Multi-heuristic and game approaches in search problems of the graph theory

B.F. Melnikov[1], E.A. Melnikova[1], S.V. Pivneva[2], N.P. Churikova[3], V.A. Dudnikov[2], M.Y. Prus[4]

[1]Russian State Social University, Wilhelm Pieck str., 4, Moscow, Russia, 129226
[2]Togliatti State University, 16, Belorusskaya str., Togliatti, Russia, 445020
[3]Samara National Research University, 34, Moskovskoye shosse, Samara, Russia, 443086
[4]State Fire Academy of EMERCOM of Russia, Boris Galushkin str., 4, Moscow, Russia, 129366

Abstract. We consider in this paper the adaptation of heuristics used for programming non-deterministic games to the problems of discrete optimization, in particular, some heuristic methods of decision-making in various discrete optimization problems. The object of each of these problems is programming anytime algorithms. Among the problems solved in this paper, there are the classical traveling salesman problem and some connected problems of minimization for nondeterministic finite automata. Considered methods for solving these problems are constructed on the basis of special combination of some heuristics, which belong to some different areas of the theory of artificial intelligence. More precisely, we shall use some modifications of unfinished branch-and-bound method; for the selecting immediate step using some heuristics, we apply dynamic risk functions; simultaneously for the selection of coefficients of the averaging-out, we also use genetic algorithms; and the reductive self-learning by the same genetic methods is also used for the start of unfinished branch-and-bound method again. This combination of heuristics represents a special approach to construction of anytime-algorithms for the discrete optimization problems. This approach can be considered as an alternative to application of methods of linear programming, and to methods of multi-agent optimization, and also to neural networks.

Keywords: incomplete branches and bounds method, heuristic algorithms, graphs, invariants, tournaments.

1. Introduction. A brief survey of the discrete optimization problems

We consider in this paper the adaptation of heuristics used for programming non-deterministic games to the problems of discrete optimization, in particular, some heuristic methods of decision-making in various discrete optimization problems (DOP). The object of each of these problems is programming anytime algorithms, i.e., the algorithms, which can provide a near-to-optimal solution in real time. The basic purposes of the paper are practical questions of construction of algorithms, as well as the creation of the corresponding theory. Let us briefly list the considered problems, more precisely, the classes of considered problems.

First, it is the classical traveling salesman problem (TSP; [1] etc.). Certainly, an universal methods for solving TSP simply cannot exist. Some last years, the authors of papers for heuristic methods of TSP-solution consider most often so-called *metric TSP*. For their solving, some methods (of linear programming, of multi-agent optimization, etc.) are used; [1, 2, 3, 4, 5] etc. However, the classical branch-and-bound method (BBM) can also be used not only for the exact (optimal) solution of considered TSP, but also for quasi-optimal heuristic solutions. We shall

write below about these things more detailed.

Second, these are some related problems of minimization for nondeterministic finite automata (Rabin-Scott automata, NFA). Probably, the main for them is state-minimization, i.e., the problem of constructing NFA, which defines the given regular language and has minimum possible number of states. Since 1970 (i.e., since [6]), there are a few changes in description of the exact algorithms for this problem: all the algorithms are exponential relative to the number of states of considered NFA. The last argument is true because all the algorithms need to construct equivalent automaton of canonical form (or, maybe, some similar graphs or other objects). Let us remark, that from the point of view of the theory of complexity of algorithms, all the algorithms of [6, 7, 8, 9, 10, 11] are equivalent.

Besides, there are other problems for NFA-minimization, the following ones:

- edge-minimization [12, 13];
- and also the star-height-minimization. There exists two solutions of the last problem ([14], and also [15] with the simplification of the proof [16]). However, the authors think that there is impossible *to make a computing algorithm* on basis of these papers.

Third, this is the problem of minimization of disjunctive normal forms (DNF). The exact algorithms for this minimization are obtained for ages (and are considered in the classical textbooks for first-year students), however the computer programs making on basis of such solutions cannot work in real time even for the number of variables, which is equal to 20, except, certainly, for a lot of trivial cases. The unified approach of this paper is used in some versions of computer programs.

Let us remark, that, certainly, these groups of problems do not formulate the whole set of problems, which can be heuristically solved by the methods considered in this paper. Let us also mention, for instance, [17]. Some other groups of problems are given in the conclusion, and, probably, each DOP can be solved in such a way.

The methods of solution DOP, considered in this paper, are constructed on the basis of special combination of some heuristics, which belongs to some different areas of the theory of artificial intelligence. Firstly, we shall use some modifications of unfinished branch-and-bound methods. Secondly, for the selecting immediate step using some heuristics, we use dynamic risk functions. Thirdly, simultaneously for the selection of coefficients of the averaging-out, we also use genetic algorithms. Fourthly, the reductive self-learning by the same genetic methods is also used for the start of unfinished branch-and-bound method. Let us consider now the detailed description of these heuristic methods of DOP-solution.

2. Unfinished branch-and-bound method

We shall consider the branch-and-bound method for arbitrary DOP; however, sometimes we shall consider BBM-examples for the most known DOP, i.e., TSP. We shall use for it the following names:

- “accidental TSP”, when all the elements of the TSP-matrix are generated by the variate having the given equipartition law;
- “metric TSP”, when we consider towns as the accidental points of the unit square (both the coordinate have the given equipartition law), and the elements of the TSP-matrix are their distances. And here is also evident the following symmetric condition: $a_{ij} = a_{ji}$ for each possible i and j.
- “quasi-metric TSP”, when all the elements of the metric TSP-matrix are multiplied a posteriori by a random number, which is obtained by a given normal low; see [18] etc.

Some last years, the papers for metric TSP are mostly published. The consideration of metric TSP as the problems of linear programming or the applications of so-called methods of multi-agent optimization was started long before 2000, [19, 20]. And the quasi-metric TSP, which is almost not considered, is more interesting, because of the following:

- first, it is more closely to various practical problems;
- second, various heuristics can be checked up here, which are not connected to use of an arrangement of cities on a plane; moreover, the reduction of this problem to a problem of linear programming is here ineffectively;
- and third, the simplification of the TSP-matrix by one step of BBM is here "less significant", than in other TSP-variants.

Therefore, the metric TSP is the most important scope for algorithms considered in this paper.

In [1], the only exact BBM was considered; it finishes by constructing the optimal solution. And in practice, we can rarely obtain the exact TSP-solution using only algorithms of [1]. Using some special programming techniques (e.g., special data structures for quick making the next step of BBM, organization of swapping by the programmer), we can only a little improve the situation. In fact, each of such programming techniques is a new heuristics, which is used in addition to considered applying BBM. However, we are considering the exact solution of TSP (and other DOP) only, and, for now, are not considering the things connected with anytime algorithms. Before the formulation such algorithms, let us consider the following definitions.

Considering a BBM-step, we have to designate the problem for the next solution, obtained by reduction of dimension, exactly *the right problem* (like [1] etc.). For example, the right problem of TSP is obtained when we include the edge between two considered towns; and the right problem of NFA-minimization is obtained when we include the selected grid (see [9] etc.). The other alternative, i.e., when we make a decision about the absence of some element in the optimal solution (e.g., if we consider TSP not containing the edge between two considered towns) is designated by *the left problem*. It is evident, that the object of each modification of BBM (for each DOP) is to obtain the case, when the probability of belonging the optimal solution in the right problem is more then the same probability for the left problem; *we make this thing using some special heuristics*. The explanation of this fact is trivial: dimension of the right problem is less than the left one.

The simple heuristics, which reforms the usual BBM into the unfinished one (and on basis of unfinished BBM we construct any anytime algorithms in this paper), is the following. Each time, when we obtain the next right problem (let us call it problem T) we make at the same time also the sequence of the right (sub)problems (SRP), i.e., T, then the right problem of T, then the right problem of the right problem of T, etc. Certainly, we make each time also the corresponding left problems, i.e., the left problem of T, then the left problem of the right problem of T, etc. This process finishes:

- when we obtain a trivial problem (e.g., of dimension 1), then we use its solution (i.e., its bound, and also the obtained path, and similar behavior) by the current quasi-optimal solution of the considered anytime algorithm;
- or when we obtain the big value of the bound, for example, if this value is more than the current (existing) quasi-optimal solution.

Let us remark, that in practice such process of SRP-constructing does not require a lot of time, and the increasing the dimension of the list of problems for the solution in the future is very reasonable.

Thus, we have described the simple process of constructing the anytime algorithm on the basis of the given version of BBM. And it is unlikely, that such process is described here at the first time (it is really very simple), however, the authors have not the references for this thing.

Let us also remark, that this algorithm of SRP-constructing is used as the sub-algorithm not only for the unfinished BBM (like this section), but also for so called algorithm of tournament self-learning.

3. Nondeterministic games and dynamic estimation of a position. Dynamic risk functions in backgammon

Because of space limitations, the rules of backgammon are not presented in the paper. Among scientific works devoted to programming of this game, we mention the papers [21, 22, 23]. However, the authors of this paper hope, that the use of dynamic risk functions (DRF) considered here simplifies conventional methods of neural network programming and learning; they are an alternative to these methods.

What is the difference between backgammon (and other nondeterministic games) and, to say, chess (or other deterministic games) from the programming standpoint? The difference is that the game tree constructed for backgammon includes not only the vertexes where the players choose a next move but also those where they wait for a particular realization of a random event. Therefore, the standard minimax method is to be generalized for programming of search in nondeterministic games. In this paper, we will only briefly describe this generalization.

We assume that the reader knows the canonical minimax method. And in nondeterministic games, we have the following alternate actions:

- a particular realization of a certain random event;
- a move of one player;
- another realization of the random event;
- and a move of the other player.

The number of possible outcomes of the random event is to be finite (otherwise, we need different models). As a result, the game tree contains additional levels between those corresponding to the players' moves. These new levels correspond to the moments when the random event is realized. It is such a tree that the generalization of the minimax method.

Assume that we can construct a static estimator of a position. By temporally eliminating indeterminacy, we preliminary estimators of the game tree positions. For this purpose, we assume first that a particular outcome of the random event has been already realized and calculate the dynamic estimator of the position in the same way as in the conventional minimax method. Then, we calculate the dynamic estimator for the next outcome of the random event, and so on for all possible outcomes.

The final dynamic estimator of the position is based on the deterministic estimators of all possible outcomes of the random event. The values of the deterministic (usually, static) estimators are averaged in a special way resulting in the dynamic estimator. From the physical standpoint, such averaging gives us the coordinate of the gravity center of a one-dimensional system of masses whose values are determined by a specially chosen function (risk function). Coordinates of the masses are equal to the values of the corresponding deterministic estimator, which is determined by only deterministic factors of the game, like in the conventional minimax. Let ai be values of deterministic estimators and f be a risk function. Then, according to [24, 25], the dynamic estimator is calculated by the formula

$$\frac{\sum_{i=1}^{k} a_i \cdot f(a_i)}{\sum_{i=1}^{k} f(a_i)}.$$

Let us note, that purpose of the paper [24] was to generalize the minimax method. The important thing to note, however, is that the program based on this generalization only, with the simplest risk function for static estimation of a backgammon position, showed good results

and won most part of the programs that the authors of [24] could find that time in Internet. This program has been gradually improving since then. Note also, that the ways of improving the program were very different from those discussed in "classical" works on programming of backgammon [21, 22, 23]. In the latter works, one or another way of calculation of static estimators of a position is optimized. Some ways to improve programs, which were used by the authors after the simplest dynamic estimator had been already introduced, are described in this paper. Note that they concern not only improvements of the static estimator of a position.

The question is to what extent the weights of the opponent's casts that are favorable for us are to be reduced? Even if we simply take the risk function $y = 1 - 0.4x$ and use this function independent of any other circumstances with the simplest static position estimator, we will get rather good results. The short description of practical results can be found in [24]. Note also that in that work, different decreasing risk functions were considered.

But to get a stronger program, it is required *to change* strategies $y = 1 - 0.4x$ during the game. One way to improve the dynamic estimator of a position (described in [24]) is as follows. We qualitatively estimate the position (whether we are about to win or to loose). Then:

- if we are about to win or loose a little, we should be pessimists and adopt a risk function similar to above-mentioned function $y = 1 - 0.4x$;
- if we loose more, the risk function should be close to constant one;
- and if the loss is great, the risk function should be increasing; in this case we need to be super-optimists and hope against hope (what else can we do?).

Of course, there are many other, intermediate, variants of risk functions. And a possible approach to dynamic selecting these intermediate variants was described in detail in [25]. Thus, the authors of this paper believe that, in the given case, the methods of modifications of plots of risk functions simplify conventional methods of neural network programming and learning. This follows, for example, from the fact that one of the authors created a good program for playing backgammon using less than 3 self-learning coefficients, whereas programs described in [22, 23] (see also above-mentioned web sites) use several hundreds such parameters for neural networks.

4. The decision-making using some different greedy heuristics simultaneously. Dynamic risk functions in discrete optimization problems

As we said before, we shall to use not only the heuristics, which forms BBM (for TSP, or for some other DOP), but also the other one. Namely, it is heuristics for selecting element, which separate the considered problem into right and left sub-problems. However, we can often replace a heuristics separating algorithm for some other. Moreover, solving a DOP using BBM, it is often desirable to choose one of some separating algorithms, depend upon the solved sub-problem. The various separating algorithms can be selected depend upon dimension of the solved problem, its bound, and also many other descriptions, which are based on the solved DOP. In classical examples of using BBM for TSP ([1] etc.), some good separating algorithms were used (it means, that they give the reasonable results comparing other ones).

However, long before [1], for the BBM-branching various other heuristics were used, see, e.g., [26]. Let us mention, for example, the following heuristics for the reduced TSP-matrix:

- the total number of zeroes;
- the sum of minimums for all the rows and columns;
- the sum of some minimum values of considered row and columns multiplied by special "damnation constants".

All these values are computed by the TSP-matrix after reducing and selecting separating element (i.e., separating edge for branching). Probably, the authors mentioned here less than 10% of the heuristics used before.

Thus, how can we use the fact, that in various situations (i.e., in various sub-problems of the same considered DOP) different heuristics relatively better are used? (This question can be putted for both exact and unfinished algorithms.) We need to make a decision for selecting the separating element for branching. We have information of various experts, i.e., of various special heuristics, so called predictors (or estimators). The predictors often give discrepant information, and we have to average it in a special way. Unlike all the algorithms published before ([27] etc.), the authors, like programming nondeterministic games, use dynamic risk functions for this thing.

Since various heuristics give values of various units, we have to normalize them for computing the final result. Using the special set of normalizing coefficients (it is adjusted, for example, for genetic algorithms, we shall consider them below) is, probably, a possible method, but it is not the main one for this paper. The authors used only one algorithm in various DOP; that was a special modification of "voting method", where each of heuristics gives the considered variants of selecting (for TSP, those are zeroes of the reduced matrix, corresponding some edges, which are the candidates for branching). After that, we use special dynamic risk functions.

Thus, selecting edge for branching is constructed for BBM by using dynamic risk functions; see previous sections for details. It is important to note, that the dynamic selecting of the particular risk function is similar to selecting it in programming nondeterministic games, considered in [24, 25] and in previous section. Since we consider here DOP (not programming nondeterministic games), we have to add here a new heuristics, i.e., one for selecting "current position estimation", i.e., evaluation of the situation, which is obtained by the solving some DOP using BBM.

Thus, let us have some various heuristics for selecting the element for the next step of BBM (or, generally speaking, for selecting the strategy of solution some DOP). Let each of possible strategies have some various expert evaluations of availability (i.e., let us have some independent expert sub-algorithms, i.e., predictors). Then the concluding strategy could be chosen by maximum of average values. However, let us consider the following example (this example is connected with backgammon programming, because it uses 36 predictors). Let expert evaluations of availability have values in the segment $[0, 1]$.

(i) Let for the 1st strategy, the 1st expert has the evaluations of availability being equal to 1, and other 35 experts have the evaluation 0.055.

(ii) And for the 2nd strategy, 2 experts have the evaluation 0.95, and other 34 experts have the evaluation 0.

Very likely, each user (expert-human) on the basis of these values chooses the 2nd strategy. However, averaging-out by the simplest algorithm (i.e., simply the simple average of expert evaluations) gives 0.081 for the 1st case and 0.053 for the 2nd one; i.e., do we have to choose the 1st strategy?

Let us make computations like to [25], i.e., having the same algorithms of DRF-constructing. For the 1st strategy, we obtain the following risk function:

$$-0.685 \cdot x^2 + 1.300 \cdot x + 0.386\,,$$

and for the 2nd strategy:

$$-0.694 \cdot x^2 + 1.374 \cdot x + 0.321\,.$$

The final values of averaging-out of expert evaluations using these risk functions are 0.111 for the 1st strategy and 0.147 for the 2nd strategy. Therefore, using such algorithms of DRF for special averaging-out of expert evaluations gives "natural" answers.

Remark that twice repeated using of averaging-out (i.e., averaging-out using preliminary values of the first step of DRF-using) chooses the 1st strategy again. However, in the limit we have "natural" answers again. Let us describe these results by the following Table 1; the names of columns are equal to the number of step of averaging-out using DRF (i.e., the number of using

algorithm constructing DRF). Then the column 0 is the simple average of expert evaluations), and the column ∞ is the limit value.

Table 1. The consistent application of DRF

	0	1	2	3	4	5	...	∞
1st strategy	0.081	0.111	0.104	0.106	0.105	0.105	...	0.105
2nd strategy	0.053	0.147	0.094	0.118	0.106	0.112	...	0.110

Let us also remark, that this example is really possible in solving real problems: in the real computations for the mentioned DOP, the situations, when the difference between values of maximum and minimum expert values is more than 0.5 (i.e., more than 50% of the segment of expert values) are very often; for example, for accidental TSP having dimension 75 and some of predictors mentioned before, they contain, by statistics of the authors, about 10%.

5. Conclusion. Some results and some open problems

We can some simplistically formulate the main thing of this paper by the following way: using dynamic risk functions and connected heuristics in various discrete optimization problems. Such combination of heuristics considered in this paper, represents a special approach to construction of anytime algorithms for the discrete optimization problems. This approach can be considered as an alternative to application of methods of linear programming, and to methods of multi-agent optimization, and also to neural networks.

A reference to *the repository* of one of the projects executed in accordance with the approach given in this paper is the following: `https://github.com/va-dudnikov/nfa`.

Thus, the main thing of this paper is using DRF in various DOP. However, in each of the papers published before, the kind of these functions was chosen on the base of the self-learning only, for example, like [25]. Now, there is submitted a paper about the kind of these functions in arbitrary case; certainly, all the needed conditions for these functions should be true. Some of these conditions were already formulated also in [25]; and among the conditions, which were not formulated there, let us mention, for example, the necessity of convergence of values after some DRF-using, for all possible values of the set of values, which are averaged by DRF.

Let us consider some problems for the solution in the future. In Section 2, we said about possible canceling process of SRP-constructing. We considered there this thing in view of description of auxiliary algorithms to receive by a high probability a new quasi-optimum solution as the result of this canceling with the following exact solving of received problem. However, such canceling the process of SRP-constructing is also used for quite other object. Exactly, we can formulate some heuristics for this process successfully solves the given DOP in case of distributed computer architecture. Then the process of SRP-constructing is canceled not owing to the fact, that on base of some heuristics, we think that the considered SRP does not give a new quasi-optimum solution. Vice versa, we cancel it to solve the last right problem of SRP by the separate process. Solving of the last problem of the SRP could be made by various methods (depending, for example, on its dimension): by complete enumeration, BBM, or by some other methods, which are considered for the given DOP.

Let us mention another difficult problem for the following solution. For different sub-classes of problems, we try to use different genomes (using different genetic algorithms for the self-learning is also possible); they are analogues of classes of positions in nondeterministic games. But this is a simple problem, rather, a problem, which main complication belongs not to the programmer, but to the experts (who understand the whole specificity of the considered DOP). There would be much more important a possibility of automatic generating conditions for belonging some DOP to a class of problems, which should be considered separately from other classes (let us

also remark, that in the case of programming intelligent games, this problem is connected with a problem of automatic generating a new parameter for the function of static estimating position). After such automatic generating we use the usual self-learning by some genetic methods, and then we make a testing, whether we really have constructed a new class of sub-problems for considered DOP. Let us remark, that the possible algorithms of such testing are evident (they are similar to usual algorithms of clustering), and the first part of this problem, i.e., automatic generating conditions of a class of problems, is much more important.

Let us remark also the following fact. Using heuristics of this paper in various DOP, the authors have none example, when the optimal solution needs more than 5% of the time required for the common solution of the considered problem. This fact marginally explains all the heuristics considered in this paper, even the optimum solution is unknown, e.g., if it cannot be obtained in the reasonable time.

6. References

[1] Hromkovič, J. *Algorithmics for Hard Problems. Introduction to Combinatorial Optimization, Randomization,* Approximation and Heuristics / J. Hromkovi˘c. — Berlin: Springer, 2004. — 548 p.

[2] Gutin, G. *The Traveling Salesman Problem and its Variations* / G. Gutin, A. Punnen — Berlin: Springer, 2002. — 829 p.

[3] Applegate, D. *The Traveling Salesman Problem. A Computational Study* / D. Applegate, R. Bixby, V. Chvátal, W. Cook. — Princeton: Princeton University Press, 2007. — 608 p.

[4] Gutin, G. *A memetic algorithm for the generalized traveling salesman problem* / G. Gutin, D. Karapetyan // Natural Computing. — 2010. — Vol. 9(1). — P. 47–60.

[5] Raman, G. *Review of different heuristic algorithms for solving Travelling Salesman Problem* / G. Raman, N. Gill // Int. J. of Advanced Research in Computer Science. - 2017. - Vol. 8(5). - P. 423–449.

[6] Kameda, T. *On the state minimization of nondeterministic finite automata* / T. Kameda, P. Weiner // IEEE Transactions on Computers. — 1970. — Vol. C-19. — P. 617–627.

[7] Hashiguchi, K. *Algorithms for determining the smallest number of nonterminals (states) sufficient for* generating (accepting) a regular language / K. Hashiguchi, Albert J. L., Monien B., Artalejo M. R.(eds) // Automata, Languages and Programming. ICALP 1991. Lecture Notes in Computer Science. - Springer, Berlin, Heidelberg, 1991. - Vol 510. - P. 641–648.

[8] Jiang, T. *Minimal NFA problems are hard* / T. Jiang, B. Ravikumar // SIAM J. Comput. — 1993. — Vol. 22(6). — P. 1117–1141.

[9] Melnikov, B. *Once more about the state-minimization of the nondeterministic finite automata* / B. Melnikov // Journal of Applied Mathematics and Computing. — 2000. — Vol. 7(3). — P. 655–662.

[10] Polák, L. *Minimizations of NFA using the universal automaton* / L. Polák // International Journal of Foundations of Computer Science. — 2005. — Vol. 16(5). — P. 999–1010.

[11] Han, Y.-S. *State elimination heuristics for short regular expressions* / Y.-S. Han // Fundamenta Informaticae. — 2013. — Vol. 128. — P. 445–462.

[12] Melnikov, B. *Edge-minimization of non-deterministic finite automata* / B. Melnikov, A. Melnikova // Korean Journal of Computational and Applied Mathematics. — 2001. — Vol. 8(3). — P. 469–479.

[13] Melnikov, B. *Once more on the edge-minimization of nondeterministic finite automata and the connected problems* / B. Melnikov // Fundamenta Informaticae. — 2010. — Vol. 104(3). — P. 267–283.

[14] Hashiguchi, K. *Algorithms for determining relative star height and star height* / K. Hashiguchi // Inform. Comput. — 1988. — Vol. 78. — P. 124–169.

[15] Kirsten, D. *Distance desert automata and the star height problem* / D. Kirsten // Theoret. Informatics Appl. — 2005. — Vol. 39. — P. 455–509.

[16] Bojanczyk, M. *Star Height via Games* / M. Bojanczyk // 30th Annual ACM/IEEE Symposium on Logic in Computer Scence (LICS-2015). — 2015. — Vol. 104(3). — P. 214–219.

[17] Melnikov, B. *On a parallel implementation of the multi-heuristic approach in the problem of comparison of genetic sequences* / B. Melnikov, A. Panin // Vektor Nauki of Togliatti State University. — 2012. — Vol. 4(22). — P. 83–86. (in Russian).

[18] Makarkin, S. Geometrical methods of solving pseudo-geometrical version of traveling salesman problem / S. Makarkin, B. Melnikov // Stochastic optimization in informatics. — 2013. — Vol. 9(2). — P. 54–72. (in Russian).
[19] Dorigo, M. *Ant Colony System: A Cooperative Learning Approach to the Traveling Salesman Problem* / M. Dorigo, L. Gambardella // IEEE Transactions on Evolutionary Computation. — 1997. — Vol. 1(1). — P. 53–66.
[20] Johnson, D. *The Traveling Salesman Problem: A Case Study in Local Optimization* / D. Johnson, L. McGeoch // Local Search in Combinatorial Optimization. - John Wiley Ed. - 1997. - P. 215–310.
[21] Berliner, H. *Computer Backgammon* / H. Berliner // Scientific American. — 1980. — Vol. 243. — P. 64–72.
[22] Tesauro, G. *A Parallel Network that Learns to Play Backgammon* / G. Tesauro, T. Sejnowski // Artificial Intelligence. — 1989. — Vol. 39. — P. 357–390.
[23] Tesauro, G. *Temporal Difference Learning and TD-Gammon* / G. Tesauro // Temporal Difference Learning and TD-Gammon. — 1995. — Vol. 38(3). — P. 58–68.
[24] Melnikov, B. *A choice of strategy in nondeterministic antagonistic games* / B. Melnikov, A. Radionov // Programming and Computer Software. — 1998. — Vol. 24(5). — P. 247–252.
[25] Melnikov, B. *Heuristics in programming of nondeterministic games* / B. Melnikov // Programming and Computer Software. — 2001. — Vol. 27(5). — P. 277–288.
[26] Bellmore, M. *The Traveling Salesman Problem: A Survey* / M. Bellmore, G. Nemhauser // Operation Reasearch. — 1968. — Vol. 16. — P. 538–558.
[27] Makarov, I. *The theory of decision-making* / I. Makarov, et al. — Moscow: Nauka, 1982. — 328 p. (in Russian).