# Unbiased tournament selection

**Conference Paper** · June 2005

DOI: 10.1145/1068009.1068198 · Source: DBLP

**2 authors:**

Artem Sokolov
Harvard Medical School
**84** PUBLICATIONS **29,543** CITATIONS
SEE PROFILE

Darrell Whitley
Colorado State University
**390** PUBLICATIONS **22,801** CITATIONS
SEE PROFILE

# Unbiased Tournament Selection

Artem Sokolov
Department of Computer Science
Colorado State University
Fort Collins, CO 80523
sokolov@cs.colostate.edu

Darrell Whitley
Department of Computer Science
Colorado State University
Fort Collins, CO 80523
whitley@cs.colostate.edu

## ABSTRACT

Tournament selection is a popular form of selection which is commonly used with genetic algorithms, genetic programming and evolutionary programming. However, tournament selection introduces a sampling bias into the selection process. We review analytic results and present empirical evidence that shows this bias has a significant impact on search performance. We introduce two new forms of *unbiased tournament selection* that remove or reduce sampling bias in tournament selection.

## Categories and Subject Descriptors

I.2.8 [**Artificial Intelligence**]: Problem Solving, Control Methods, and Search; G.1.6 [**Numerical Analysis**]: Optimization—*Global Optimization*

## Keywords

Evolutionary Computation, Genetic Algorithms, Tournament Selection, Loss of Diversity

## 1. INTRODUCTION

Selection is an important aspect of evolutionary computation. It dictates what members of the current population affect the next generation. More fit individuals are generally given a higher chance to participate in the recombination process. The motivation is that highly fit members of a population possess good properties that, recombined in the right way, could lead to even better solutions.

The primary concern of all selection schemes is what's known as the *loss of diversity*. In a generational genetic algorithm (GA), not every individual makes it into the intermediate population. As a result, information encoded in the current population is not transferred into the next generation in its entirety. Loss of diversity has been measured and analyzed for a number of popular selection algorithms([6], [3], [4], and [7]).

Of particular interest is tournament selection, which is commonly used with genetic algorithms, genetic programming and evolutionary programming. As pointed out by Poli [8], the loss of diversity in tournament selection is caused by two factors. He distinguishes between the notions of not being *sampled* and not being *selected*. The first holds true about an individual that does not get to participate in a tournament. The second refers to an individual losing a tournament and, thus, not making it into the intermediate population. Poli notes that the objective function is unnecessarily evaluated on individuals that don't affect the next generation and suggests ways to eliminate this computation [8].

This paper takes a different approach. We propose a selection scheme that eliminates the loss of diversity in tournament selection due to individuals not being sampled. Our algorithm is based on reducing variance in the number of times a particular individual is picked to participate in a tournament. We demonstrate that using our selection algorithm yields better results than the traditional tournament selection when used in a generational genetic algorithm.

We also introduce a new scheduling problem into the literature: Radar Scheduling. The problem is a variation of resource-constrained project scheduling, as defined in [10]. The implementation presented in this papers consists of a greedy scheduler that constructs and evaluates a schedule based on some given permutation of tasks. A GA works with permutations and uses the scheduler to calculate fitness of every solution.

The paper is organized in the following way. In Section 2 we briefly overview some of the popular selection schemes and introduce our novel algorithm. Applications and empirical results are presented in Section 3. Section 4 consists of additional elaboration on the observed behavior. Finally, Section 5 summarizes the work done and describes possible directions to be taken for further analysis.

## 2. SELECTION METHODS

We present an overview of the three commonly used selection algorithms (a very comprehensive analysis can be found in [3]); we also review factors that contribute to the loss of diversity in each algorithm. We then introduce a description of a new algorithm, which we call `Unbiased Tournament Selection`.

## 2.1 Fitness Proportional Selection

Fitness proportional (also known as "Roulette-Wheel") selection is different from the other algorithms discussed in this paper in that, as the name suggests, individuals are being drawn with probabilities directly proportional to their fitness values. This is in contrast to using the *rank* of an individual to assign these probabilities. The major implication of this difference is that fitness proportional selection is more sensitive to outlying fitness values. An exceptionally fit individual is likely to dominate the intermediate population, leading to a decrease in diversity of the offsprings.

A very common way of implementing fitness proportional selection is via stochastic universal sampling[2]. The method places evenly spaced markers against an interval divided into regions corresponding to individuals in the population. Larger regions are allocated to the more fit individuals. The offset for the first marker is chosen from $[0, 1]$ uniformly at random and all markers are then used to pull out the corresponding individuals for recombination (Figure 1).

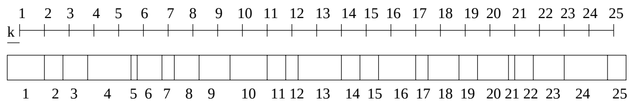

**Figure 1: Stochastic universal sampling**

In addition to the above implementation, we also applied a shift factor to all fitness values to ensure that the lowest fitness value is equal to 1. This was done prior to running the selection algorithm to normalize the region sizes. Note that the probability distribution of individuals getting selected for recombination is more uniform when fitness values lie in $[a, b]$ rather than $[1, b - a + 1]$ whenever $a >> 1$ [5].

Note that universal stochastic sampling does not guarantee that all individuals get selected. For instance, in Figure 1 the fifth individual is not hit with a marker and does not make it into the intermediate population.

The remaining three algorithms make use of an individual's rank in the corresponding population to compute the probability of that individual being selected for recombination.

## 2.2 Rank-Based Selection

Whitley proposed the following implementation of rank-based selection[14]:

$$j = \left\lfloor \frac{N}{2(c-1)} \left( c - \sqrt{c^2 - 4(c-1)\chi} \right) \right\rfloor$$

where $\chi$ is a random value drawn uniformly from $[0, 1)$ and $N$ is the number of individuals in the population. The returned rank $j$ can be used to index into the population sorted by fitness value, with $j = 0$ and $j = N - 1$ representing the most and least fit individuals respectively. The computation is performed to select one individual at a time and needs to be carried out $N$ number of times (drawing a new value of $\chi$ every time) when composing the intermediate population. This is where the loss of diversity is experienced because, depending on the values of $\chi$, some individuals might not be sampled and, thus, not get selected. This implementation was introduced to be used with one-at-a-time reproduction in the GENITOR steady state genetic algorithm.

An alternative implementation of ranked based selection resembles fitness proportional selection. The same stochastic universal sampling is used but regions are allocated proportionally to the rank of individuals, rather than the fitness values.

The variable of interest in the formula above is c, which is called selective pressure and usually chosen to be in $(1, 2]$. The formula itself is used for *linear* ranking, where selective pressure, in effect, controls the slope of the line. Linear ranking assigns the probability of an individual getting selected to be linearly dependent on the position of that individual in the population sorted by rank. As $c$ approaches 1, all individuals have equal chance of being selected for recombination. On the other hand, a selective pressure value of 2 constitutes the probability of selecting the most fit individual being two times greater than that of selecting an individual with median fitness value. In this case, the probability of selecting the worst individual is zero.

We have fixed the value of selective pressure at $c = 2$ throughout all the experiments in this paper. This choice allows for better comparison of performance since the other two rank based selection algorithms will exhibit similar probability distributions.

## 2.3 Random Tournament Selection

In this paper, we label the traditional tournament selection scheme as *random tournament selection*. Perhaps the two most appealing aspects of using this algorithm is the ease of implementation and a large potential for parallelism. Given a population, $t$ individuals get picked uniformly at random and the best is chosen for recombination. The process is repeated the number of times necessary to achieve the desired size of the intermediate population.

The algorithm is embarrassingly parallel as each tournament is independent. In the best-case scenario — if one had the same number of processors as the population size — all tournaments could be run simultaneously. This would involve nothing more than each processor sampling the population $t$ times and selecting the best of those individuals.

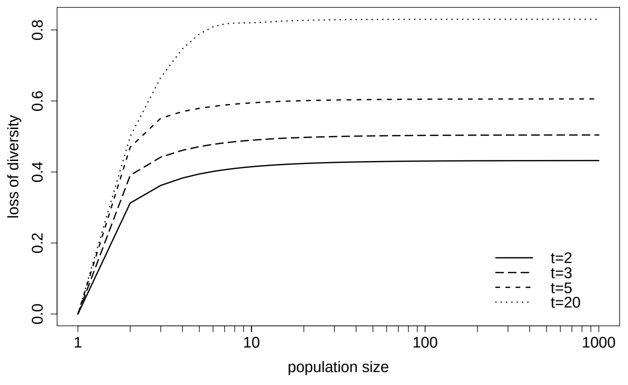

**Figure 2: Loss of diversity for different values of tournament size $t$**

The number of individuals that get picked, $t$, affects the probability distribution in a way similar to selective pressure in a rank-based algorithm. Larger values correspond to higher probability of the most fit individual being selected

```
     Population Ranks: 1 2 3 4 5 6 7 8

Sampled Pairs: 5 5 4 3 2 6 6 5 2 4 2 3 8 4 3 6
Tournament:    \_/ \_/ \_/ \_/ \_/ \_/ \_/ \_/
                |   |   |   |   |   |   |   |
Winners:        5   3   2   5   2   2   4   3

Not Sampled:  1, 7          Not Selected: 6, 8
```

**Figure 3: A random tournament scenario. More fit individuals are represented by lower rank values. Note the difference between "not sampled" and "not selected"**

relative to the rest of the population. We fixed $t$ to be 2 because higher selective pressure will flood the intermediate population with replicas of highly fit individuals and additional unnecessary loss of diversity will be experienced.

Motoki [7] computes the expected loss of diversity in tournament selection to be:

$$D_T(t, N) = \frac{1}{N} \sum_{k=1}^{N} \left(1 - \frac{k^t - (k-1)^t}{N^t}\right)^N$$

where N is the population size and $t$ is the tournament size. As can be seen in Figure 2, larger values of $t$ do indeed lead to higher expected loss of diversity. Blickle and Thiele[3] demonstrate that large values of selective pressure lead to similar problems in rank-based selection as well.

Note that the expected loss of diversity grows very rapidly with the population size and quickly levels off to being nearly horizontal at $N = 8$. Since populations that small are rather uncommon, this graph demonstrates that large values $t$ present a serious issue. This further justifies our choice of tournament size $t = 2$.

As Poli notes, there are two factors that lead to the loss of diversity in random tournament selection. Some individuals might not get *sampled* to participate in a tournament at all. Other individuals might not be *selected* for the intermediate population because they lost in a tournament. Figure 3 shows an example of random tournament selection applied to a population of size eight. The most and least fit individuals are represented by numbers 1 and 8 respectively. Individuals 1 and 7 are not *sampled*, while individuals 6 and 8 are not *selected*.

Note that the notions of not being sampled and not being selected are indistinguishable in fitness proportional and rank-based selection schemes. This is due to a single decision process involved in each of the two algorithms. For instance, rank-based selection algorithm simply samples one individual at a time. In each case, the same individual is selected for recombination. Random tournament selection, on the other hand, involves two decision processes: who gets to participate in a tournament and who gets added to the intermediate population.

Poli calculates that the number of individuals neglected in the first decision if two offsprings are produced by recombination is $N(1-1/N)^{tN}$; he also presents a figure (Figure 4 in this paper) that shows the percentage of population ignored for difference choices of parameter $t$.

Two interacting effects can be observed in Figure 4 and Figure 2. When $t = 2$ and the population is at least 10, approximately 40% of the population is lost due to selection

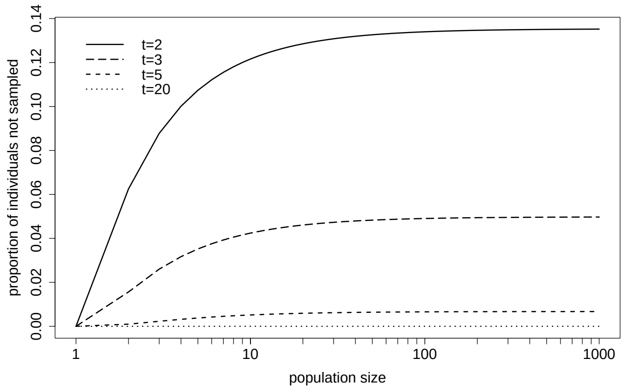

**Figure 4: Percentage of individuals not sampled for random tournament with different tournament size values**

effects, while close to 13% percent of individuals are lost due to a failure to sample these individual. As the tournament size increases, so does the expected loss of diversity; however, the fraction of that loss due to individuals not being sampled decreases. However, larger tournaments size means that a smaller portion of the population actually contributes to genetic diversity–making the search increasingly greedy in nature. We propose a tournament selection algorithm that eliminates loss of diversity related to the failure to sample; its impact is of particular significance for smaller values of $t$.

### 2.3.1 *Selective Bias less than 2.0*

We can define selective pressure as the ratio between the probabilities of selecting the best individual and the individual with median fitness value. For rank-based selection, selective pressure is the parameter $c$ in the given implementation. Selective pressure in tournament selection is controlled through the tournament size that, unlike $c$, cannot be lower than two. However, tournament selection can be implemented with selective pressure less than 2.0. Two individuals are compared, but the best is chosen with a probability greater than 0.5 but less than 1.0; if $p_s$ is the probability of keeping the best individual, the selective pressure is $2p_s$. However, while a lower selection pressure can reduce the loss of diversity due to selection, it does nothing to change the failure to sample.

## 2.4 Unbiased Tournament Selection

The main motivation behind our novel algorithm is to make participation in a tournament more "fair" through the use of a variance reduction mechanism. Unbiased tournament selection scheme introduces additional control over what individuals get to participate in each tournament by completely or partially eliminating uniform population sampling.

In essence, unbiased tournament selection lines up two different permutations of the population and performs a pairwise comparison. An individual with higher fitness, or lower objective function value, is selected for recombination from each pair. The most straightforward way of implementing this algorithm is the following:

```
Population objective function values:
     1.5, 2.3, 15.6, 3.4, 7.8

Unbiased Tournament Selection:
   i    P[i]        f(i)    f(P[i])       I[i]
  ----------------------------------------------
   1     5   <->    1.5       7.8    -->    1
   2     3   <->    2.3      15.6    -->    2
   3     4   <->   15.6       3.4    -->    4
   4     1   <->    3.4       1.5    -->    1
   5     2   <->    7.8       2.3    -->    2
```

**Figure 5: Example of unbiased tournament selection. A more fit individual is characterized by lower objective function value.**

1. Construct $P$ to be a random permutation of indices into the population, such that $P[i] \neq i,\ \forall i$.
2. The intermediate population $I$ is chosen using tournament selection. Indices are given by $I[i] = P[i]$ if $f(P[i]) < f(i)$, and $I[i] = i$ otherwise. (We let $f(j)$ return the objective function value of $j^{th}$ individual.)

Constructing a random permutation in Step 1 can be done by uniformly sampling the population indices without replacement. The constraint $P[i] \neq i$ is easily enforced by withholding $i^{th}$ individual from participating in the sampling procedure. There are not enough degrees of freedom to guarantee that this constraint holds for the last element of the permutation, but a violation can be fixed by swapping the last element with its immediate predecessor.

Figure 5 demonstrates application of this algorithm to a population of size five. Note that each individual gets two chances to participate in a tournament. This fact eliminates the potential of better individuals not getting chosen for recombination, i.e. the bias that was present in Random Tournament selection.

The best individual will get selected *exactly* twice, the worst is never selected. These values are only expected in random tournament selection. Thus, we have effectively reduced the variance for these to zero.

The expected number of times an individual with median fitness getting selected is still 1 but this is very much in line with our previous discussion on probability distribution of other selection algorithms.

## 2.5 Parallel Unbiased Tournament Selection

Unbiased tournament selection completely eliminates random population sampling. However, the inherent parallelism present in the original random tournament selection is lost. Since each tournament is no longer independent, there is no efficient way to run them in parallel. We propose an alternative implementation that sacrifices some selection consistency to allow for tournament independence.

Assume we have $n$ processors enumerated $1, 2, ...n$, where $n$ is the population size. For each processor $i$:

1. Sample $k$ uniformly from $[1, n]$
2. Add individual $i$ into the intermediate population if $f(i) < f(k)$. Add individual $k$ otherwise. (Again, we let $f(j)$ return the objective function value of $j^{th}$ individual.)

```
Parent 1:       a  b  c  d  e  f
Cross Points:      *     *  *
Parent 2:       e  c  a  b  f  d
Child:          a  e  c  b  d  f
```

**Figure 6: Syswerda's order crossover operator**

Note that we no longer enforce $i \neq k$. The sole intent of using this constraint in the first place was to ensure that each individual participates in a tournament twice. Since no such guarantee is available in the parallel case, the constraint was deemed unneeded as it only provided additional overhead.

The second implementation is only a partial elimination of uniform population sampling. We still sample the population once per tournament, but we are, at the same time, guaranteed that everybody will participate at least once. Overall we have removed the loss of diversity due to individuals not being sampled.

We have used the first implementation for empirical comparison because of our interest in the variance reduction. The parallel case does not offer the nice property that the best individual will be selected exactly twice (as well as the worst individual not getting selected at all) and is of less interest. In Section 4, we revisit the idea of variance reduction by constructing graphs of how many times an individual gets selected by each selection scheme.

## 3. APPLICATIONS AND RESULTS

We ran a generational GA with each selection algorithm on three problems: one with permutation-based solution representation and two under bit encoding. As dictated by the solution encoding of each problem, either Syswerda's order crossover[11] or single point crossover[12] operators were used. The former is a common operator for recombining chromosomes with permutation representations. A simple example is shown in Figure 6. Several cross points along the first string are chosen uniformly at random. The corresponding characters $b$, $d$, and $e$ are reordered to match their relative position in the second string (i.e. $e$ comes before $b$, and $b$ comes before $d$). Note that the position of all other characters ($a$, $c$, and $f$) is unaffected.

No mutation was done to allow for more careful performance assessment. For each problem, we ran the GA for 30 generations keeping the best individual from the previous population at each iteration (elitist strategy). The most fit individual is reported at the end of a run. Relevant statistics after performing 30 runs with each selection algorithm are presented in the tables below. The particular focus, of course, is made on comparing the two tournament algorithms.

All problems examined in this paper are minimization problems. Therefore, we will refer to individuals with lower values of an objective function as being more fit.

### 3.1 Radar Scheduling Problem

This particular problem can be defined as resource-constrained project scheduling (RCPS) without ordering constraints. The following model was used in this paper. Suppose we want to track a number of objects in the Earth's orbit. We have a number of fixed-length, periodic time frames for each object, corresponding to when it can be picked up by the radar. The length of each task is assumed smaller or equal to the length of the corresponding frames. We impose a constraint that tracking must be completed within a single time frame (i.e. a task is not interruptible) and that each task needs to be scheduled only once. A single resource — radar power — has the capacity to allow for tracking of multiple objects at the same time.

The problem is constructed by implementing a greedy scheduler, which would take a permutation of enumerated tasks and produce and evaluate a schedule. The scheduler places each task in a permutation in the first feasible spot that has enough resources to support the job throughout its duration. If no such spot exists, the task is discarded and a penalty is incurred. The goal is to minimize this penalty through a manipulation of task ordering in a permutation.

We ran a genetic algorithm with each selection method for 30 trials. The problem instance included 150 tasks with randomly generated properties. The radar is assumed to have enough power to track up to 5 objects simultaneously. (Some of the settings we have used are based on knowledge of real tracking systems; other settings were chosen arbitrarily). The following penalty distribution was used:

| Job Priority | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| # of Jobs | 3 | 10 | 17 | 40 | 80 |
| Penalty | 1500 | 200 | 30 | 5 | 1 |

Results are shown in table 1. The best objective value obtained with each selection algorithm is displayed in the first column. The average performance across thirty trials is presented in the next two columns. The last column displays the results of performing a pairwise $t$-test to compare all selection algorithms to unbiased tournament. A $p$-value of less than 0.05 indicates statistically significant difference in performance. Although unbiased tournament outperforms other selection algorithms, it does so with statistical significance only for the smaller two values of the population size.

Also of interest is the fact that a GA with fitness proportional selection performs well for a small population size but loses its advantage as the population size grows and other selection algorithms gain a higher increase in performance. A possible explanation for this behavior is that the loss of diversity experienced by fitness proportional selection grows faster with the population size than in the case of other selection schemes. As we will see in Section 4, fitness proportional selection has a different sampling distribution that puts more selective pressure on highly fit individuals. This likely leads to a larger portion of the population being discarded and justifies our earlier choices of selective pressure for rank based algorithms.

## 3.2 Problems Under Bit Encoding

For assessing performance of selection algorithms on problems with bit representation we took two well-known evaluation functions: Schwefel [9] and Rana (F102 in [13]). We borrowed the following expansion method from [15] to scale these functions up to ten dimensions:

$$f(x) = \frac{\sum_{i=1}^{\lfloor n/2 \rfloor} f(x_{2i-1}, x_{2i}) + \sum_{i=1}^{\lfloor (n-1)/2 \rfloor} f(x_{2i+1}, x_{2i})}{n-1}$$

where n denotes the number of dimensions.

We used 10 bits of precision for each dimension and sin-

| POPSIZE=20 | Best | Mean | St.D | $p$-value |
|---|---|---|---|---|
| Fitness Prop. | 159.00 | 510.30 | 209.41 | 0.084 |
| Rank-Based | 225.00 | 558.47 | 252.23 | 0.019 |
| Random Tourn. | 205.00 | 573.20 | 188.80 | 0.002 |
| Unbiased Tourn. | 228.00 | 425.37 | 161.80 | 1.000 |

| POPSIZE=100 | Best | Mean | St.D | $p$-value |
|---|---|---|---|---|
| Fitness Prop. | 189.00 | 273.93 | 44.15 | 0.000 |
| Rank-Based | 135.00 | 192.97 | 34.65 | 0.019 |
| Random Tourn. | 124.00 | 190.83 | 30.92 | 0.025 |
| Unbiased Tourn. | 118.00 | 173.57 | 27.16 | 1.000 |

| POPSIZE=200 | Best | Mean | St.D | $p$-value |
|---|---|---|---|---|
| Fitness Prop. | 198.00 | 277.17 | 42.82 | 0.000 |
| Rank-Based | 122.00 | 162.57 | 27.56 | 0.834 |
| Random Tourn. | 126.00 | 160.77 | 20.38 | 0.944 |
| Unbiased Tourn. | 110.00 | 161.17 | 23.63 | 1.000 |

**Table 1: A Radar Scheduling Problem. Presented values were computed over 30 trials. The goal is to minimize the total penalty value for unscheduled tasks.**

| POPSIZE=20 | Best | Mean | St.D | $p$-value |
|---|---|---|---|---|
| Fitness Prop. | -379.11 | -289.79 | 40.11 | 0.880 |
| Rank-Based | -347.50 | -275.14 | 38.36 | 0.270 |
| Random Tourn. | -344.13 | -254.85 | 51.54 | 0.015 |
| Unbiased Tourn. | -386.00 | -288.01 | 50.34 | 1.000 |

| POPSIZE=100 | Best | Mean | St.D | $p$-value |
|---|---|---|---|---|
| Fitness Prop. | -453.82 | -409.31 | 21.15 | 0.012 |
| Rank-Based | -437.68 | -399.78 | 21.43 | 0.000 |
| Random Tourn. | -439.43 | -395.59 | 25.44 | 0.000 |
| Unbiased Tourn. | -456.32 | -423.29 | 20.70 | 1.000 |

| POPSIZE=200 | Best | Mean | St.D | $p$-value |
|---|---|---|---|---|
| Fitness Prop. | -457.37 | -429.86 | 14.81 | 0.000 |
| Rank-Based | -464.84 | -438.29 | 18.16 | 0.185 |
| Random Tourn. | -465.06 | -432.24 | 17.56 | 0.006 |
| Unbiased Tourn. | -476.85 | -443.87 | 13.70 | 1.000 |

**Table 2: Rana 10D. Statistics were computed across 30 runs. Global optimal is -511.7**

| POPSIZE=20 | Best | Mean | St.D | $p$-value |
|---|---|---|---|---|
| Fitness Prop. | -3163.4 | -2564.0 | 342.2 | 0.207 |
| Rank-Based | -3338.4 | -2324.8 | 489.2 | 0.303 |
| Random Tourn. | -3015.2 | -2269.4 | 314.8 | 0.062 |
| Unbiased Tourn. | -3255.5 | -2443.4 | 388.2 | 1.000 |

| POPSIZE=100 | Best | Mean | St.D | $p$-value |
|---|---|---|---|---|
| Fitness Prop. | -4115.8 | -3825.5 | 152.8 | 0.004 |
| Rank-Based | -4015.7 | -3762.4 | 142.2 | 0.000 |
| Random Tourn. | -4038.4 | -3720.1 | 186.3 | 0.000 |
| Unbiased Tourn. | -4131.1 | -3928.5 | 108.0 | 1.000 |

| POPSIZE=200 | Best | Mean | St.D | $p$-value |
|---|---|---|---|---|
| Fitness Prop. | -4136.4 | -4037.0 | 69.3 | 0.000 |
| Rank-Based | -4178.7 | -4020.2 | 111.4 | 0.000 |
| Random Tourn. | -4165.3 | -4044.2 | 91.8 | 0.002 |
| Unbiased Tourn. | -4183.7 | -4110.6 | 63.4 | 1.000 |

**Table 3: Schwefel 10D. Statistics were computed across 30 runs. Global optimum is -4189.8**

gle point crossover operator. All other parameters were kept the same (i.e. as in our experiments with permutation-based problems). Results for Rana and Schwefel are shown in Table 2 and Table 3 respectively. It can be seen that unbiased tournament outperforms all other selection algorithms with statistical significance for a population size of 100. Fitness Proportional selection again proves to be a competitive algorithm for smaller population size.

From these two tables we can clearly see the advantage of using unbiased tournament over the traditional random tournament selection scheme. A GA with the former selection algorithm was able to locate better solutions in both problems for all population sizes.

During our experiments with bitstring problems we have clearly experienced the loss of diversity. After 30 generations, we found populations of size 20 to consist entirely of copies of a single individual. The diversity of larger populations was also found to be greatly reduced often leaving only a few distinct individuals.

To a large extent, the loss of diversity in problems with permutation solution encoding was significantly lower. This, perhaps, explains why larger population sizes were yielding consistent performance across all selection algorithms in our permutation problem. Maintaining a diverse population leads to all selection algorithms finding similarly good solutions and as the population size is increased they begin to show identical performance. Since a genetic algorithm was suffering from premature convergence with bitstring problems, we saw more difference in performance when a good selection scheme was used.

## 4. BIAS AND DIVERSITY

In this section we explore two questions. First, to what degree does the behavior of algorithms differ from expected when associated with a selection bias of 2.0? Second, since the ultimate problem is loss of diversity, does explicit enforcement of diversity reduce differences in performance that have been observed among selection algorithms?

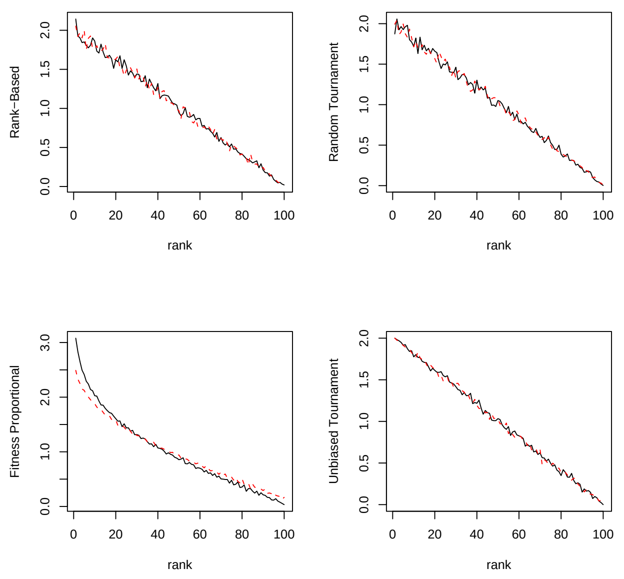

**Figure 7: Rana10D with POPSIZE = 100. Solid line ( — ) represents the number of times an individual got selected during the first 15 generations. Similarly, the dashed line ( - - ) shows how many times an individual was selected in the last 15 generations. All values are averages over 30 trials.**

### 4.1 Deviation From Expected Probability

In Section 2, we showed that rank-based, random tournament, and unbiased tournament selection algorithms exhibit the property where the probability that the best individual will get selected for recombination is two times higher than the probability of a similar event happening to an individual with median fitness value. We have investigated how many times a particular individual does actually get selected for recombination. Figure 7 shows a typical picture that was seen throughout our experiments. The number of times an individual was selected is plotted against that individual's rank in a population. The solid and dashed lines refer to the first and the second halves of the search respectively. Notice that unbiased tournament selection comes the closest to the expected probability distribution: if we were to compute the expected value for the number of times every individual gets selected for recombination, we should see a straight line with slope equal to 2. Perhaps the most important observation

| | L1 norm | | L2 norm | |
|---|---|---|---|---|
| | Mean | St.D | Mean | St.D |
| Rank-Based | 13.434 | 0.796 | 1.7714 | 0.1177 |
| Random Tourn. | 13.744 | 1.077 | 1.7951 | 0.1169 |
| Unbiased Tourn. | 7.954 | 0.591 | 1.0263 | 0.0797 |

**Table 4: Rana10D with POPSIZE = 100. Deviation from expected probability distribution: L1 and L2 norms. Results are averaged over 30 trials.**

here is the significantly smaller amount of noise present in unbiased tournament near the best and worst individuals.

Table 4 presents deviations from the straight line summed up over all individuals. Fitness proportional algorithm was omitted, since the probability distribution is largely different from the other three algorithms. Notice that the deviation values are significantly smaller for Unbiased Tournament. This indicates success of our variance reduction mechanism. Note also that, since all algorithms base their computation on rank rather than fitness directly, the presented values would not vary from problem to problem, as long as population size remains the same. In fact, we observed that the values remained more or less the same even as the population size varied between 20, 100, and 200.

We would like to reiterate the fact that fitness proportional selection applies higher selective pressure to more fit individuals than other selection schemes. This is the most likely cause for the algorithm behavior seen earlier. As the population size grows, a larger portion of it does not get selected for recombination by fitness proportional selection than by other algorithms. This, in turn, leads to higher loss of diversity and solutions of lower quality.

## 4.2 Enforced Diversity

In Section 2 we have also showed that all selection algorithms suffer from the loss of diversity for one reason or another. The original motivation for using unbiased tournament was to reduce the number of factors that lead to that. There were two such factors present in random tournament (not *sampled* and not *selected*) and we have effectively reduced that number to one (not *selected* only).

We now demonstrate that when diversity becomes enforced the difference among factors that lead to the loss of diversity becomes less relevant. We enforce diversity by changing the way the next generations are constructed. Instead of saving the best individual and replacing the rest of the population with children, we take the union of children and parents, sort the result by rank and truncate to the original size. To borrow the notation from evolutionary strategy literature[1], we have converted a $(\mu, \lambda)$ algorithm into $(\mu + \lambda)$ where duplicates are not allowed. The latter constraint is key to maintaining a diverse population.

Constructing the next generation in this way is a step towards Whitley's steady-state GA [14], where children are injected into the parent population. The framework, nevertheless, retains its generational property of creating an intermediate population via application of a selection algorithm.

Table 5 and Table 6 show the results of performing 30 runs of the new method on bitstring-encoded and permutation-encoded problems respectively. Although overall solutions found with each selection scheme are better, larger $p$-values demonstrate that the difference in performance is no longer as significant as before. One can say that enforced diversity in a way makes up for "sloppiness" in selection.

Nevertheless, Unbiased Tournament proves to be quite competitive.

| Rana10D | Best | Mean | St.D | $p$-value |
|---|---|---|---|---|
| Fitness Prop. | -461.25 | -416.18 | 21.96 | 0.010 |
| Rank-Based | -453.06 | -427.52 | 17.52 | 0.584 |
| Random Tourn. | -458.97 | -425.65 | 15.81 | 0.320 |
| Unbiased Tourn. | -464.61 | -430.05 | 18.04 | 1.000 |

| Schwefel10D | Best | Mean | St.D | $p$-value |
|---|---|---|---|---|
| Fitness Prop. | -3986.6 | -3775.1 | 109.7 | 0.001 |
| Rank-Based | -4135.6 | -3836.9 | 178.5 | 0.214 |
| Random Tourn. | -4150.0 | -3808.5 | 175.5 | 0.051 |
| Unbiased Tourn. | -4135.9 | -3886.5 | 121.9 | 1.000 |

**Table 5: Rana10D(top) and Schwefel10D(bottom) with POPSIZE=100 and enforced diversity**

| Radar Scheduling | Best | Mean | St.D | $p$-value |
|---|---|---|---|---|
| Fitness Prop. | 123.00 | 157.93 | 23.099 | 0.563 |
| Rank-Based | 102.00 | 146.13 | 21.077 | 0.138 |
| Random Tourn. | 113.00 | 173.80 | 54.577 | 0.081 |
| Unbiased Tourn. | 128.00 | 154.53 | 22.153 | 1.000 |

**Table 6: Radar Scheduling with POPSIZE=100 and enforced diversity**

## 5. CONCLUSION AND FUTURE WORK

Given the popularity of tournament selection, it is important to understand that there are minor algorithm changes that can improve the performance of a generational genetic algorithm. As has been shown, eliminating the loss of diversity due to population members not being sampled leads to better performance. It is also important to note that the effects of failing to sample the most fit individual in the population can be very dramatic when population sizes are small.

We introduce two simple forms of tournament selection that eliminate loss of diversity due to a failure to sample. One of these algorithms, parallel unbiased tournament selection, is still embarrassingly parallel and independent in nature. The other algorithm, which we call unbiased tournament selection and which is analyzed empirically in this paper, has been shown to enhance search on a small but representative sample of test problems. We have also pointed out that the use of either tournament selection scheme yielded no significant difference in performance when diversity was enforced.

Future work should further evaluate parallel unbiased tournament selection. We would also like to examine the impact of bias for tournament selection when the selection pressure is less than or greater than 2.0. It would be desirable to have a way of calculating the loss of diversity in both cases.

## 6. ACKNOWLEDGMENTS

This material is based upon work supported by the National Science Foundation under Grant No. 0117209.

## 7. REFERENCES

[1] Back, T. and Schwefel, H.-P. "An overview of evolutionary algorithms for parameter optimization". *Evolutionary Computation, 1(1):1-23.* 1993.

[2] Baker, J. E. "Reducing Bias and Inefficiency in the Selection Algorithm". In *Proceedings of the Second International Conference on Genetic Algorithms and their Application*, pp. 14-21. Lawrence Erlbaum Associates, Inc. Mahwah, NJ, USA, 1987.

[3] Blickle T. and L. Thiele. “A Comparison of Selection Schemes used in Genetic Algorithms”. *TIK Report No. 11, Computer Engineering and Communication Networks Lab (TIK)*, Swiss Federal Institute of Technology (ETH) Zrich, Switzerland, December 1995.

[4] Blickle T. and L. Thiele. “A Comparison of selection schemes used in evolutionary algorithms”. *Evolutionary Computation*, 4(4): 361-394, 1997.

[5] Goldberg D. “Genetic Algorithms in Search, Optimization and Machine Learning”. *Addison-Wesley*, Reading, MA, 1989.

[6] Goldberg D. and K. Deb. “A Comparative Analysis of Selection Schemes Used in Genetic Algorithms”. In Rawlins, G.J.E., editor, *Foundations of Genetic Algorithms*, pages 69-93, Morgan Kaufmann, San Mateo, California, 1991.

[7] Motoki T. “Calculating the expected loss of diversity of selection schemes”. *Evolutionary Computation*, 10(4): 397-422, 2002.

[8] Poli R. “Tournament Selection, Iterated Coupon-collection Problem, and Backward-chaining Evolutionary Algorithms”. *Proceedings of the Foundations of Genetic Algorithms Workshop (FOGA 8)*, Springer 2005.

[9] Schwefel H.-P. “Evolution and Optimum Seeking”, p.328. *Wiley, New York*, 1995.

[10] Smith D., J. Frank, and A.K. Jonsson. “Bridging the Gap Between Planning and Scheduling”. *Knowledge Engineering Review, 15(1):61-94*, 2000.

[11] Syswerda G. “Schedule Optimization Using Genetic Algorithms”. In L. Davis, ed., *Handbook of Genetic Algorithms*, 332-349, Van Nostrand Reinhold, New York, 1991.

[12] Whitley D. “A Genetic Algorithm Tutorial”, *Statistics and Computing (4):65-85*, 1994.

[13] Whitley D., K. Mathias, S. Rana, J. Dzubera. “Evaluating evolutionary algorithms”. *Artificial Intelligence 85 (1996) 245-276*. Elsevier Science.

[14] Whitley D. “The GENITOR Algorithm and Selection Pressure: Why Rank-Based Allocation of Reproductive Trials is Best”, in *Proceedings of The Third International Conference on Genetic Algorithms*, 116-121, San Mateo, California, USA: Morgan Kaufmann Publishers, 1989.

[15] Whitley D., M. Lunacek, J. Knight. “Ruffled by Ridges: How Evolutionary Algorithms Can Fail”. In K. Deb, ed., *GECCO (2) 2004: 294-306*. Springer-Verlag.