Proceedings of the
10th CONGRESS OF
YUGOSLAV MATHEMATICIANS
Belgrade, 21–24.01.2001, 331–336

Section:
Computer sciences

EXPERIMENTAL RESULTS IN APPLYING OF FINE GRAINED TOURNAMENT SELECTION

Vladimir Filipović, Dušan Tošić and Jozef Kratica

Abstract. In this article the results achieved by applying rank-based selection, tournament selection and fine-grained tournament selection in genetic algorithms (GA) are described. Several known optimization problems that are solved with genetic algorithms are discussed, and results are compared. The results for different population size (within usual boundaries) are also presented.

1. Genetic algorithms (GA)

Under the most frequent classification, genetic algorithms and some related techniques, together with Fuzzy logic and Neural networks, are part of the so called Soft computing. Among authors in GA area it is widely accepted [1] that only those modifications, which have analogy in nature, can significantly improve theirs performances.

According to Darwin [7], individuals in population are competing for resources. The facts that lie in essence of natural selection process are:

- better fitted individuals more often survive and they have stronger influence on forming new generations;
- individual in new generation is formed by recombination of parent's genetic material;
- from time to time mutation (random change of genetic material) takes place.

Great deals of popularity of GA lies in fact that getting new solutions has, indeed, parallel nature. Holland [2] first observed that parallel nature of reproductive paradigm and inherent efficiency of parallel processing.

Also, GA are easy to hybridize: they can easily be combined with the other algorithms that solve specific problem. New, hybrid algorithm has good features of both base algorithms.

Related notions to GA are:

- *search space*—space of all possible solutions;
- *population*—set of actual candidates for solution; elements of population are called items, or search nodes, or points in search space;
- *string space*—space which contains string representations of items in population;

- *functions for conversion* between points in search space and points in string space (coding and decoding);
- *set of genetic operators* for generating new strings (and new items);
- *fitness function*—it evaluate fitness (degree of adaptation) for each item in population;
- *stochastic control* of genetic operators.

Basic steps in GA are:

1. *Initialization*—generating initial population by random sampling.
2. *Evaluation*—calculating fitness for all items in population.
3. *Selection*—choosing surviving items in population, according to values of fitness function.
4. *Recombination* (includes crossover and mutation)—changing item's representation.
5. *Repeating steps 1–4* until fulfilling finishing criteria.

2. Tournament and fine grained tournament selection

The tournament selection (TS) is one of the most popular selection operators. Its popularity grows in recent time, because this operator is excellent suited for applying in parallel genetic algorithms.

In tournament selection, element of population is chosen for passing into next generation if it is better adopted (has better fitness value) than several randomly selected opponents. The tournament size N_{tour} is selection parameter.

Algorithm can be outlined in the following way:

Input: Population a (size of a is n), tournament size N_{tour}, $N_{tour} \in N$
Output: Population after selection a' (size of a' is n)
Tournament_Selection:

```
begin
  for i := 1 to n do
    a'_i := best fitted item of N_tour elements randomly selected from population;
  return a';
end;
```

The running time for this algorithm is $O(n * N_{tour})$. Usually it is $N_{tour} \ll n$, so algorithm is linear, with the time of execution $O(n)$. Thus, tournament selection differs from explicitly ranking schemes such as linear ranking and exponential ranking, because it does not need to sort firstly population during its work.

The tournament selection allows parallel execution during choosing members of new generation. It is, just like ranking schemes, invariant under translation (adding same value to the fitness of all items in population) and scaling (multiplying fitness of all items in population with some value).

But, tournament selection does not allow precise ratio setting between exploration and exploitation [4] and that is crucial for using this selection method in practice. Ratio between exploration and exploitation governs search process in GA. The level of exploration (looking for new solutions) and exploitation (using

previously acquired knowledge) is in tournament selection determined with N_{tour}, that can be one of few integer values (usually 2–3 values are good candidates). Often it happens that with smaller tournament size search process runs too slow, and that with bigger tournament size it runs too fast and premature converges.

More details about theoretical features of tournament selection can be found in [5].

In [5] one improvement is described of the tournament selection, named *fine grained tournament selection* (FGTS).

Instead of integer parameter N_{tour} (which represents tournament size), new operator allows real valued parameter F_{tour}—wanted average tournament size. This parameter governs selection procedure, so average tournament size in population should be as close as possible to it.

Just like in tournament selection, item is chosed if it is better than its randomly chosed opponents. Moreover, this time tournament size is not fixed within population. So, during one selection step tournaments will be held with different number of competitors.

The current implementation of fine grained tournament selection during one step of execution holds tournaments with difference among sizes less or equal to 1. Sizes of n held tournaments are chosen with intention that its average becomes as close as possible to real value F_{tour}.

Algorithm looks like:

Input: Population a (size of a is n), wished average tournament size F_{tour}, $F_{tour} \in R$
Output: Population after selection a' (size of a' is n)
Fine_Grained_Tournament_Selection:

```
begin
F⁻tour := trunc(Ftour)
F⁺tour := trunc(Ftour) + 1
n⁻ := trunc(n * (1 - (t - trunc(Ftour) ) ) )
n⁺ := n - trunc(n * ( 1 - ( t - trunc(Ftour) ) ) )
  /* tournaments with size F⁻tour */
  for i := 1 to n⁻ do
    a'i := best fitted item of F⁻tour elements randomly selected from population;
    /* tournaments with size F⁺tour */
  for i := n⁻ + 1 to n do
    a'i := best fitted item of F⁺tour elements randomly selected from population;
  return a'
end;
```

The fine grained tournament selection provides that ratio between exploration and exploitation can be set very precise. This selection method (like tournament selection) also has analogy with some processes in nature.

Theoretical properties of new selection operator are also discussed in [5]. It is shown that new operator preserves good features of classical tournament selection

(algorithm has linear time of execution, it allows execution in parallel, it is invariant to translation and scaling, etc.).

In this way it is proved that classical tournament selection being special case of fine grained tournament selection.

3. Problem definition

The 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 a loss of generality we can normalize the customer demands to $b_j = 1$.

The following has to be decided:

- facilities to be established and
- quantities to be supplied from facility i to customer j

such that the total cost (including fixed and variable costs) is minimized.

Mathematically, the SPLP is formulated in the following way:

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

subject to:

$$\sum_{i=1}^{m} x_{ij} = 1 \qquad \forall j \in J; \tag{2}$$

$$0 \leq x_{ij} \leq y_i \ \wedge \ y_i \in \{0,1\}, \qquad \forall i \in I, \forall j \in J; \tag{3}$$

where x_{ij} represents the quantity supplied from facility i to customer j, and y_i indicates whether facility i is established ($y_i = 1$) or not ($y_i = 0$). Let the set of established facilities be $E = \{\, i \mid y_i = 1 \,\}$ with cardinality $e = |E|$.

4. GA implementation

The implementation used for comparison among several selection methods is based on binary representation, uniform crossover, simple mutation, steady-state generation replacement with elitist strategy and caching of GA.

Three types of selection methods are compared:

- Rank based selection (RBS), with rank 2.5 for the best individual down to 0.7 for the worst, by step 0.012. For SPLP, this selection scheme successfully prevents premature convergence in local optima and loosing the genetic material.
- Classical tournament selection with tournament size $N_{tour} \in \{5, 6\}$.
- Fine grained tournament selection with wanted average tournament size $F_{tour} \in \{4.5, 5.5, 5.6, 5.8, 6.2, 6.4\}$

The uniform crossover is performed, that provides minimal disruption of individual's genes. It is important part for success of GA, because interaction among genes in individual's gene code is not too big. The simple mutation is used and its run-time is improved by realization through normal distribution. Mutation rate depends on problem size and it exponentially decreases from $0.4/n$ to $0.15/n$.

Steady-state generation replacement with elitist strategy is used. In each generation, only 1/3 of population is replaced. Remaining 2/3 of population is directly passed to the next generation. Those elitist individuals do not need reevaluation (their objective value is already evaluated). The maximal number of generation is $2 * n$. If best individual is same for n generations, GA terminates.

The population size is 150 individuals and population is randomly initialized in first generation. Binary code is used for encoding of variables.

Finally, the run-time performance of the GA is improved by caching technique [3]. Least recently used (LRU) strategy, which is simple but effective, is used for caching GA. Caching is implemented by hash-queue data structure.

5. Computational results

Applying of the fine grained tournament selection on SPLP gives better results in comparison to both rank-based and classical tournament selection (see Tables 1–3). The results from these tables are obtained running GA on Pentium III/600MHz PC, with 330 megabytes of psychical memory. All these values are averages determined from 20 independent runs per problem instance.

In order to get better visibility, results are displayed summarily per instance group. Rows in tables represent problem instance groups and columns represent selection type.

Inst.	*RBS*	*FGTS*(4.5)	*TS*(5)	*FGTS*(5.5)	*TGTS*(5.6)	*FGTS*(5.8)	*TS*(6)	*FGTS*(6.2)	*FGTS*(6.4)
41–74	17.7	10.4	9.1	9.0	8.8	8.9	8.8	*8.5*	8.8
	0.10	0.07	0.06	0.06	0.05	0.05	0.05	*0.05*	0.05
81–104	*33.6*	34.4	49.1	53.5	41.3	46.6	67.1	77.1	87.1
	0.17	0.17	0.21	0.24	0.18	0.20	0.30	0.34	0.38

Table 1. Comparison of selection operators for ORLIB instances 41–74, 81–104

Inst.	*RBS*	*FGTS*(4.5)	*TS*(5)	*FGTS*(5.5)	*TGTS*(5.6)	*FGTS*(5.8)	*TS*(6)	*FGTS*(6.2)	*FGTS*(6.4)
111–134	*109.2*	145.3	128.9	136.6	136.3	151.2	146.0	164.4	148.6
	0.51	0.65	0.52	0.62	0.55	0.61	0.65	0.73	0.66
A–C	1328	1633	1433	1209	1694	2347	*1078*	1890	2076
	16.28	16.78	12.72	12.24	12.47	16.81	*10.74*	18.22	20.06

Table 2. Comparison of selection operators for ORLIB instances 111–134, A–C

The tables contain results for: rank-based selection (RBS), tournament selection (TS) with N_{tour} = 5, 6 and fine grained tournament selection (FGTS) with F_{tour} = 4.5, 5.5, 5.6, 5.8, 6.2, 6.4). Every cell in table has two values. Upper

value is average number of generations and lower value is average running time (in seconds).

Inst.	*RBS*	*FGTS*(4.5)	*TS*(5)	*FGTS*(5.5)	*TGTS*(5.6)	*FGTS*(5.8)	*TS*(6)	*FGTS*(6.2)	*FGTS*(6.4)
MO	112.4	78.6	85.2	96.4	83.6	92.2	96.7	*83.5*	87.4
	0.59	0.46	0.36	0.49	*0.35*	0.38	0.49	0.43	0.44
MP	181.7	131.6	128.6	127.9	131.5	122.8	131.3	*122.6*	130.7
	1.18	0.83	0.67	0.79	0.55	*0.51*	0.81	0.76	0.81
MQ	269.9	205.9	206.2	208.6	*192.1*	204.4	210.4	207.0	200.3
	2.69	1.51	1.28	1.51	*0.93*	0.99	1.50	1.47	1.42
MR	423.7	347.0	357.0	340.6	343.4	345.7	*332.4*	345.6	340.7
	4.68	3.47	3.14	3.31	*2.52*	2.53	3.19	3.31	3.24
MS	879.4	746.9	732.4	756.1	723.6	766.2	743.8	740.5	*721.3*
	23.18	17.24	15.78	16.77	*14.47*	14.94	16.31	16.16	15.65

Table 3. Comparison of selection operators for generated instances

In all executions of every program instance same result is obtained that is equal to optimal or previously best known solution. Best times for each instance group in table are given in the italic typeface.

It becomes clear from obtained results that almost all ORLIB instances have not sufficient size to properly test behavior of algorithm on large-scale instances. Thus, selection strategies are tested on problem instances generated and described in [6].

In most cases, best results (or results that are very close to best ones) are produced by fine grained tournament selection, with $F_{tour} = 5.6$. Improvement to other selection method is significant (usually 10%–20%) and for some (mainly large-sized) instances it is even bigger.

REFERENCES

[1] D. E. Goldberg, *Genetic algorithms in search, optimization and machine learning*, Reading, Mass., USA, Addison-Weseley Publ. Comp., 1989.

[2] J. Holland, *Adaptation in Natural and Artificial Systems*, The University of Michigan Press, 1975.

[3] J. Kratica, *Improving performances of the genetic algorithm by caching*, Computers and Artificial Intelligence, **18**, 3 (1999), 271–283.

[4] T. Blickle and L. Thiele, *A mathematical analysis of tournament selection*, in: *Proc. of the Sixth International Conference on Genetic Algorithms*, pp. 9-16, San Mateo, California, 1995.

[5] V. Filipović, *Proposition for improvement tournament selection operator in genetic algorithms*, MS thesis, Faculty of Mathematics, Belgrade, 1998 (in Serbian).

[6] J. Kratica, *Parallelization of genetic algorithms for solving some NP-complete problems*, PhD thesis, Faculty of Mathematics, Belgrade, 2000 (in Serbian).

[7] Č. Darvin, *Postank vrsta*, Nolit, Beograd, 1985 (in Serbian).

University of Belgrade, Faculty of Mathematics, Studentski trg 16, 11000 Belgrade, Yugoslavia
E-mail: {dtosic|vladaf}@matf.bg.ac.yu
SANU, Mathematical Institute, Kneza Mihaila 35/I, 11001 Belgrade, p.p. 367, Yugoslavia
E-mail: jkratica@mi.sanu.ac.yu URL: http://www.geocities.com/jkratica/