Computing and Informatics, Vol. 22, 2003, ??–??

FINE-GRAINED TOURNAMENT SELECTION OPERATOR IN GENETIC ALGORITHMS

Vladimir Filipović

Faculty of Mathematics
Studentski trg 16
11 000 Belgrade, Serbia and Montenegro
e-mail: vladaf@matf.bg.ac.yu

Manuscript received 6 March 2002; revised 23 May 2003
Communicated by Enn Tyugu

Abstract. Tournament selection is one of the most popular selection operators in Genetic Algorithms. Recently, its popularity is increasing because this operator is well suited for Parallel Genetic Algorithms applications. In this paper, new selection operator is proposed. The new operator, which should be an improvement of the tournament selection, is named "Fine-grained Tournament Selection" (FGTS).

It is shown that classical tournament selection is a special case of the FGTS and that new operator preserves its good features. Furthermore, theoretical estimations for the FGTS are made. Estimations for the FGTS are similar to those for the classical tournament selection. Finally, classical tournament selection, rank-based selection and FGTS are experimentally compared on a real world NP-hard problem and the obtained results are discussed.

Keywords: Genetic algorithms, tournament selection, theoretical estimations, NP-hard problems, SPLP

1 GENETIC ALGORITHMS

Genetic algorithms (GA) are applied in some optimization problems: job shop scheduling, traveling salesman, economic modeling, pipeline projecting etc. [?, ?]. They are also applied in machine learning, game theory; neural networks, in geometric reasoning [?, ?, ?].

GA are inspired by the Darwin's evolution theory and the Mendel's laws of inheritance. If Darwin's theory describes natural processes, then GA imitate natural processes in attempting to solve particular problem.

It is widely accepted among authors in GA field is that only modifications that have analogy in nature can significantly improve their performances [?, ?]. Such modifications can be applied to various problem domains.

Great deal of popularity of GA is based on the fact that new solutions are achieved through parallel process. Holland [?] observed the parallel nature of reproductive paradigm and inherent efficiency of the parallel processing ten years before formal genesis of the GA.

In addition, GA are easy to hybridize: they can be combined with other algorithms that solve specific problem. New, hybrid algorithm has good features of both base algorithms (explained in detail in [?]).

GA have the following structure:

- Search space — space of all possible solutions.
- Population — set of actual candidates for solution; elements of population are individuals (items, or search nodes, or points in search space).
- String space — contains string representations of individuals in population.
- Functions that convert points in search space to the points in string space and vice versa (coding and decoding).
- Set of genetic operators for generating new strings (and new individuals).
- Fitness function — evaluates fitness (degree of adaptation) for each individual in population.
- Stochastic control — controls genetic operators and governs execution of GA.

Basic steps in GA are:

1. Initialization — the initial population is created (usually by random sampling).
2. Evaluation — fitness is calculated for individuals in the population.
3. Selection — surviving individuals are chosen from the population (usually according to values of fitness function).
4. Recombination (includes crossover and mutation) — changes in the individual's representation.
5. Repeat steps 1 to 4 until the population fulfils finishing criteria.

The features that made distinction between GA and any other problem-solving method are as follows:

- GA work with codes that represent parameters, not with the pure parameters.
- GA search in several points (population) simultaneously, therefore they are very robust.
- GA use probabilistic rather than deterministic transition rules.

- GA (in their pure form) do not exploit additional information about the nature of a problem.

Simple Genetic Algorithm (SGA) represents starting point for all other modifications of GA. It has the following operators: one-point crossover, one-point mutation, and roulette selection. Individuals in SGA have binary representation (binary string of fixed length represents individual).

In order to improve SGA performances, the authors have used different modifications of SGA during:

- setting of stochastic parameters — controlling application of genetic operators (see [?]).
- representation — transformation of the search space to the string space. Examples of such modifications are Gray coding and multidimensional coding (see [?]).
- initialization — formation of the string that represents an individual (see [?]).
- evaluation — computing fitness value of each individual in population (see [?, ?]).
- design of genetic operators - crossover, mutation (see [15]) and selection (see [?, ?] and [?]), etc.
- execution — reordering of the genetic operators (see [?, ?]).
- replacement of the individuals in population — steady state and elitist strategy are the nost popular ones (see [?]).
- exploitation of explicit parallelism - Parallel GA (see [?, ?] and [?]).

The theory and modifications of GA are developing simultaneously. This theory should give answers to questions concerning quality of proposed modifications (see [?, ?] and [?]).

2 SELECTION

Selection operator chooses individuals in the population that will survive. Better-fitted individual will survive with higher probability. Authors usually design selection so that "luck" has influence, e.g. selection operator usually does not work in a deterministic way. Therefore, worse fitted individuals sometimes win and pass their genetic code. Hence, selection operator more precisely emulates phenomena and processes in nature. In nature, very often, better-fitted individual does not survive. In addition, better-fitted individuals (in that moment of time) do not carry all the good genes.

Well-designed selection operator should maintain population diversity — that is the key issue for achieving quality of seeking for the best solution and for avoiding premature convergence.

GA (see [?]) have inherent conflicts between exploitation and exploration. *Exploitation* means that, during the solution searching, algorithm uses information obtained in the past (about previously visited points in search space) in order to determine smaller regions that are promising for further search. *Exploration* is a procedure that obtains new information — it visits new regions in the search space in order to find promising points or sub regions. In contrast to exploitation, exploration includes jumps into the unknown.

Selection operator *exploits* already obtained information about individual's fitness in the population during the adaptive process of looking for solution. Recombination operators (especially mutation) change genetic material in the population and directed search into the new regions. Therefore, population accepts new individuals and new alternatives (new opportunities for finding solution) that are *explored.*

Optimal ratio between exploitation and exploration differs for different GA and for different problems. Setting values of the stochastic parameters that control genetic operators is one way to achieve balance between exploitation and exploration (for a specific problem).

Several different ways to theoretically estimate quality of the proposed selection are developed. Goldberg and Deb [**?**] introduced takeover time — number of generations needed for the best individual to spread over the whole population, assuming that GA are executed without recombination. Neri also studied GA without recombination operators [**?**]. Baeck analyzed the most popular selection operators and their takeover time, too. Beside derivation of the takeover time in papers [**?**, **?**, **?**], Baeck also derived probability of surviving selection for the fixed individual. Tidor and De La Maza [**?**] analyzed whether selection is invariant to the translation and scaling. In all these approaches, only small part of overall selection's behavior is being studied (somewhere the best individual, somewhere average fitness in population). More precise and more general framework for analyzing specific selection operator is analyzing the fitness distribution of population before and after applying that selection operator, as Blickle and Thiele did in [**?**]. In order to analyze selection operator, following quantities are defined and several claims are proved.

Probability of selecting (e.g. probability of being chosen) for fixed individual a_k during execution of selection method is denoted with $p_s(a_k)$. For the purposes of theoretical analysis, it will be assumed that individuals in population are sorted in descending order according to its fitness $f(a_k)$.

Population *fitness distribution function*, denoted as $s : R \to {Z^+}_0$, is the function that projects arbitrary fitness value $f \in R$ into the number of individuals in the population with such fitness. Its continuous counterpart, denoted as $\overline{s(f)}$, is called population fitness distribution function, too.

Suppose that n_f is number of distinct fitness values in population and suppose that fitness values are sorted in the ascending order, e.g. $f_1 < f_2 < \ldots < f_{n_f-1} < f_{n_f}$. *Cumulative fitness distribution*, denoted as $S(f_k)$, is number of the individuals in population with fitness less or equal to f_k:

$$S(f_k) = \begin{cases} 0, & k < 1 \\ \sum_{j=1}^{k}(s(f_j)), & 1 \leq k \leq n_f \\ n, & k > n_f \end{cases}$$

Continuous counterpart of the previously defined cumulative fitness distribution, denoted as $\overline{S(f)}$, is: $\overline{S(f)} = \int_{f_1}^{f_{n_f}} \overline{s(x)} dx$.

Now, it is possible to define selection. *Selection method* Ω is the function that transforms fitness distribution s into other fitness distribution s'. So, $s' = \Omega(s, param)$, knowing that *param* is optional parameter list for specified selection method. *Expected fitness distribution upon applying selection method* Ω, denoted as Ω^*, is defined as mathematical expectation of fitness after selection method execution, e.g. $\Omega^*(s, param) = E(\Omega(s, param))$. This equation is often denoted as $s^* = \Omega^*(s, param)$. The goal of the analysis is to anticipate s^* that depends on fixed fitness distribution s.

Mean population's fitness distribution before selection, denoted as M, is $M = \frac{1}{n}\sum_{j=1}^{n} f_j$, and *mean population's fitness distribution after selection*, denoted as M^*, is $M^* = \frac{1}{n}\sum_{j=1}^{n_f} s(f_j) f_j$. Furthermore, *standard deviation of population's fitness distribution before selection*, denoted as σ^2, is $\sigma^2 = \frac{1}{n_f}\sum_{j=1}^{n_f}(s(f_j) - M)^2$, and *standard deviation of population's fitness distribution after selection*, denoted as σ^{*2}, is $\sigma^{*2} = \frac{1}{n}\sum_{j=1}^{n}(f_j - M^*)^2$.

Continuous counterpart of the mean population's fitness distribution before selection, denoted as $\overline{M}$, is: $\overline{M} = \frac{1}{n}\int_{f_1}^{f_{n_f}} \overline{s}(f) f df$, and continuous counterpart of the mean population's fitness distribution after selection, denoted as $\overline{M^*}$, is $\overline{M} = \frac{1}{n}\int_{f_1}^{f_{n_f}} \overline{s^*}(f) f df$. Furthermore, continuous counterpart of the standard deviation of population's fitness distribution before selection, denoted as $(\overline{\sigma})^2$, is $(\overline{\sigma})^2 = \frac{1}{n}\int_{f_1}^{f_{n_f}} \overline{s}(f)(f - \overline{M})^2 df = \frac{1}{n}\int_{f_1}^{f_{n_f}} f^2 \overline{s}(f) df - \overline{M}^2$.

Reproduction rate of the fixed selection method, denoted as $\overline{R(f)}$, is defined in the following way:

$$\overline{R(f)} = \begin{cases} \frac{\overline{s^*}(f)}{\overline{s}(f)}, & \overline{s}(f) > 0 \\ 0, & \overline{s}(f) = 0 \end{cases}$$

Loss of diversity for fixed selection method, denoted as p_d, is the ratio between the number of individuals in the population that are not chosen during selection phase and overall number of individuals in the population. Loss of diversity is a number between zero and one. It should be as low as possible because high loss of diversity increases premature convergence risk.

Selection intensity (also denoted as selection pressure) is often used in different contexts and for different properties of the selection method. The term "selection intensity" is used in the same way as in the classical genetics.

Selection intensity $I(\Omega, \overline{s})$ for fixed selection method Ω and fitness distribution $\overline{s}(f)$ is $I(\Omega, \overline{s}) = \frac{\overline{M^* - M}}{\sigma}$. It is clear that selection intensity depends on the initial population fitness distribution and that different distributions will lead to different selection intensities for the same selection method. *Standardized selection intensity* for fixed selection method Ω, denoted as $I(\Omega)$, is $I(\Omega) = \int_{-\infty}^{-\infty} \overline{\Omega^*} \left(N(0,1)(f)\right) f df$. Standardized selection intensity is the mathematical expectation of average fitness population after application of selection method Ω to normalized Gaussian fitness distribution $N(0,1)(f)$ (density is given with formula $\frac{1}{\sqrt{2\pi}}e^{-\frac{f^2}{2}}$).

3 TOURNAMENT SELECTION

Tournament selection is one of the most popular selection operators. Its popularity has been growing in the recent time, because this operator is well suited for applications in Parallel Genetic Algorithms (more details in [?, ?]).

Tournaments are small competitions among the individuals. In tournament selection, an individual passes into the next generation if it is better fitted than opponents randomly selected from the population. Tournament size N_{tour} is the selection parameter.

Algorithm (written in pseudo Pascal) looks as follows:

```
Input: Population $a$ (size of array $a$ is $n$), tournament size $N_{tour}$, $N_{tour} \in N$
Output: Population after selection $a'$ (size of array $a'$ is $n$)
Tournament_Selection:
begin
   for $i := 1$ to $n$ do
      $a[i]' :=$ best fitted among $N_{tour}$ individuals randomly selected from population $a$;
   return $a'$
end
```

Execution time for this algorithm is $O(nN_{tour})$. Usually, N_{tour} does not depend on n, so algorithm is linear and time is $O(n)$ for fixed N_{tour}. Thus, tournament selection differs from explicitly ranking schemes such as linear ranking (see [?]) and exponential ranking, because it does not need to sort the population during its work first.

Different authors analyzed theoretical features of the tournament selection and they deduced the following important formulas [?, ?] and [?].

Probability of selecting fixed individual (where k denotes position of that fixed individual in population, assuming that population is sorted by fitness in non-ascending order):

$$p_s(a_k) = \frac{1}{n^{N_{tour}}}\left(k^{N_{tour}} - (k-1)^{N_{tour}}\right). \tag{3.1}$$

Expected distribution of fitness after execution of tournament selection:

$$s^*(f_k) = {\Omega^*}_{tour}(s, N_{tour})(f_k) = n\left(\left(\frac{S(f_k)}{n}\right)^{N_{tour}} - \left(\frac{S(f_{k-1})}{n}\right)^{N_{tour}}\right). \tag{3.2}$$

Expected continuous distribution of fitness after execution of tournament selection:

$$\overline{s^{*}(f)} = \overline{\Omega^{*}{}_{tour}}(s, N_{tour})(f) = N_{tour}\overline{s}(f)\left(\frac{\overline{S}(f)}{n}\right)^{N_{tour}-1}. \tag{3.3}$$

Reproduction rate in tournament selection:

$$\overline{R_{tour}}(f) = N_{tour}\left(\frac{\overline{S}(f)}{n}\right)^{N_{tour}-1}. \tag{3.4}$$

Loss of diversity in tournament selection:

$$p_{d,tour} = N_{tour}{}^{-\frac{1}{N_{tour}-1}} - N_{tour}{}^{-\frac{N_{tour}}{N_{tour}-1}}. \tag{3.5}$$

Standardized selection intensity for the tournament selection:

$$I(tour, N_{tour}) = I_{tour}(N_{tour}) = \int_{-\infty}^{+\infty} N_{tour}x\frac{1}{\sqrt{2\pi}}e^{-\frac{x^2}{2}}\left(\int_{-\infty}^{x}\frac{1}{\sqrt{2\pi}}e^{-\frac{y^2}{2}}dy\right)^{N_{tour}-1} dx. \tag{3.6}$$

4 FINE-GRAINED TOURNAMENT SELECTION

In spite of many good characteristics of the tournament selection, there is one major problem: the ratio between exploration and exploitation cannot be set precisely, which is crucial for using this selection method in practice. The ratio between exploration and exploitation governs the search process in GA. Level of exploration (looking for new solutions) and exploitation (using previously acquired knowledge) is determined by N_{tour} in the tournament selection.

The parameter N_{tour} can be one of few integer values (usually 2–3 values are good candidates). Very often, the search process converges too slow with smaller tournament size and too fast with bigger tournament size. It is desirable to create a selection method that preserves good features of the tournament selection and (at the same time) allows that setting of the ratio between exploration and exploitation becomes more precise.

Fine-grained Tournament Selection (denoted as FGTS) is an improvement of the tournament selection. FGTS is controlled by real value parameter F_{tour} (the *desired average tournament size*) instead of the integer parameter N_{tour} (the tournament size). Similarly to the tournament selection, an individual is chosen if it is the best individual on the tournament. However, unlike tournament selection, size of the tournament is not unique in the whole population, i.e., tournaments with different number of competitors can be held within one step of the selection.

The parameter F_{tour} governs the selection procedure; therefore, average tournament size in population should be as close to F_{tour} as possible.

It is already noticed that, among many modifications of SGA, the modifications where analogy with phenomena and processes in the nature exists have most general applicability [?, ?]. In nature, large number of animal species (higher primates) live in groups (known as herds, flocks or packs) where only the fittest male individual in the group spreads his genetic material to new generation. Determining who is the fittest in the group is made by holding the tournament. Therefore, the tournament selection has the analogy in the processes in nature. Knowing the fact that cardinality of groups is not the same in all regions, it can be concluded that FGTS (allowing that the size of simultaneous tournaments is not the same everywhere) emulates processes in nature better than the classical tournament selection.

During FGTS design, several implementations are created and tested. Chosen FGTS implementation creates tournaments whose sizes vary at the most by one. Other tested FGTS implementations (those that are not chosen) have more complex structure, require slightly more time for execution, without any improvement in the quality of obtained results.

Therefore, sizes of the tournaments are ${F_{tour}}^- = [F_{tour}]$, ${F_{tour}}^+ = [F_{tour}] + 1$. The size of all of n held tournaments is either ${F_{tour}}^-$ or ${F_{tour}}^+$. The number of tournaments with size ${F_{tour}}^-$ (denoted as n^-) and the number of tournaments with size ${F_{tour}}^+$ (denoted as n^+), have to fulfill two conditions: their sum should be n and average tournament size should be as close to real value F_{tour} as possible:

$$\left\{ \begin{array}{lll} n^+ & + n^- & = n \\ n^+ {F_{tour}}^+ & + n^- {F_{tour}}^- & = n F_{tour} \end{array} \right.$$

The explicit formulas for n^+ and n^- are obtained by solving these equations. The algorithm (written in pseudo Pascal) looks as follows:

```
Input: Population $a$ (size of array $a$ is $n$), desired average tournament size $F_{tour}$, $F_{tour} \in R$
Output: Population after selection $a'$ (size of array $a'$ is $n$)
Fine_Grained_Tournament_Selection:
begin
    ${F_{tour}}^-$ := trunc( $F_{tour}$);
    ${F_{tour}}^+$ := trunc( $F_{tour}$) + 1;
    $n^-$ := trunc ( $n$ * (${F_{tour}}^+$ - $F_{tour}$) );
    $n^+$ := $n$ - $n^-$;
    /* tournaments with size ${F_{tour}}^-$ */
    for $i$ := 1 to $n^-$ do
        $a[i]'$ := best fitted among ${F_{tour}}^-$ individuals randomly selected from population $a$;
    /* tournaments with size ${F_{tour}}^+$ */
    for $i$ := $n^- + 1$ to $n$ do
        $a[i]'$ := best fitted among ${F_{tour}}^+$ individuals randomly selected from population $a$;
    return $a'$
end
```

Execution time for this algorithm is $O(nF_{tour})$. Due to the fact that F_{tour} usually does not depend on n, the algorithm is linear and the time is $O(n)$.

The following claims establish a link between the FGTS and the classical tournament selection.

Theorem 4.1. If the parameter F_{tour} (desired average tournament size) is an integer, then FGTS is the same as the classical tournament selection, i.e. FGTS is a superset of classical tournament selection.

Proof. If F_{tour} is an integer, then the following is true for FGTS: ${F_{tour}}^- = F_{tour}$, ${F_{tour}}^+ = F_{tour} + 1$, $n^- = n$, $n^+ = 0$. Therefore, n tournaments of the integer size F_{tour} are held during the selection, which is exactly the same procedure as in the classical tournament selection. □

Theorem 4.2. The difference between average size of tournaments held during one step of FGTS and the parameter F_{tour} has upper bound $1/n$.

Proof. The average tournament size in FGTS (denoted as ${F_{tour}}'$) is ${F_{tour}}' = \frac{n^+ {F_{tour}}^+ + n^- {F_{tour}}^-}{n}$. The value ${F_{tour}}'$ can be expressed as

$$\begin{aligned} {F_{tour}}' &= \tfrac{n^+ {F_{tour}}^+ + n^-({F_{tour}}^+ - 1)}{n} = \tfrac{n {F_{tour}}^+ - n^-}{n} = {F_{tour}}^+ - \tfrac{n^-}{n} \\ &= [F_{tour}] + 1 - \tfrac{[n(1-(F_{tour}-[F_{tour}]))]}{n}. \end{aligned} \tag{4.1}$$

Since $x - 1 \leq [x] \leq x$, it follows:

$$[F_{tour}] + 1 - \tfrac{n(1-(F_{tour}-[F_{tour}]))-1}{n} \leq {F_{tour}}' \leq [F_{tour}] + 1 - \tfrac{n(1-(F_{tour}-[F_{tour}]))}{n}$$

$$F_{tour} - \tfrac{1}{n} \leq {F_{tour}}' \leq F_{tour}.$$

The theorem is proved by substituting the above inequality into the expression $|F_{tour} - {F_{tour}}'|$. □

The following theoretical estimations are of the same kind as estimations in equations 3.1.–3.6. that are deduced for the classical tournament selection.

Proposition 4.3. For fixed individual a_k the probability of being chosen during FGTS (individual is in position k in population that is implicitly sorted by fitness in descending order) can be calculated as follows:

$$p_s(a_k) = \frac{n^+}{n^{{F_{tour}}^+}} \left(k^{{F_{tour}}^+} - (k-1)^{{F_{tour}}^+}\right) + \frac{n^-}{n^{{F_{tour}}^-}} \left(k^{{F_{tour}}^-} - (k-1)^{{F_{tour}}^-}\right).$$

Proof. Suppose that F_{tour} is parameter of the selection method and ${F_{tour}}^+$, ${F_{tour}}^-$, n^+, n^- are calculated according to previous formulas. Equation 3.1 states that fixed individual a_k is chosen in the tournament of size $s (s \in N)$ with probability $p_{chose}(a_k) = \frac{1}{n^s}(k^s - (k-1)^s)$. In the FGTS with parameter F_{tour}, probability that tournament has size ${F_{tour}}^+$ is $\frac{n^+}{n}$ and that tournament has size ${F_{tour}}^-$ is $\frac{n^-}{n}$. According to total probability formula,

$$p_s(a_k) = \frac{n^+}{n} \frac{1}{n^{{F_{tour}}^+}} \left(k^{{F_{tour}}^+} - (k-1)^{{F_{tour}}^+}\right) + \frac{n^-}{n} \frac{1}{n^{{F_{tour}}^-}} \left(k^{{F_{tour}}^-} - (k-1)^{{F_{tour}}^-}\right).$$

Claim follows directly from the last equation. □

Proposition 4.4. FGTS is invariant to the translation and scaling.

Proof. In FGTS, individual is selected due to its relative position, not due to the explicit fitness. Translation and scaling of the population does not change individual's relative position, therefore FGTS is invariant to both translation and scaling. □

Proposition 4.5. Expected distribution of fitness $s^* = \Omega_{ftour}{}^*(s, F_{tour})$ after execution of the FGTS, with desired average tournament size F_{tour} applied to the fitness distribution s is

$$s^*(f_k) = \Omega_{ftour}{}^*(s, F_{tour})(f_k) =$$

$$\frac{n^+}{n^{F_{tour}{}^+}}\left((S(f_k))^{F_{tour}{}^+} - (S(f_{k-1}))^{F_{tour}{}^+}\right) + \frac{n^-}{n^{F_{tour}{}^-}}\left((S(f_k))^{F_{tour}{}^-} - (S(f_{k-1}))^{F_{tour}{}^-}\right).$$

Proof. First, the expected number of individuals with fitness less or equal to f_k (denoted as $S^*(f_k)$) is calculated. Fixed individual with fitness less or equal to f_k wins the tournament, if all competing individuals in the tournament have fitness f_k or less. Probability for choosing individual with fitness less or equal to f_k is $\frac{S(f_k)}{n}$. Therefore, the expected number of individuals that survive FGTS and have fitness less or equal to f_k is

$$S^*(f_k) = n^+ \left(\frac{S(f_k)}{n}\right)^{F_{tour}{}^+} + n^- \left(\frac{S(f_k)}{n}\right)^{F_{tour}{}^-}.$$

From the definition of expected fitness distribution can be derived $s^*(f_k) = S^*(f_k) - S^*(f_{k-1})$; thus:

$$s^*(f_k) = \Omega_{ftour}{}^*(s, F_{tour})(f_k) =$$

$$\left(n^+ \left(\frac{S(f_k)}{n}\right)^{F_{tour}{}^+} + n^- \left(\frac{S(f_k)}{n}\right)^{F_{tour}{}^-}\right) -$$

$$\left(n^+ \left(\frac{S(f_{k-1})}{n}\right)^{F_{tour}{}^+} + n^- \left(\frac{S(f_{k-1})}{n}\right)^{F_{tour}{}^-}\right).$$

Moreover, the following can be obtained:

$$s^*(f_k) = \Omega_{ftour}{}^*(s, F_{tour})(f_k) =$$

$$n^+ \left(\left(\frac{S(f_k)}{n}\right)^{F_{tour}{}^+} - \left(\frac{S(f_{k-1})}{n}\right)^{F_{tour}{}^+}\right) + n^- \left(\left(\frac{S(f_k)}{n}\right)^{F_{tour}{}^-} - \left(\frac{S(f_{k-1})}{n}\right)^{F_{tour}{}^-}\right).$$

Claim follows directly from the last equation. □

Note. The previous claim shows the influence of the value of the parameter F_{tour} on the selection method. It can be shown that Proposition 4.3. is special case of Proposition 4.5 (the case when fitness function is injective). In that case, for each $k \in \{1, 2, \ldots, n\}$ is $s(f_k) = 1$; therefore $S(f_k) = k$.

Proposition 4.6. Let us assume that $\overline{s}(f)$ represents continuous fitness distribution. After FGTS execution (desired average size of the tournament is F_{tour}), expected continuous distribution of fitness $\overline{s^*}(f) = \overline{\Omega_{ftour}}^*(s, F_{tour})$ is

$$\overline{s^*}(f) = \overline{\Omega_{ftour}}^*(s, F_{tour})(f) =$$

$$\overline{s}(f)\left(\frac{n^+ {F_{tour}}^+}{n^{{F_{tour}}^+}}\left(\overline{S}(f)\right)^{{F_{tour}}^+-1} + \frac{n^- {F_{tour}}^-}{n^{{F_{tour}}^-}}\left(\overline{S}(f)\right)^{{F_{tour}}^--1}\right).$$

Proof. The individual with fitness less or equal to f wins the tournament in case that all competing individuals in the tournament have fitness f or less. Knowing that probability of choosing an individual with fitness less or equal to f is $\frac{\overline{S}(f)}{n}$, it can be obtained that:

$$\overline{S^*}(f) = n^+\left(\frac{\overline{S}(f)}{n}\right)^{{F_{tour}}^+} + n^-\left(\frac{\overline{S}(f)}{n}\right)^{{F_{tour}}^-}.$$

According to the definition in chapter 2, $\overline{S}(f) = \int_{f_1}^{f} \overline{s}(x)dx$ and differencing gives $\overline{s}(f) = \frac{d\left(\overline{S}(f)\right)}{df}$.

Therefore, $\overline{s^*}(f)$ is derivation of $\overline{S^*}(f)$, by f; thus:

$$\overline{s^*}(f) = \frac{d\left(\overline{S^*}(f)\right)}{df} = \frac{d\left(n^+\left(\frac{\overline{S}(f)}{n}\right)^{{F_{tour}}^+} + n^-\left(\frac{\overline{S}(f)}{n}\right)^{{F_{tour}}^-}\right)}{df}$$

$$\overline{s^*}(f) = \frac{n^+ {F_{tour}}^+}{n^{{F_{tour}}^+}}\left(\overline{S}(f)\right)^{{F_{tour}}^+-1}\frac{d\overline{S}(f)}{df} + \frac{n^- {F_{tour}}^-}{n^{{F_{tour}}^-}}\left(\overline{S}(f)\right)^{{F_{tour}}^--1}\frac{d\overline{S}(f)}{df}.$$

□

Theorem 4.7. Reproduction rate in the FGTS is

$$\overline{R_{tour}}(f) = \frac{n^+ {F_{tour}}^+}{n^{{F_{tour}}^+}}\left(\overline{S}(f)\right)^{{F_{tour}}^+-1} + \frac{n^- {F_{tour}}^-}{n^{{F_{tour}}^-}}\left(\overline{S}(f)\right)^{{F_{tour}}^--1}.$$

Proof. It can be concluded directly from the definition of reproduction rate and the previous proposition (proposition 4.6):

$$\overline{R_{tour}}(f) = \frac{\overline{s^*}(f)}{\overline{s}(f)} = \frac{\overline{s}(f)\left(\frac{n^+ {F_{tour}}^+}{n^{{F_{tour}}^+}}\left(\overline{S}(f)\right)^{{F_{tour}}^+-1} + \frac{n^- {F_{tour}}^-}{n^{{F_{tour}}^-}}\left(\overline{S}(f)\right)^{{F_{tour}}^--1}\right)}{\overline{s}(f)}.$$

□

Note. The previous theorem indicates that individuals with the lowest fitness have the lowest reproduction rate and that the best fitted individuals have the highest reproduction rate. In other words, reproduction rate for the FGTS is increasing function of the fitness.

Theorem 4.8. Standardized selection intensity for the FGTS is

$$\begin{aligned} I(ftour, F_{ftour}) = \int_{-\infty}^{+\infty} x \frac{1}{\sqrt{2\pi}} e^{\frac{-x^2}{2}} \Bigg(& \frac{n^{+} {F_{tour}}^{+}}{n^{{F_{tour}}^{+}}} \left(\int_{-\infty}^{x} \frac{1}{\sqrt{2\pi}} e^{\frac{-y^2}{2}} dy \right)^{{F_{tour}}^{+}-1} + \\ & \frac{n^{-} {F_{tour}}^{-}}{n^{{F_{tour}}^{-}}} \left(\int_{-\infty}^{x} \frac{1}{\sqrt{2\pi}} e^{\frac{-y^2}{2}} dy \right)^{{F_{tour}}^{-}-1} \Bigg) dx. \end{aligned}$$

Proof. Directly from the definition of the standardized selection intensity and from Proposition 4.5. □

FGTS, like tournament selection, can easily be decentralized.

5 COMPARISION OF THE SELECTION OPERATORS APPLIED ON NP-HARD PROBLEM

Introduction of FGTS is justified only if it gives better results during genetic search than the classical tournament selection. Therefore, it is necessary to compare performances of the selection operators on some real-world problem (see [**?**]).

Combinatorial optimization problems are important part of the global optimization. One such problem is Simple Plant Location Problem (SPLP). This problem is also known as Uncapacitated Warehouse Location Problem or Uncapacitated Facility Location Problem.

Consider a set $I = \{1, \ldots, m\}$ of candidate sites for facility location, and a set $J = \{1, \ldots, n\}$ of customers. Each facility $i \in I$ has a fixed cost f_i. Every customer $j \in J$, has a demand b_j, and c_{ij} is the unit transportation cost from facility i to customer j.

Without loss of generality, the customer demands can be normalized to $b_j = 1$ (proved in [**?**]). It has to be decided which facilities will be established, and the quantities that will be supplied from facility i to customer j, such that the total cost (including fixed and variable costs) is minimized.

Mathematically, the SPLP is

$$min \left(\sum_{i=1}^{m} \sum_{j=1}^{n} c_{ij} x_{ij} + \sum_{i=1}^{m} f_i y_i \right)$$

subject to

$$(\forall j \in J) \sum_{i=1}^{m} x_{ij} = 1$$

$$(\forall i \in I)\,(\forall j \in J)\,(0 \leq x_{ij} \leq y_i \;\wedge\; y_i \in \{0,1\})$$

where

- x_{ij} represents the quantity supplied from facility i to customer j,
- y_i indicates whether facility i is established ($y_i = 1$) or not ($y_i = 0$).

Let the set of established facilities be $E = \{i | y_i = 1\}$ with cardinality $e = |E|$.

Although some special cases of the SPLP are solvable in polynomial time, in general, SPLP is a NP-hard problem (proved in [**?**]). GA are successfully used to solve some NP-hard problems (papers [**?**, **?**, **?**, **?**] are examples of such approach). That approach allows solving SPLP instances of moderate and large size within acceptable time interval.

This problem is successfully solved in [**?**, **?**, **?**] and it is experimentally concluded in those papers that the best results (for solving SPLP problem instances) is obtained with the following scheme:

- The binary encoding of facility sites is used for *representation*. Each individual is represented by the binary string, where one denotes that particular facility is established, while zero shows it is not. The array y_i ($i = 1, \ldots, m$) is obtained from the individual string. This array indicates the established facilities. Since the capacity of the facilities has no limit, if every customer chooses the most suitable facility (with minimal transportation cost), the total cost is minimal.
- The choice of the data structure is very important for the fast implementation of *objective value function*. Beside transportation cost matrix, this implementation also uses the indexed lists of facility sites. Two different methods for computing objective value have been proposed. The choice of particular method depends on the number of established facilities e. The threshold is $e_0 = c\sqrt{m}$, where c is constant from interval $[0.4, 0.5]$. Its value is obtained experimentally to achieve the best performance. If e is large ($e > e_0$), the algorithm is looking for the first facility in indexed list where is $y_i = 1$. The established facility that is found has minimal transportation cost for given customer. In the case of small number e, previous procedure gives poor results, so another strategy is performed: instead of using indexed lists, this procedure uses array of ordinals. The array o contains only established facilities. Looking for the most suitable facility for each customer is done by searching array of ordinals.
- Uniform crossover is used as *crossover* operator. It uses a randomly created crossover mask. If the crossover mask has value one in the specific bit position, then the offspring's bit in that position will be copied from the first parent. Otherwise, if the mask has value zero in that bit position, then the offspring's bit will be copied from the second parent. A new crossover mask is generated for each pair of parents, with probability p_{unif} that a bit is one. The probability that a bit is zero is $1 - p_{unif}$. In this implementation, the crossover rate is $p_{cross} = 0.85$, and the probability of uniform crossover is $p_{unif} = 0.3$.

- The simple mutation with rate p_{mut} that exponentially decreases from $\frac{0.4}{n}$ to $\frac{0.15}{n}$ is used as *mutation* operator. To provide faster execution, the simple mutation operator is performed by using a Gausian distribution. Only the muted genes are processed by this method. The number of muted genes is relatively small in comparison to entire population. Hence, the run-time performance of a simple mutation operator is improved without changing its nature.

- The *population size* is $N_{pop} = 150$ individuals. GA are implemented with steady-state replacement of generations by using *elitist strategy.* In every generation, only 1/3 of population (50 individuals) is replaced and 2/3 of population ($N_{elite} = 100$ individuals) remain from the previous generation. So, 50 worst ranked individuals in the population are replaced by the new ones. These new individuals (1/3 of population) are generated by means of the crossover and mutation. Every elite individual is passed directly into the next generation, giving one copy of it. To prevent undeserved domination of elite individuals over the population their fitness is decreased by the formula

$$f_i = \begin{cases} f_i - \overline{f}, & f_i > \overline{f} \\ 0, & f_i \leq \overline{f} \end{cases}$$

  $1 \leq i \leq N_{elite}$, where $\overline{f} = \frac{1}{N_{pop}} \sum_{i=1}^{N_{pop}} f_i$ is average fitness in entire population. Duplicate individual strings are discarded, and more *diversity* of the population is maintained to avoid premature convergence. Particular individual is discarded by setting its fitness to zero.

- First generation is randomly *initialized* in order to maintain the maximal diversity of the population.

- Maximum number of generations is $N_{gener} = 2000$, except in the case of large-scale test instances, where is $N_{gener} = 4000$. The *finishing criterion* is based on the number of consecutive generations with unchanged best individual. If that number exceeds the value

$$N_{repeat} = \begin{cases} 2\sqrt{mn}, & \text{for ORLIB instances} \\ \sqrt{mn}, & \text{for generated instances} \end{cases}$$

  execution of GA is stopped.

- It is not possible to prove optimality of the obtained solution by GA. If the optimal solution is known in advance, it can be used for error measurement. In the case when optimal solution is not known in advance, the best solution by GA is used for error measurement.

- The run-time performance of GA in this implementation is also improved by *caching* (see [**?**, **?**]). The caching technique decreases run-rime and has no influence on other aspects of GA. It is used to avoid attempts to compute the same objective value. If an objective value is computed for a particular string and the same string appears again, the cached values are used to avoid repetition of computing.

- Rank based selection (with rank 2.5 for the best individual down to 0.7 for the worst, by step 0.012), is used as the *selection* operator. For SPLP, this selection scheme successfully prevents premature convergence in local optima and losing the genetic material.

The instances 41–134 and A to C (used in this section) are taken from ORLIB [?].

Problem instance	Size	File sizes
41–44, 51, 61–64, 71–74	16x50	10 KB
81–84, 91–94, 101–104	25x50	15 KB
111–114, 121–124, 131–134	50x50	31 KB
A–C	100x1000	1.2 MB

Table 1. Parameters for SPLP instances taken from ORLIB

Almost all ORLIB instances have not sufficient size to test behavior of algorithm on large-scale instances properly. Thus, selection strategies are tested on problem instances generated and described in [?]. The instances, shown in Table 2, have a small number of useless facility sites (facility sites that have no chance to be established), and a very large number of suboptimal solutions. Therefore, solving by dual based and other Branch-and-Bound techniques is very difficult for generated instances.

Problem instance	Size	File sizes
MO1-MO5	100x100	100 KB
MP1-MP5	200x200	400 KB
MQ1-MQ5	300x300	900 KB
MR1-MR5	500x500	2.4 MB
MS1-MS5	1000x1000	9.5 MB

Table 2. Parameters for generated instances

In comparison among selection operators over SPLP problem instances, all previously determined parameters (those already proven as the best) are adopted without change. Three types of selection methods are compared:

- *Rank based* selection, with parameters that are previously described.
- Classical *tournament* selection, with tournament size $N_{tour} \in \{5, 6\}$.
- *Fine-grained tournament selection* with desired average tournament size $F_{tour} \in \{4.5, 5.5, 5.6, 5.8, 6.2, 6.4\}$.

Application of the FGTS on SPLP gives better results in comparison to classical tournament selection (see Table 3). These improvements are not small and become even bigger on large problem instances. Moreover, FGTS shows better results than the previous champion (linear ranking).

Results from these tables are obtained by running GA on Pentium III/600 MHz PC, with 330 MB memory size. All these values are averages determined from 20 independent runs per problem instance.

Due to better visibility, results are summarized per instance group. Columns in tables represent problem instance groups and rows represent selection type.

Table 3 contains the results for rank based selection, tournament selection ($N_{tour} = 5, 6$) and FGTS ($F_{tour} = 4.5, 5.5, 5.6, 5.8, 6.2, 6.4$) for all instances. Every cell in the table has two values. The upper value is the average number of generations and lower value is the average running time (in seconds).

In all executions of every problem instance, the result that is equal to optimal or previously best-known solution is obtained. The best times for each instance group in table are bolded.

Selection	41–74	81–104	111–134	A–C	MO	MP	MQ	MR	MS
Rank based	17.7	33.6	109.2	1328	112.4	181.7	269.9	423.7	879.4
(r = 2.5 to 0.7)	0.10	0.17	0.51	16.28	0.59	1.18	2.69	4.68	23.18
FGTS	10.4	34.4	145.3	1633	76.8	131.6	205.9	347	746.9
($F_{tour} = 4.5$)	0.07	0.17	0.65	16.78	0.46	0.83	1.51	3.47	17.24
T	9.1	49.1	128.9	1433	85.2	128.6	206.2	357	732.4
($N_{tour} = 5$)	0.06	0.21	0.52	12.72	0.36	0.67	1.28	3.14	15.78
FGTS	9	53.5	136.6	**1209**	96.4	127.9	208.6	340.6	756.1
($F_{tour} = 5.5$)	0.06	0.24	0.62	**12.24**	0.49	0.79	1.51	3.31	16.77
FGTS	**8.8**	**41.3**	**136.3**	1694	**83.6**	**131.5**	**192.1**	**343.4**	**723.6**
($F_{tour} = 5.6$)	**0.05**	**0.18**	**0.55**	12.47	**0.35**	**0.55**	**0.93**	**2.52**	**14.47**
FGTS	8.9	46.6	151.2	2347	92.2	**122.8**	204.4	345.7	766.2
($F_{tour} = 5.8$)	0.05	0.20	0.61	16.81	0.38	**0.51**	0.99	2.53	14.94
T	**8.8**	67.1	146	1078	96.7	131.3	210.4	332.4	743.8
($N_{tour} = 6$)	**0.05**	0.30	0.65	10.74	0.49	0.81	1.50	3.19	16.31
FGTS	**8.5**	77.1	164.4	1890	83.5	122.6	207	345.6	740.5
($F_{tour} = 6.2$)	**0.05**	0.34	0.73	18.22	0.43	0.76	1.47	3.31	16.16
FGTS	**8.8**	87.1	148.6	2076	87.4	130.7	200.3	340.7	721.3
($F_{tour} = 6.4$)	**0.05**	0.38	0.66	20.06	0.44	0.81	1.42	3.24	15.65

Table 3. Comparison of selection operators for ORLIB instances 41–74, 81–104, 111–134 and A–C and for generated instances MO, MP, MQ, MR, MS

In most cases, the best results (or results that are very close to the best) are produced by FGTS, with ($F_{tour} = 5.6$) (marked with double line). Improvement to other selection method is significant (usually 10 %–20 %). The improvement grows with the size of the instances.

6 CONCLUSION

In this paper Fine-grained Tournament Selection (FGTS) is presented. FGTS is generalization of the classical tournament selection and it keeps all its good features.

FGTS is applied to Simple Plant Location Problem (SPLP). Such approach is very successful in practice and is recommended for large-scale problem instances (more than 1000 facility locations and customers). On SPLP, FGTS significantly outperforms both the rank based and the classical tournament selection.

It would be interesting to design and analyze more sophisticated FGTS operators, that have more than two sizes of tournament and where the number of competitors in specific tournament depends on fitness landscape (defined in [14]) and/or network topology (in the case of Parallel Genetic Algorithms).

REFERENCES

[1] Baeck, T.—Hoffmeister, F.—Schwefel H. P.: A Survey of Evolution Strategies. In: Proc. of the 4th Int. Conf. on Genetic Alghorithms — ICGA '91, pp. 2–9.

[2] Baeck, T.—Hoffmeister, F.: Extended Selection Mechanisms in Genetic Algorithms. In: Proceedings of the Fourth International Conference on Genetic Alghorithms — ICGA '91, Morgan Kaufmann, San Mateo, CA, 1991, pp. 92–99.

[3] Baeck, T.: Self-adaptation in Genetic Algorithms. In: Proceedings of the First European Conference on Artificial Life, MIT Press, 1992.

[4] Baeck, T.: Selective Pressure in Evolutionary Algorithms: A Characterization of Selection Mechanisms. In: Proceedings of the First IEEE Conference on Evolutionary Computation — ICEC '94, 1994, pp. 57–62.

[5] Baeck, T.: Generalized Convergence Models for Tournament and $(\lambda, \mu)-$Selection. In: Proceedings of the Sixth International Conference on Genetic Alghorithms — ICGA '95, Morgan Kaufmann, San Mateo, CA, 1995, pp. 2–8.

[6] Bauer, R. J. Jr.: Genetic Alghorithms and Investment Strategies, John Wiley & Sons, Inc, New York, 1984.

[7] Beasley, J. E.: Lagrangean heuristic for location problems. European Journal of Operational Research, Vol. 65, 1993, pp. 383–399.

[8] Beasley, J. E.: Obtaining Test Problems via Internet. Journal of Global Optimization, Vol. 8, 1996, pp. 429–433, `http://mscmga.ms.ic.ac.uk/info.html`.

[9] Blickle, T.—Thiele, L.: A Mathematical Analysis of Tournament Selection. In: Proceedings of the Sixth International Conference on Genetic Alghorithms — ICGA '95, Morgan Kaufmann, San Mateo, CA, 1995, pp. 9–16.

[10] Cantu-Paz, E.: Efficient and Accurate Parallel Genetic Algorithms, Kluwer Academic Publishers, Boston, 2000.

[11] De Jong, K. A.—Sarma J.: On Decentralizing Selection Algorithms. In: Proceedings of the Sixth International Conference on Genetic Alghorithms — ICGA '95, Morgan Kaufmann, San Mateo, CA, 1995, pp. 17–25.

[12] De La Maza, M.—Tidor, B.: An Analysis of Selection Procedures with Particular Attention Paid to Proportional and Boltzmann Selection. In: Proceedings of the Fifth International Conference on Genetic Alghorithms — ICGA '93, Morgan Kaufmann, San Mateo, CA, 1993, pp. 124–131.

[13] Filipovic, V.—Tosic, D.—Kratica, J.: Experimental Results in Applying of Fine-grained Tournament Selection. In: Proceedings of the 10th Congress of Yugoslav Mathematicians, Belgrade, 2001, pp. 331–336.

[14] Forest, S.—Mitchell, M.: Relative Building-Block Fitness and the Building-Block Hypothesis. In: Foundation of Genetic Alghorithms 2 — FOGA 2, Morgan Kaufmann, San Mateo, CA, 1992, pp. 109–126.

[15] Goldberg, D. E.: Genetic Alghorithms in Search, Optimization & Machine Learning, Addison-Wesley Publishing Company Inc, Reading, MA, 1989.

[16] Goldberg, D. E.—Deb, K.: A Comparative Analysis of Selection Schemes Used in Genetic Algorithm. In: Rawlins B. (Eds.): Foundations of Genetic Alghorithms — FOGA , Morgan Kaufmann, San Mateo, CA, 1991, pp. 69–93.

[17] Harvik, G. R.: Finding Multimodal Solutions Using Restricted Tournament Selection. In: Proceedings of the Sixth International Conference on Genetic Alghorithms — ICGA '95, Morgan Kaufmann, San Mateo, CA, 1995. pp. 24–31.

[18] Holland, J. H.: Adaptation in Natural and Artificial Systems. MIT Press, Cambridge, Massachusetts, 1992.

[19] Kratica, J.—Filipovic, V.—Tosic, D.: Solving the Uncapacitated Warehouse Location Problem by SGA with Add-Heuristic. In: Proceedings of the XV ECPD International conference on material handling and warehousing, Faculty of Mechanical Engineering, Belgrade, 1998, pp. 2.28–2.32.

[20] Kratica, J.: Improving Performances of the Genetic Algorithm by Caching. Computers and Artificial Intelligence, Vol. 18, 1999, No. 3, pp. 271–283.

[21] Kratica, J.: Improvement of Simple Genetic Algorithm for Solving the Uncapacitated Warehouse Location Problem. Advances in Soft Computing — Engineering Design and Manufacturing, Springer-Verlag London Ltd., 1999, pp. 390–402.

[22] Kratica, J.—Tosic, D.—Filipovic, V.—Ljubic, I.: Genetic Algorithm for Designing a Spread-Spectrum Radar Polyphase Code. In: Proceedings of the 5th Online World Conference on Soft Computing Methods in Industrial Applications — WSC5, September 2000, pp. 191–197.

[23] Kratica, J.—Filipovic, V.—Tosic, D.—Ljubic, I.: Solving the Simple Plant Location Problem by Genetic Algorithm. RAIRO Operations Research, Vol. 35, 2001, No. 1, pp. 127–142.

[24] Kratica, J.—Tosic, D.—Filipovic, V.—Ljubic, I.: A Genetic Algorithm for the Uncapacitated Network Design Problem. Soft Computing in Industry — Recent Applications, Springer-Verlag, Berlin, 2002, pp. 329–338.

[25] Kuo, T.—Hwang, S.Y.: A Genetic Algorithm with Disruptive Selection. In: Proceedings of the Fifth International Conference on Genetic Alghorithms — ICGA '93, Morgan Kaufmann, San Mateo, CA, 1993, pp. 65–69.

[26] Michalewicz, Z.: Genetic Alghorithms + Data Structures = Evolution Programs, third edition, Springer, Berlin, 1995.

[27] Neri, F.—Saitta, L.: Analysis of Genetic Alghorithms Evolution under Pure Selection. In: Proceedings of the Sixth International Conference on Genetic Alghorithms — ICGA '95, Morgan Kaufmann, San Mateo, CA, 1995, pp. 32–41.

[28] Reevs, C. R.: Genetic Alghorithms and Neighbourhood Search. In: Evolutionary Computing, AISB Workshop Leeds, april 1994, pp. 115–130.

[29] Thierens, D.—Goldberg, D.: Convergence Models of Genetic Algorithm Selection Schemes. In: Parallel Problem Solving from Nature — PPSN III, Lecture Notes in Computer Science 866, Springer-Verlag, Berlin, 1994, pp. 119–129.

[30] 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 Alghorithms — ICGA '89, Morgan Kaufmann, San Mateo, CA, 1989, pp. 116–123.

Vladimir Filipović (born in 1968) received his Master degree in computer science from the Department of Computer Science of the Faculty of Mathematics in Belgrade, on the theme "Proposition for improvement tournament selection operator in genetic algorithms". He works as a teaching assistant at the above department. His research interests include evolutionary algorithms, Web services (SOAP), operational research and software design.