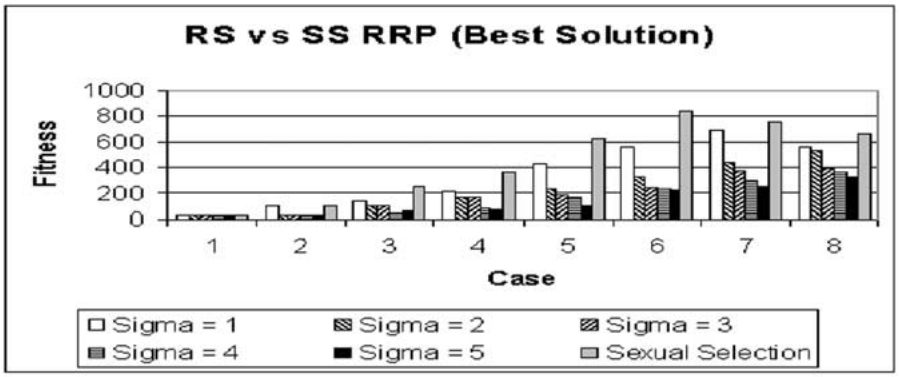

Figure 11. Best Results obtained by Roulette Selection compared against Sexual Selection (population = 50, generation = 500).

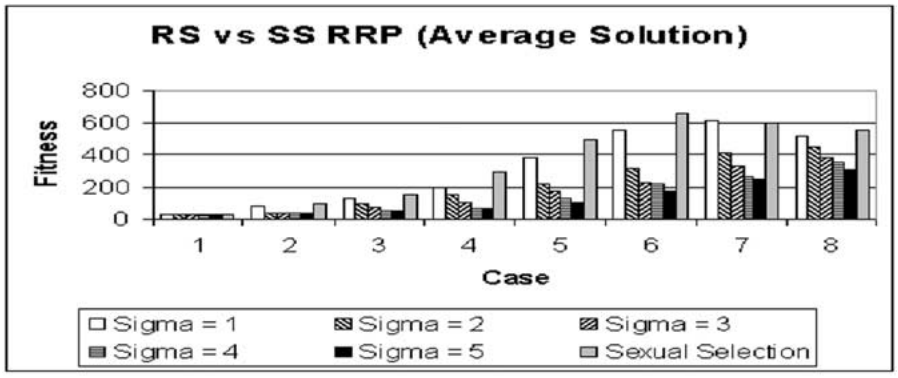

Figure 12. Average Results obtained by Roulette Selection compared against Sexual Selection (population = 50, generation = 500).

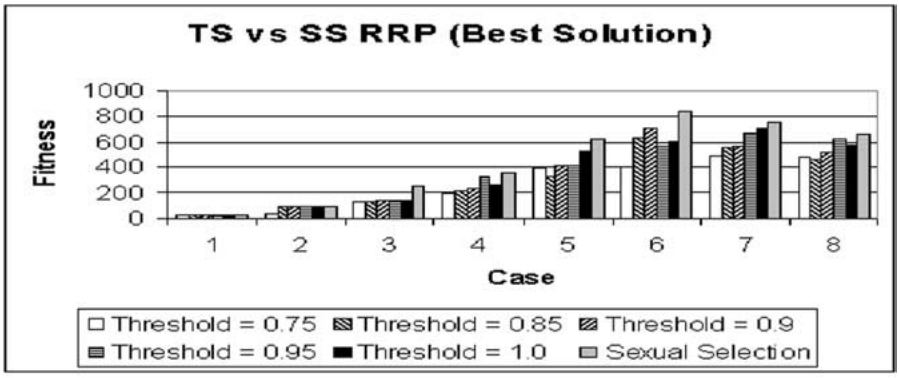

Figure 13. Best Results obtained by Tournament Selection compared against Sexual Selection (population = 50, generation = 500).

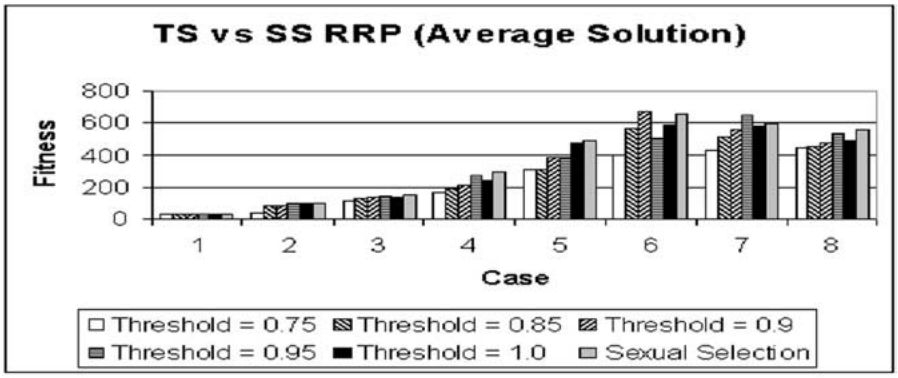

Figure 14. Average Results obtained by Tournament Selection compared against Sexual Selection (population = 50, generation = 500).

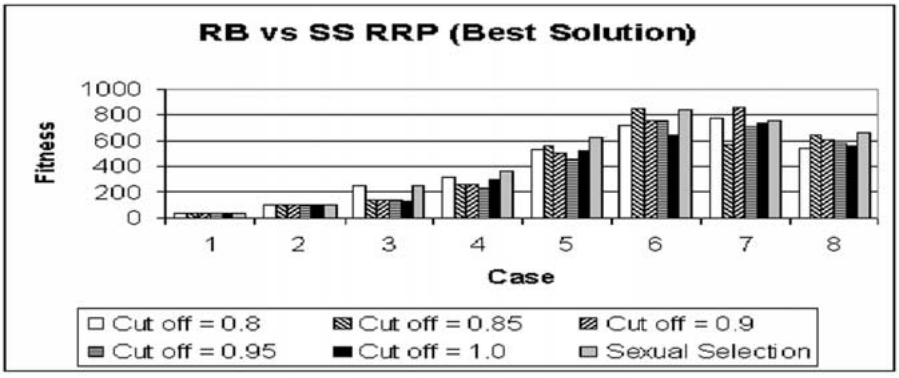

Figure 15. Best Results obtained by Rank-based Selection compared against Sexual Selection (population = 50, generation = 500).

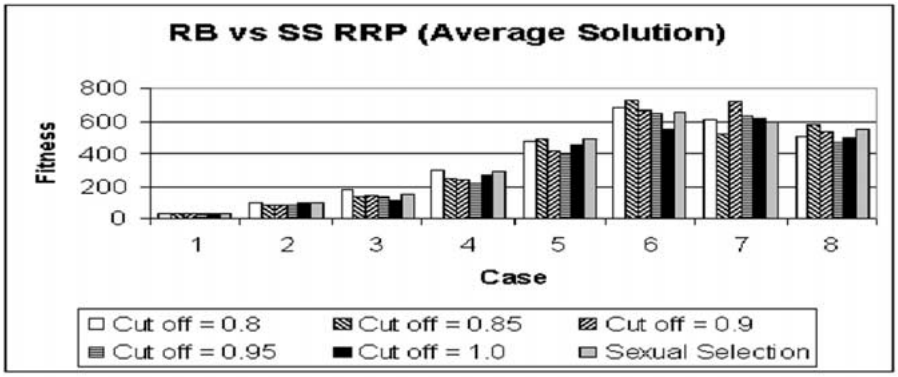

Figure 16. Average Results obtained by Rank-based Selection compared against Sexual Selection (population = 50, generation = 500).

The OSSP involves a set of jobs $J = j_1, j_2, \ldots, j_n$. Each job j_i consists of m operations, O_{ij} $(j = 1, \ldots, m)$, where each O_{ij} has to be processed on machine M_j for a period of time represented by P_{ij} without preemption.

The basic assumptions of the OSSP is that each machine can only process one operation at any time, and each job can only be processed by one machine at any time; this means that jobs cannot have two or more operations processed in parallel. There are no constraints on the order in which operations within a job are being processed.

The objective of the OSSP is *makespan* minimization. The makespan of a schedule is defined as the total elapsed time of the schedule.

The primary concern in solving the OSSP is the start time allocated to each of the operations, $O_{1,1}$ to $O_{n,m}$. Hence the representation of the problem is a string of length $m \times n$. The string will contain the sequence at which operations will be scheduled to their respective machines. Each operation will be scheduled to the respective machine at the *earliest possible time*.

Due to the constraints of the OSSP, the earliest possible time of any operations has two restrictions; firstly, the machine must be free, and secondly, the job to which the operation belongs must not have any on-going operations. If either of these two restrictions is violated, the earliest start time of the current operation will be pushed back until there are no violations of these two restrictions, i.e. the machine has completed its last operation and the completion of the on-going operation of the job is achieved.

The following is an example which demonstrates this. Suppose we have a 4×4 OSSP (4 jobs and 4 machines, 16 operations in total). The operating time of each operation in each job is given in Table 2.

From the table, we can see that operation $O_{1,1}$ has a operating time of 85 time units, likewise, operation $O_{4,4}$ has a operating time of 75 time units.

Table 2. Operating times of a 4 × 4 OSSP.

Machines	Job J_1	Job J_2	Job J_3	Job J_4
M_1	85	23	39	55
M_2	85	74	56	78
M_3	3	96	92	11
M_4	67	45	70	75

In this example, a valid representation of the solution would be a string containing any permutation of operations $O_{1,1}$ to $O_{4,4}$. Let's say we have a solution represented by:

$O_{1,1}$, $O_{1,2}$, $O_{2,2}$, $O_{3,4}$, $O_{2,1}$, $O_{4,1}$, $O_{2,3}$, $O_{2,4}$, $O_{1,3}$, $O_{3,3}$, $O_{4,2}$, $O_{3,1}$, $O_{4,4}$, $O_{1,4}$, $O_{3,2}$, $O_{4,3}$.

This represents the order in which the operations are scheduled.

We now examine this scheduling process:

- First to be scheduled is operation $O_{1,1}$, since machine M_1 is currently unoccupied and job j_1 has no on-going tasks, the earliest start time for $O_{1,1}$ is determined to be at time 0.
- The next operation to be scheduled is $O_{1,2}$, although machine M_2 is unoccupied at time 0, we note that $O_{1,2}$ and $O_{1,1}$ both belong to job j_1, hence the operation $O_{1,2}$ has to wait for operation $O_{1,1}$ to be completed before it can commence. Referring to Table 2, we see that the operating time for $O_{1,1}$ is 85 time units, which means that the earliest start time for $O_{1,2}$ will be at time 85.
- The next operation to be scheduled is $O_{2,2}$. We note that job j_2 has no on-going operations, however, since operation $O_{1,2}$ is scheduled before operation $O_{2,2}$, operation $O_{2,2}$ has to wait until operation $O_{1,2}$ is completed and machine M_2 is released before it can commence. In this case, the earliest start time for operation $O_{2,2}$ is the earliest start time for $O_{1,2}$ + the operating time of $O_{1,2}$, which is 85 + 85 = 170.
- The same applies to the rest of the operations following the order given in the solution string above. The end result is shown in Table 3.

As mentioned above, the objective of the OSSP is makespan minimization; hence the fitness function is the makespan of the schedule. From the example given above, we can see from Table 3 that the makespan of the schedule is 553 time units.

The encoding scheme employed for solving the OSSP required that there are no missing or duplicate operations on the solution chromosome. Hence, a normal single/multiple-point crossover technique cannot be used. Instead,

Table 3. 4×4 OSSP resultant operating times for schedule: $O_{1,1}$, $O_{1,2}$, $O_{2,2}$, $O_{3,4}$, $O_{2,1}$, $O_{4,1}$, $O_{2,3}$, $O_{2,4}$, $O_{1,3}$, $O_{3,3}$, $O_{4,2}$, $O_{3,1}$, $O_{4,4}$, $O_{1,4}$, $O_{3,2}$, $O_{4,3}$.

Machines	Job	Operation	Operating Time	Start Time	End Time
M_1	J_1	$O_{1,1}$	85	0	85
M_2	J_1	$O_{1,2}$	85	85	170
M_2	J_2	$O_{2,2}$	74	170	244
M_4	J_3	$O_{3,4}$	70	0	70
M_1	J_2	$O_{2,1}$	23	244	267
M_1	J_4	$O_{4,1}$	55	267	322
M_3	J_2	$O_{2,3}$	96	267	363
M_4	J_2	$O_{2,4}$	45	363	408
M_3	J_1	$O_{1,3}$	3	363	366
M_3	J_3	$O_{3,3}$	92	366	458
M_2	J_4	$O_{4,2}$	78	322	400
M_1	J_3	$O_{3,1}$	39	458	497
M_4	J_4	$O_{4,4}$	75	408	483
M_4	J_1	$O_{1,4}$	67	483	550
M_2	J_3	$O_{3,2}$	56	497	553
M_3	J_4	$O_{4,3}$	11	483	494

some other crossover methods which are suitable for sequencing problems are used. These include the *partially-mapped crossover* (PMX), the *order crossover* (OX) and the *cycle crossover* (CX) methods (see (Goldberg, 1989) and (Fox and McMahon, 1991)). There are more recent methods such as the *neighbourhood relationship crossover* (NRX) and the *meta-ordering crossover operator* (MOX) (see (Aşveren and Molitor, 1996)). We have chosen to use the PMX for our experiments because it is one of the most frequently used recombination technique for solving the Traveling Salesman Problem (another well known sequencing problem with encoding scheme that is very similar to the OSSP).

We give an example of PMX here:

Let A, B, C, D, E, F, G, H, I and J be the set of operations O_{ij}. Suppose we have two individuals, Parent 1 and Parent 2, with the following sequences,

Parent 1 = I, H, D, E, F, G, A, C, B, J

Parent 2 = H, G, A, B, C, J, I, E, D, F

Two crossover points are arbitrarily selected; suppose in this case, the crossover points are selected to be at position 3 and 7 (inclusive).

Table 4. Details of Test Cases for OSSP.

Case	Number of Jobs	Operations per Job
1	5	5
2	8	8
3	10	10
4	12	12
5	15	15
6	18	18
7	20	20
8	25	25

PMX works by finding the mappings of elements within the portion bounded by the two crossover points, between Parent 1 and Parent 2 and swapping the mapped elements in each parent to produce the child.

Suppose now we map Parent 2 to Parent 1. We see that element D is mapped to A; hence we swap element D and A in Parent 1 to get Parent 1′.

Parent 1 = I, H, **D**, E, F, G, A, C, B, J

Parent 2 = H, G, **A**, B, C, J, I, E, D, F

Parent 1′ = I, H, **A**, E, F, G, **D**, C, B, J

Next we see that element E is mapped to B; again we swap elements E and B in Parent 1′ to get:

Parent 1′ = I, H, A, **E**, F, G, D, C, B, J

Parent 2 = H, G, A, **B**, C, J, I, E, D, F

Parent 1″ = I, H, A, **B**, F, G, D, C, **E**, J

Following the same process, we swap F and C, G and J, and finally D and I to get a child individual:

Child 1 (from Parent 1) = D, H, A, B, C, J, I, F, E, G

Here, letters F, E and G are the result of swaps with letters between the crossover points. The same process can be carried out by mapping Parent 1 to Parent 2 and swapping elements in Parent 2 to obtain Child 2.

We experiment with 8 test cases with increasing magnitude and difficulty; the details of the test cases is shown in Table 4. The results of the head-to-head comparisons between the selection schemes for GA setting of 30 individuals evolving over 300 generations are presented in Figures 17 to 22 while the results for 50 individuals over 500 generations are presented in Figures 23 to 28. It should be noted that the objective of the OSSP is makespan minimization and hence solutions with low values are superior.

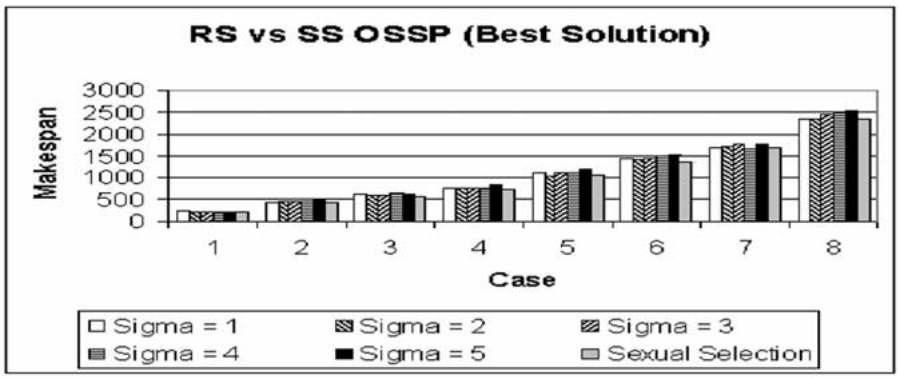

Figure 17. Best Results obtained by Roulette Selection compared against Sexual Selection (population = 30, generation = 300).

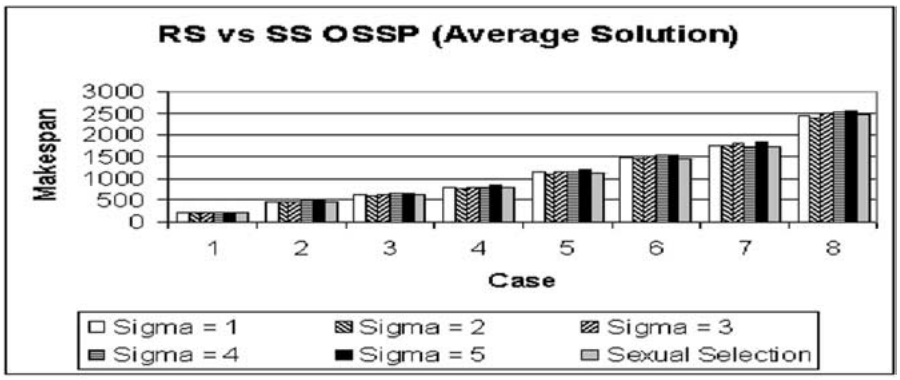

Figure 18. Average Results obtained by Roulette Selection compared against Sexual Selection (population = 30, generation = 300).

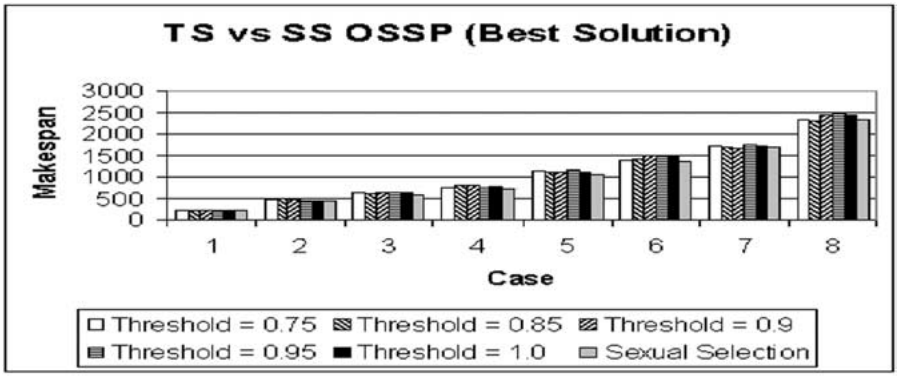

Figure 19. Best Results obtained by Tournament Selection compared against Sexual Selection (population = 30, generation = 300).

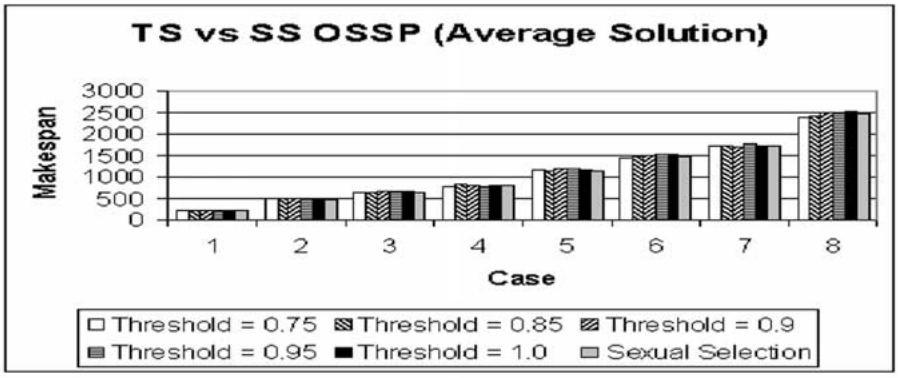

Figure 20. Average Results obtained by Tournament Selection compared against Sexual Selection (population = 30, generation = 300).

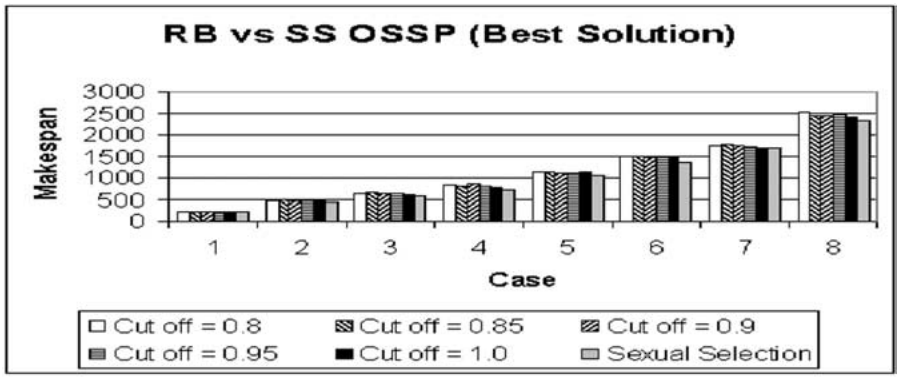

Figure 21. Best Results obtained by Rank-based Selection compared against Sexual Selection (population = 30, generation = 300).

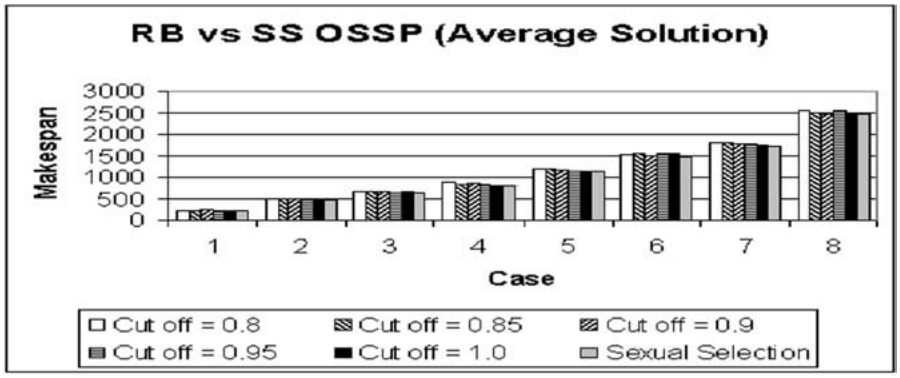

Figure 22. Average Results obtained by Rank-based Selection compared against Sexual Selection (population = 30, generation = 300).

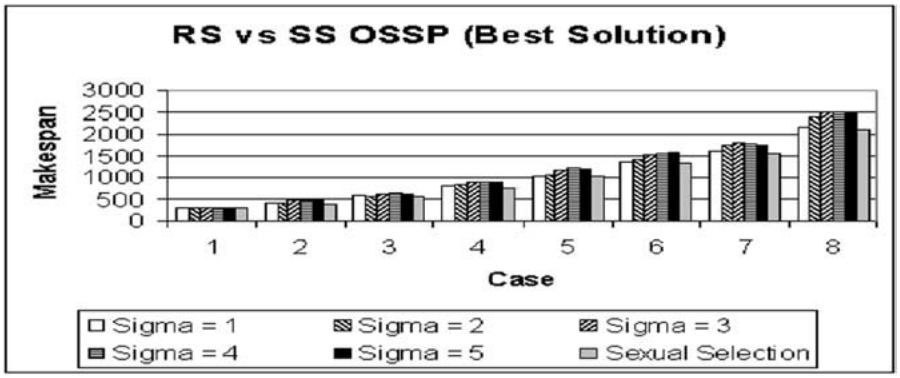

Figure 23. Best Results obtained by Roulette Selection compared against Sexual Selection (population = 50, generation = 500).

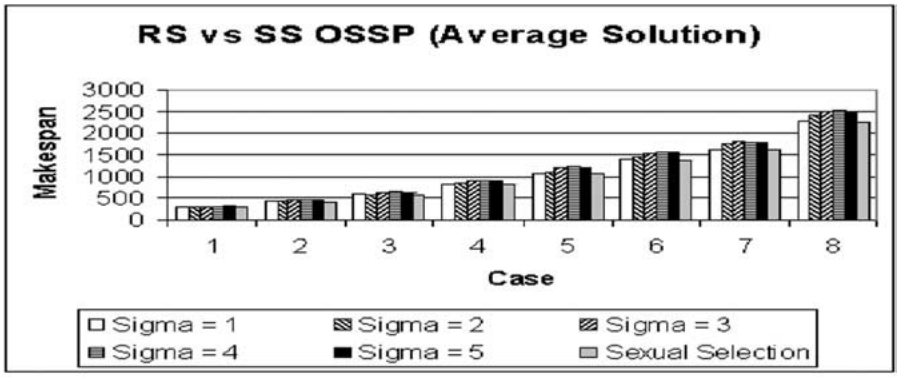

Figure 24. Average Results obtained by Roulette Selection compared against Sexual Selection (population = 50, generation = 500).

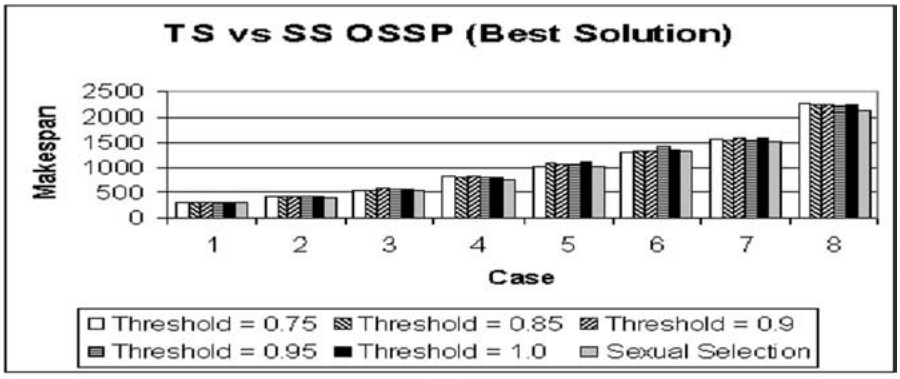

Figure 25. Best Results obtained by Tournament Selection compared against Sexual Selection (population = 50, generation = 500).

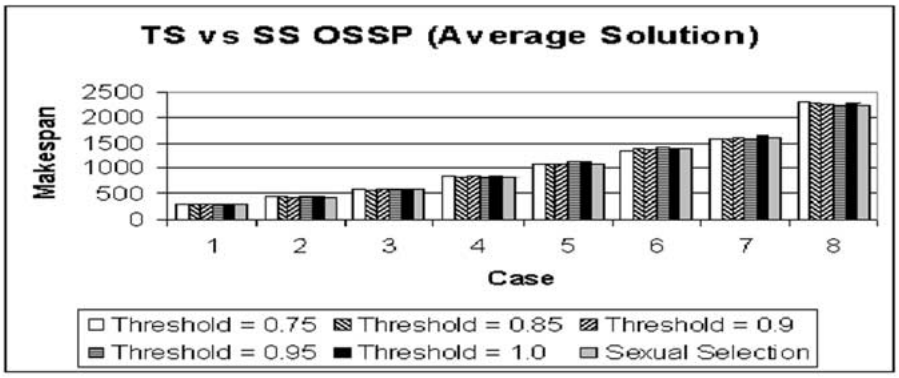

Figure 26. Average Results obtained by Tournament Selection compared against Sexual Selection (population = 50, generation = 500).

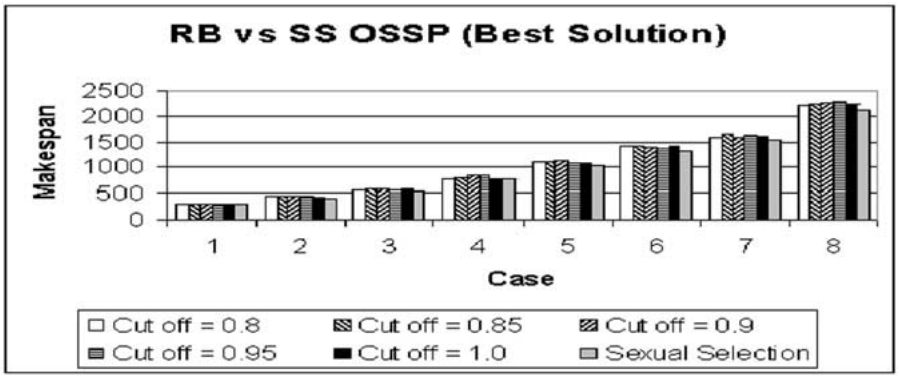

Figure 27. Best Results obtained by Rank-based Selection compared against Sexual Selection (population = 50, generation = 500).

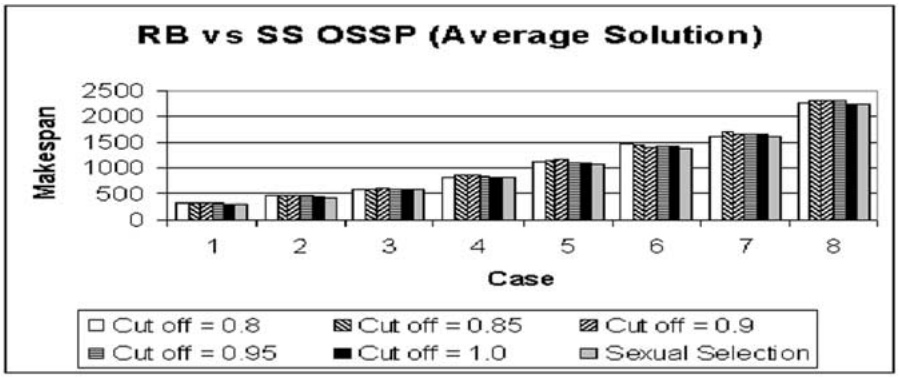

Figure 28. Average Results obtained by Rank-based Selection compared against Sexual Selection (population = 50, generation = 500).

8. The Job Shop Scheduling Problem

The Job Shop Scheduling Problem (JSSP) is a variant of the Open Shop Scheduling Problem where the main difference is that there exists a precedence relationship between operations belonging to the same job. It is a combinatorial problem of considerable industrial importance especially in manufacturing. Genetic Algorithms have been successfully used to generate good solutions for the JSSP (see (Falkenauer and Bouffouix, 1991), (Vaessens et al., 1994) and (Fang et al., 1993) for example).

Similar to the OSSP, the JSSP involves a set of jobs $J = j_1, j_2 \ldots j_n$. Each job j_i consists of m operations, O_{ij} $(j = 1 \ldots m)$, where O_{ij} have to be processed on machine M_j for a period of time represented by P_{ij} without preemption.

The basic assumptions of the JSSP is that each machine can only process one operation at any time, and each job can only be processed by one machine at any time; this means that jobs cannot have two or more operations processed in parallel. However, unlike the OSSP, operations in the JSSP have precedence relationships. For ease of representation, we assume that operation O_{ij} can only commence upon the completion of O_{ij-1} for all values of $j \geq 1$

As with OSSP, the objective function of the JSSP is also *makespan* minimization.

A large part of the representation for the OSSP can be reused for encoding the JSSP. The primary concern when solving the JSSP is also the start time allocated to each of the operations, $O_{1,1}$ to $O_{n,m}$. The representation of the problem is a string of length $m \times n$. The string will contain the sequence at which operations will be scheduled to their respective machines. Each operation will be scheduled to the respective machine at the *earliest possible time*.

Similar to the OSSP, the earliest possible time of any operation in the JSSP has two restrictions; firstly, the machine must be free, and secondly, the job to which the operation belongs to must not have any on-going operations. If either of the two restrictions is violated, the earliest start time of the current operation will be pushed back until there are no violations of the these restrictions, i.e. the machine has completed its last operation and the completion of the on-going operation for the job is achieved.

However, since the one of the main requirements for encoding the JSSP is to maintain the precedence relationship between each operations within each job, the encoding is done in two separate stages. The first stage determines the order in which jobs are scheduled, while the second stage determines the order in which operations are scheduled.

We illustrate with an example. Suppose we have a 4×4 JSSP (4 jobs and 4 machines, 16 operations in total).

To encode the JSSP, we will have to first encode the sequence in which jobs are scheduled. Since a valid schedule will contain the order in which all the 16 operations will be scheduled, each job will appear 4 times in the schedule. Hence to perform the first stage of the encoding, each job is randomly placed into the schedule 4 times.

Suppose we produce the following partial schedule:

$j_1, j_1, j_2, j_3, j_2, j_4, j_2, j_2, j_1, j_3, j_4, j_3, j_4, j_1, j_3, j_4$

This can be translated to produce a partial schedule which looks like:

$O_{1,*}, O_{1,*}, O_{2,*}, O_{3,*}, O_{2,*}, O_{4,*}, O_{2,*}, O_{2,*}, O_{1,*}, O_{3,*}, O_{4,*}, O_{3,*}, O_{4,*}, O_{1,*}, O_{3,*}, O_{4,*}$.

Where $'*'$ represents the operation number in each job which will be addressed in the second stage of the encoding.

Since we assume that the operations in each job precedes each other in the same order of their operation number, that is, operation 1 must be completed before operation 2 can commence and operation 2 must be completed before operation 3 can commence and so on. The second stage of the encoding is to simply fill up the operation numbers in this order.

Doing this, we will generate the following valid schedule:

$O_{1,1}, O_{1,2}, O_{2,1}, O_{3,1}, O_{2,2}, O_{4,1}, O_{2,3}, O_{2,4}, O_{1,3}, O_{3,2}, O_{4,2}, O_{3,3}, O_{4,3}, O_{1,4}, O_{3,4}, O_{4,4}$.

The process of actually scheduling the task and obtaining the makespan is exactly the same as for the OSSP.

Maintaining a precedence relationship among the operations in the schedule is of the utmost priority when performing recombination; hence, we have used a variant of the two-point crossover proposed in (Hartmann, 1998). The processes are illustrated in Figure 29.

Assume we have now selected two individuals (female and male) for recombination and let $P1$ and $P2$ be the two crossover points.

1. Operations from point 0 to $P1$ are duplicated directly from the female individual to the child individual.
2. The first $(P2 - P1)$ operations in the male that are not already in the child are then added to the child individual with their relative positions with each other preserved.
3. The remaining $n - P2$ operations in the child individual are taken from the female; again, only operations that have not appeared in the child are selected and their relative positions are also maintained.

Maintaining the relative positions of the operations as they appear in the parent list would ensure that the child solution generated does not violate the original precedence constraint (see (Hartmann. 1998) for proof).

Kai Song Goh and Andrew Lim and Brian Rodrigues

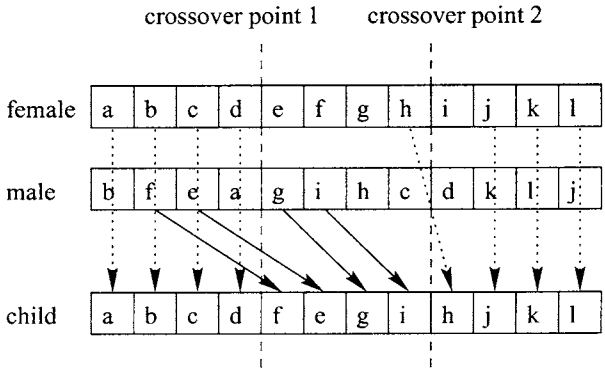

Figure 29. 2-point crossover replacing the portion of the female bounded by the 2 crossover points.

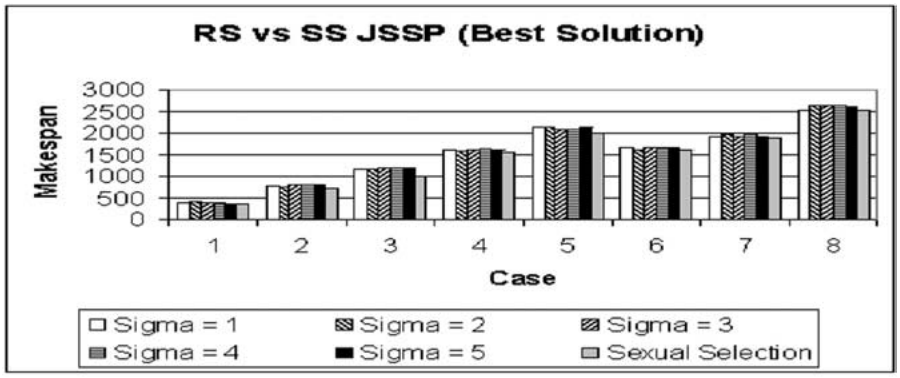

Figure 30. Best Results obtained by Roulette Selection compared against Sexual Selection (population = 30, generation = 300).

We have used the same test set as the one used for OSSP for our experiments for the JSSP. Once again, the objective is makespan minimization and the results for experiments with population size of 30 individuals evolving for 300 generations are presented in Figures 30 to 35 and that of 50 individuals for 500 generations are presented in Figures 36 to 41.

9. Conclusion

From our experiments, we have found that scaling definitely improves the results of Roulette Wheel Selection. It should be noted that Rank-based Selection even without the cutoff percentage p (see Section 3) is essentially

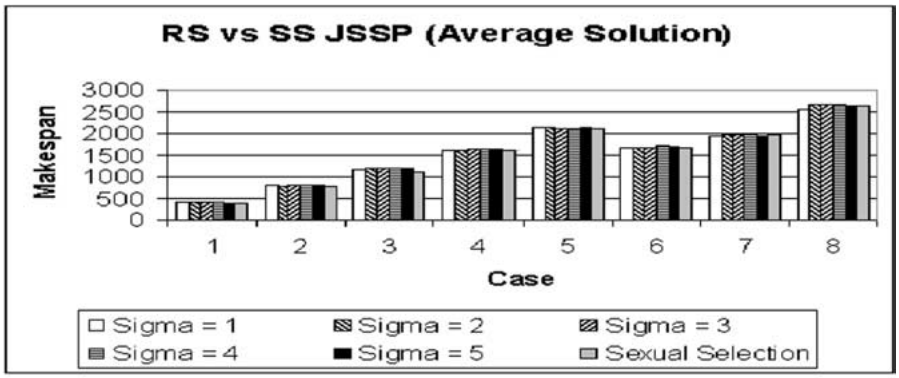

Figure 31. Average Results obtained by Roulette Selection compared against Sexual Selection (population = 30, generation = 300).

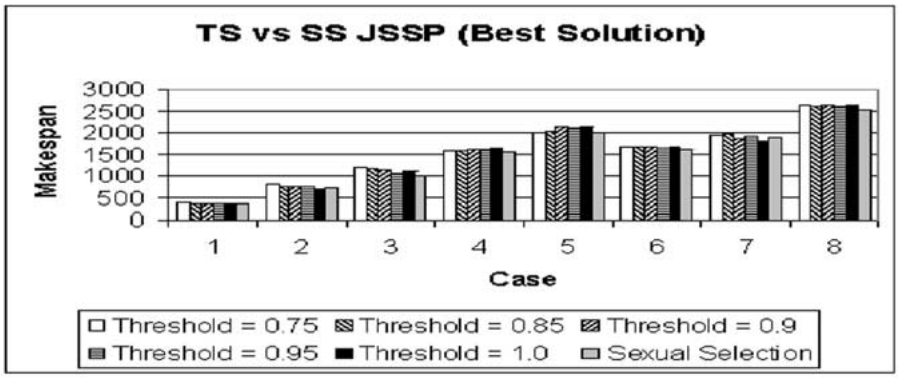

Figure 32. Best Results obtained by Tournament Selection compared against Sexual Selection (population = 30, generation = 300).

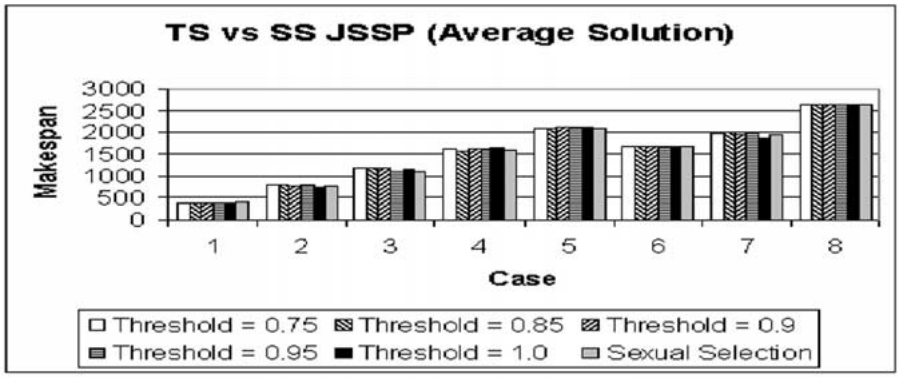

Figure 33. Average Results obtained by Tournament Selection compared against Sexual Selection (population=30, generation = 300).

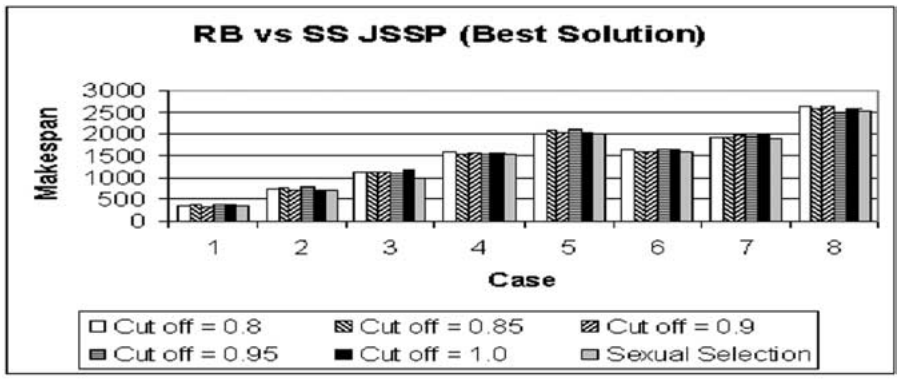

Figure 34. Best Results obtained by Rank-based Selection compared against Sexual Selection (population = 30, generation = 300).

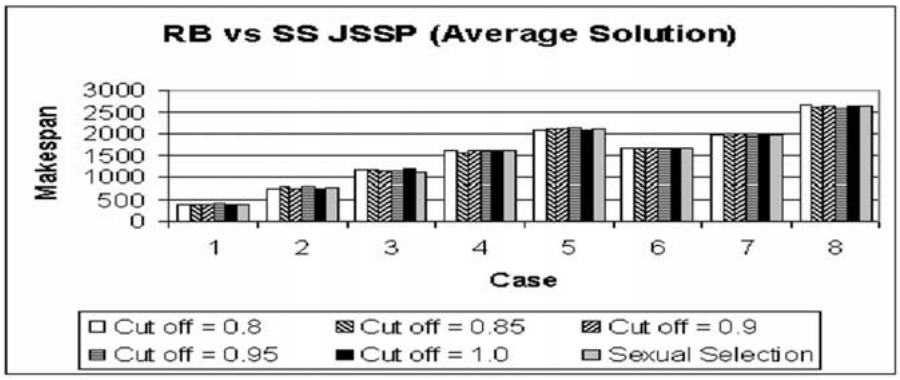

Figure 35. Average Results obtained by Rank-based Selection compared against Sexual Selection (population = 30, generation = 300).

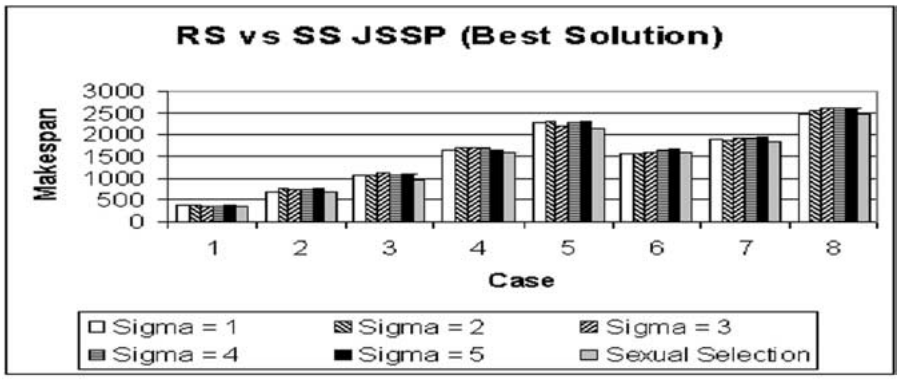

Figure 36. Best Results obtained by Roulette Selection compared against Sexual Selection (population = 50, generation = 500).

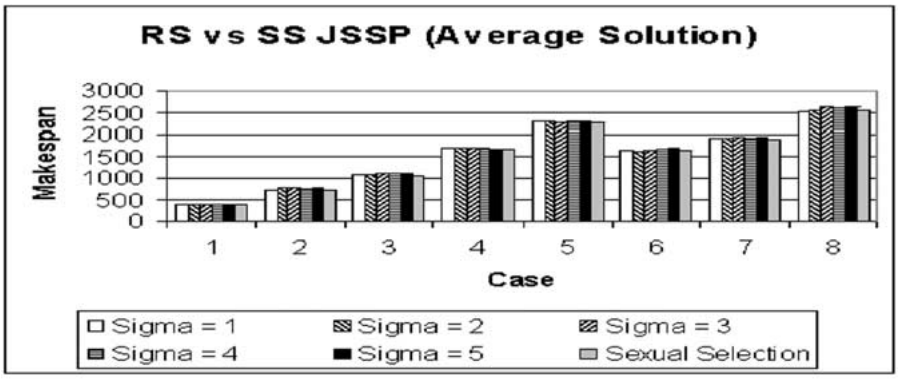

Figure 37. Average Results obtained by Roulette Selection compared against Sexual Selection (population = 50, generation = 500).

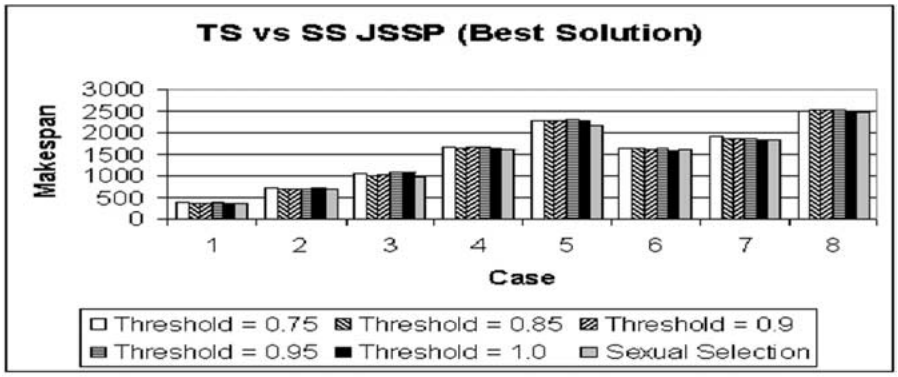

Figure 38. Best Results obtained by Tournament Selection compared against Sexual Selection (population = 50, generation = 500).

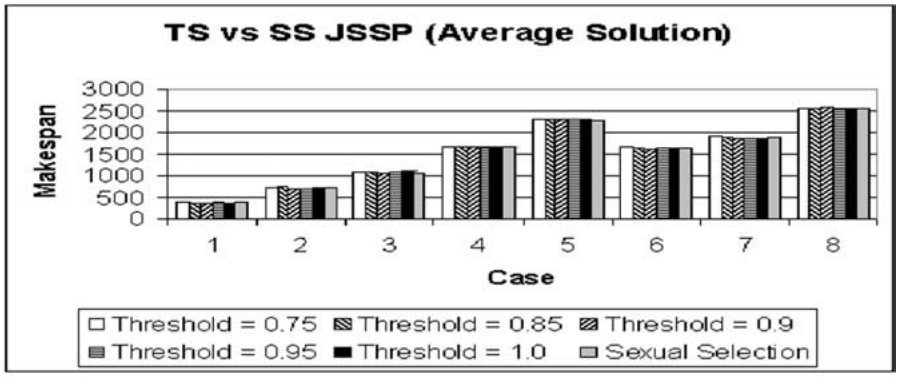

Figure 39. Average Results obtained by Tournament Selection compared against Sexual Selection (population = 50, generation = 500).

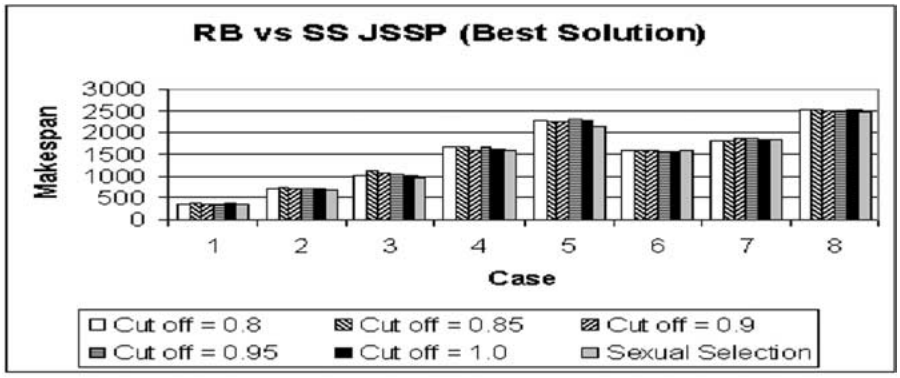

Figure 40. Best Results obtained by Rank-based Selection compared against Sexual Selection (population = 50, generation = 500).

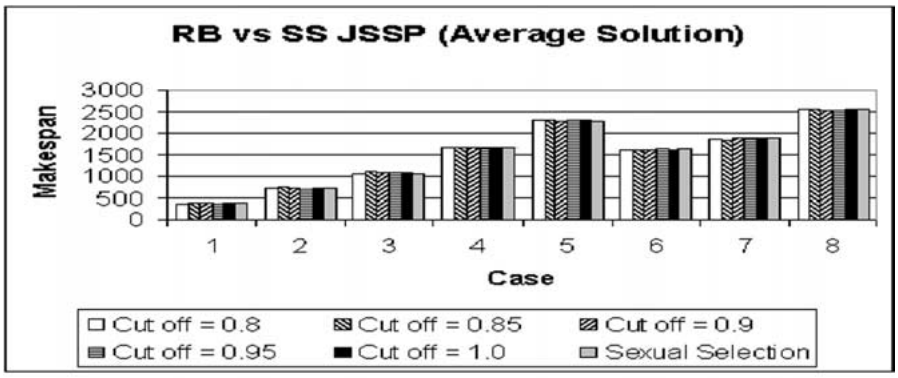

Figure 41. Average Results obtained by Rank-based Selection compared against Sexual Selection (population = 50, generation = 500).

a selection scheme with built-in fitness scaling. This could explain why in cases where no scaling is used, Rank-based Selection performs better then Roulette Wheel and Tournament Selection schemes. The new Sexual Selection Scheme proposed performs either on-par or better than Roulette Wheel selection on average when no fitness scaling is used. This is true for both the average case and for the best result in both problems. The new scheme also performs better on average when compared to Tournament Selection in the more difficult test cases when no scaling is used. This could be due to the over-bias to fitter solutions which is in the nature of the Tournament Selection problem.

In GA, the selection stage is generally not problem-dependent; hence selection schemes can be used to solve different problems with little or no modifications. We are confident that the new selection scheme as proposed

here will work as well with other optimization, scheduling and planning problems.

References

Aşveren, T. & Molitor, P. (1996). New Crossover Methods for Sequencing Problems. In: Voigt, H.-M., Ebeling, W., Rechenberg, I. & Schwefel, H.-P. (eds.) *Parallel Problem Solving from Nature – PPSN IV*, 290–299. Berlin, Springer.

Back, T. (1994). Selective Pressure in Evolutionary Algorithms: A Characterization of Selection Mechanisms. *1st IEEE Conference on Evolutionary Computing* **1**: 57–62.

Back, T. & Hoffmeister, F. (1991). Extended Selection Mechanisms in Genetic Algorithm. In: Belew, R. K. & Brooker, L. B. (eds.) *4th International Conference on Genetic Algorithm*, 92–99. San Mateo, CA.

Blickle, T. & Thiele, L. (1995). *A Comparison of Selection Schemes used in Genetic Algorithms*. TIK-Report No. 11 Computer Engineering and Communication Networks Lab, ETH, Zurich, Switzerland.

Darwin, C. (1888). *The Origin of Species by Means of Natural Selection, or The Preservation of Favoured Races in the Struggle for Life*. London: Murray, sixth edition.

Falkenauer, E. & Bouffouix, S. (1991). A Genetic Algorithm for Job Shop. *Proceedings 1991 IEEE International Conference on Robotics and Automation* **1**, 824–829. Sacramento, CA, IEEE Computer Society Press, Los Alamitos, CA.

Fang, H.-L., Ross, P. & Corne, D. (1993). A Promising Genetic Algorithm Approach to Job-Shop Scheduling, Rescheduling and Open-Shop Scheduling Problems. *Proceedings of the Fifth International Conference on Gentic Algorithms*, 375–382.

Fang, H.-L., Ross, P. & Corne, D. (1994). A Promising Hybrid GA/Heuristic Approach for Open–Shop Scheduling Problems. In: Cohn, A. (ed.) *11th European Conf. on Artificial Intelligence*, 590–594. Chichester, John Wiley and Sons, Ltd.

Fox, B. R. and McMahon, M. B. (1991). Genetic Operators for Sequencing Problems. *Foundation of Genetic Algorithms*, 284–300.

Goldberg, D. E. (1989). *Genetic Algorithms in Search, Optimization & Machine Learning*. Addison-Wesley Publishing Company.

Hartmann, S. (1998). A competitive genetic algorithm for resource-constrained project scheduling. *Naval Research Logistics* **45**.

Holland, J. H. (1975). *Adaption in Natural and Artificial Systems*. University of Michigan.

Khuri, S. & Miryala, S. R. (1999). Genetic Algorithms for Solving Open Shop Scheduling Problems. *Portuguese Conference on Artificial Intelligence*, 357–368.

Kolarov, K. (1995). The Role of Selection in Evolutionary Algorithm. *1995 IEEE International Conference on Evolutionary Computing*, 86–91. Perth.

Lis, J. & Eiben, A. E. (1996). A Multi-Sexual Genetic Algorithm for Multiobjective Optimization. In: Fukuda, T. & Furuhashi, T. (eds.). *Proceedings of the 1996 International Conference on Evolutionary Computation*, 59–64. Nagoya, Japan, IEEE.

Louis, S. J. & Xu, Z. (1996). Genetic algorithms for Open Shop Scheduling and Re-Scheduling. In: Cohen, M. E. & Hudson, D. L. (eds.) *ISCA 11th Int Conf on Computers and their Applications*, 99–102.

Matsui, K. (1999). New Selection Method to Improve the Population Diversity in Genetic Algorithm. *1999 IEEE International Conference on Systems, Man and Cybernetics* **1**: 625–630.

Mitchell, M., Forrest, S. & Holland, J. H. (1992). The Royal Road for Genetic Algorithms: Fitness landscapes and GA Performance. *Toward a Practice of Autonomous Systems: Proceedings of the First European Conference on Artificial Life*, 245–254. Cambridge, MA, MIT Press.

Mitchell, M., Holland, J. H. & Forrest, S. (1994). When will a Genetic Algorithm Outperform Hill Climbing. In: Cowan, J. D., Tesauro, G. & Alspector, J. (eds.) *Advances in Neural Information Processing Systems* **6**, 51–58. Morgan Kaufmann Publishers, Inc.

Muhlenbein, H. & Schlierkamp-Voosen, D. (1993). Analysis of Selection, Mutation and Recombination in Genetic Algorithms. *Neural Network World* **3**: 907–933.

Ratford, M., Tuson, A. & Thompson, H. (1997a). *Applying Sexual Selection as a Mechanism for Obtaining Multiple Distinct Solutions*. Presented at Emerging Technologies '97.

Ratford, M., Tuson, A. & Thompson, H. (1997b). The Single Chromosome's Guide To Dating. In: *Third International Conference On Artificial Neural Networks And Genetic Algorithms*.

Ronald, E. (1995). When Selection meets Seduction. *The Sixth International Conference on Genetic Algorithms*, 167–173. Pittsburg, USA, Morgan Kaufmann.

Vaessens, R., Aarts, E. & Lenstra, J. (1994). *Job-Shop Scheduling by Local Search*. COSOR Memorandum 94-05, Eindhoven University of Technology, Eindhoven, The Netherlands.

van Nimwegen, E., Crutchfield, J. P. & Mitchell, M. (1999). Statistical Dynamics of the Royal Road Genetic Algorithm. *Theoretical Computer Science* **229**(1): 41–102.