29. Castellano G., Fanelli A. M. Information granulation via neural network based learning. IFSA World Congress and 20th NAFIPS International Conference. Vancouver, Canada, 2001, vol. 5, p. 3059–3064.

30. Shabalov A., Semenkin E., Efimov S. Ensembles techniques in data mining. Saarbruecken: LAMBERT Academic Publishing, 2012, 100 p. (in Russian)

31. Shabalov A., Semenkin E., Galushin P. Automatized Design Application Of Intelligent Information Technologies for Data Mining Problems. Joint IEEE Conference "The 7th Inter-national Conference on Natural Computation & The 8th International Conference on Fuzzy Systems and Knowledge Discovery", Shanghai, China, p. 2659–2662 (2011).

32. Galushin P., Semenkina O, Shabalov A. Comparative analysis of two distribution building optimization algorithms. 9th International Symposium on Distributed Computing and Artificial Intelligence, Salamanca, Spain, 2012.

33. Vorozheykyn A., Semenkin E. Development and investigation of adaptive probabilistic genetic algorithm for multi-criterion conditional optimization. Proceedings of international theoretical and practical conferences "Intelligent systems" (AIS'08) and (Intelligent CAD) (CAD'08). Moscow, Fizmatlit, 2008, no. 1, p. 15–21. (in Russian)

34. Semenkin E., Shabalov A V. An automated design system of intelligent information technologies ensembles. Program product and systems. 2012, № 4 (100), p. 51–54. (in Russian)

35. Semenkin E., Shabalov A. Program system π-IT-on for an automated design of intelligent information technologies ensembles. XIII national conference on the artificial intelligence with international participation CAI-2012: conference proceedings. Vol. 4. Belgorod: Publishing house BSTU, 2012, p. 109–116. (in Russian)

36. Frank A., Asuncion A. (2010). UCI Machine Learning Repository. Available at: http://archive.ics.uci.edu/ml.

37. Yu J. J. Q., Lam A. Y. S., Li V. O. K. Evolutionary Artificial Neural Network Based on Chemical Reaction Optimization. In: IEEE Congress on Evolutionary Computation (CEC'2011), New Orleans, LA, 2011.

38. Bukhtoyarov V., Semenkin E., Sergienko R. Evolutionary approach for automatic design of neural networks ensembles for modeling and time series forecasting, Iadis international conference: Intelligent systems and Agents, MCCSIS, 2011.

УДК 519.8

SELF-CONFIGURING EVOLUTIONARY ALGORITHMS FOR TRAVELLING SALESMAN PROBLEM

O. E. Semenkina, E. A. Popov, O. E. Semenkina

Siberian State Aerospace University named after academician M. F. Reshetnev
31, Krasnoyarsky Rabochy Av., Krasnoyarsk, 660014, Russian Federation
E-mail: oleese@mail.ru, epopov@bmail.ru, semenkina.olga@mail.ru

This paper considers genetic algorithm (GA) and ant colony optimization algorithm (ACO) with the automated choice of operators for the travelling salesman problem solving. The choice is based on operator probabilistic rates calculated during algorithm execution. The performance comparison with other heuristics such as Lin-Kernigan heuristic (3-opt) and Intelligent Water Drops algorithm (IWDs) is fulfilled and competitive results are demonstrated.

Keywords: genetic algorithm, travelling salesman problem, ant colony algorithm.

САМОКОНФИГУРИРУЮЩИЙСЯ ЭВОЛЮЦИОННЫЙ АЛГОРИТМ ДЛЯ РЕШЕНИЯ ЗАДАЧИ КОММИВОЯЖЕРА

О. Е. Семенкина, Е. А. Попов, О. Э. Семенкина

Сибирский государственный аэрокосмический университет имени академика М. Ф. Решетнева
Российская Федерация, 660014, Красноярск, просп. им. газ. «Красноярский рабочий», 31
E-mail: oleese@mail.ru, epopov@bmail.ru, semenkina.olga@mail.ru

Рассматриваются генетический алгоритм (ГА) и алгоритм оптимизации на основе муравьиных колоний с автоматическим выбором операторов для решения задачи коммивояжера. Выбор основан на вероятностном ранжировании операторов в течение работы алгоритма. Представлено сравнение эффективности с другими алгоритмами, такими как алгоритм Лин-Кернигана и алгоритм интеллектуальных водяных капель, показаны соответствующие численные результаты.

Ключевые слова: генетический алгоритм, задача коммивояжера, алгоритм муравьиных колоний.

Considered way of GA self-configuration was introduced in [1] where its usefulness was demonstrated on benchmark problems and in applied problems of neural networks weights adjustment. This approach was then successfully used in solving real world optimization problems with algorithmically given functions and mixed variables [2]. This made the approach to be a candidate for the development of adaptive algorithms of combinatorial optimization. This paper considers self-configuring GA, self-configuring ACO and their application in one of the most known combinatorial optimization problem named Travelling Salesman problem (TSP). The traveling salesman problem belongs to the class of NP-complete problems, is often used to test the newly created algorithms of combinatorial optimization and it has a lot of applications in routing, scheduling and many other fields [3]. Performance of these algorithms is compared with other heuristics namely Lin-Kernigan heuristic and Intelligent Water Drops algorithm. Results of numerical experiments on benchmark problems show that suggested approach demonstrates competitive effectiveness.

Algorithms description

Lin-Kernighan heuristic. One of the classical methods for solving the traveling salesman problem is a local search [4], in particular so called *k*-opt algorithm (Lin-Kernighan heuristic [5]). TSP solution is presented by cyclic graph *f*. The *k*-opt neighborhood $N_k(f)$ includes all the tours which can be obtained by removing *k* edges from the original tour *f* and adding *k* different edges such that the resulting tour is feasible. The essence of the algorithm is to consider neighborhood of current solution. If there exists a graph *g* in this neighborhood with better objective function value, then g becomes current solution. The procedure is repeated as long as the current solution can be improved.

Exploring the whole $N_k(f)$ takes $O(n^k)$ operations and, thus, 2-opt and rarely 3-opt are used in practice. This paper deals with 3-opt because it is more efficient.

Intelligent water drops algorithm. Intelligent water drops algorithm (IWDs) [6] possesses a few properties of a natural water drops. The paths that a river follows have been created by a swarm of water drops. Thus, any swarm of water drops will influence the rivers path. On the other side, for a swarm of water drops, the river is the part of the environment that has an influence over it. A large influence on the movement of the river shows which type of soil and how resistant it is to the flow, as it determines the speed drops. Thus, the path of the water drops swarm depends on path of the river, type of soil and its resistance. So the formation of a natural river is the result of a competition between the water drops and the environment that resists its movement. Notice that all natural rivers are full of twists and turns. This is due to the influence of gravity which pulls the water through the path of least resistance to the lowest point.

It is assumed that each drop of water is able to transfer an amount of soil from one place to another. Furthermore, the soil is transferred from the fast parts of the river to the slow parts. This makes the fast parts deeper, allowing them to hold a greater volume of water. The quantity of soil a water drop is able to transfer depends on its velocity. Furthermore, the velocity of a water drop depends on the amount of soil in its way. The velocity of a water drop grows faster on a path with less soil. Water drops prefer a path with the least amount of soil.

On the basis of the above properties Shah-Hosseini in 2007, proposed the Intelligent water drops algorithm [6]. Every intelligent water drop (IWD) has two important properties: the amount of soil that it carries, and its velocity. For each IWD, the values of both properties, soil and velocity, may change as the IWD flows in its environment. From the mathematical point of view, the environment is a problem for river sand swarm of water drops looking for the optimal path.

Velocity of IWD, that moves from its location *i* to the location *j*, is increased by an amount

$$\Delta vel = \frac{a_v}{b_v + c_v soil^2(i,j)},$$

where parameters a_v, b_v and c_v should be chosen as positive numbers.

IWD's soil is increased by removing some soil of the path *ij*. The amount of soil added to the IWD is calculated by

$$\Delta soil(i,j) = \frac{a_s}{b_s + c_s \cdot time^2(i,j;vel^{IWD})},$$

where a_s, b_s and c_s are positive parameters. Time is calculated by the simple laws of physics for linear motion.

$$time(i,j;vel^{IWD}(t+1)) = \frac{HUD(i,j)}{vel^{IWD}}$$

Local heuristic function *HUD*(. , .) has been defined for a given problem to measure the undesirability of an IWD to move from one location to the next. For TSP it is calculated as follow:

$$HUD(i,j) = \left\| \overline{e}(i) - \overline{e}(j) \right\|$$

where $\overline{e}(i)$ is vector of coordinates, $\|\cdot\|$ – Euclidean metric.

Soil amount between i and j is updated by the amount of soil removed by the IWD by formula:

$$soil(i,j) = (1-\rho_n)\cdot soil(i,j) - \rho_n \cdot \Delta soil(i,j)$$

The soil of the IWD is increased by the amount of soil as shown below:

$$soil^{IWD} = soil^{IWD} + \Delta soil(i,j)$$

The probability of choosing location *j* after *i* is proportional to the amount of the soil on the path between locations *i* and *j* and can be calculated by formula:

$$p_i^{IWD}(j) = \frac{f(soil(i,j))}{\sum_{k \notin V_v} f(soil(i,k))},$$

where

$$f(soil(i,j)) = \frac{1}{\varepsilon_s + g(soil(i,j))}$$

and

$$g(soil(i,j)) = \begin{cases} soil(i,j) & if \quad \min_{l \in V_c}(soil(i,l)) \geq 0 \\ soil(i,j) - \min_{l \in V_c}(soil(i,l)) & otherwise \end{cases}$$

The constant parameter ε_s is a small positive number to prevent division by zero. V_c is a list of visited nodes.

Effectiveness of the algorithm depends on many parameters. Some of them (velocity updating parameters a_v, b_v and c_v, soil updating parameters a_s, b_s и c_s, global soil updating parameter ρ_{IWD}) were fixed in our experiments according to the recommendations of the algorithm's author (1, 0.01, 1, 1, 0,01, 1 and 0,9 accordingly). However, the remaining parameters required settings for a specific task (significance of the best solution in upgrading of soil matrix α, local soil updating parameter ρ_n, initial soil on each edge of the graph *InitSoil*, initial velocity of each drop (*InitVelocity*). Each parameter can take a large number of values, but in this study we did not set ourselves the aim of fine-tuning algorithm for a specific task and therefore considered only 24 variants: α = 0,1, 0,3 or 0,5, ρ_n = 0,9 or 0,7, *InitSoil = 1000 or 10000* and *InitVelocity = 20 or 200*.

Ant colony optimization algorithm. Ant colony optimization algorithm (ACO) [7] is a nature-inspired optimization meta-heuristic based on the behavior and organization of ant colonies in their search for a food. Being almost blind animals, ants anyway can find shortest path from the nest to the food. For information exchange, ants use a ferment, or more exactly pheromone, that they leave on the traversed path. The probability that the ant will choose a certain path is proportional to the amount of pheromone on it.

Solutions in ACO are represented as permutation of n cities and ants chose next city using taboo-list (list of visited cities) and pheromone matrix at every stage. ACO has some adjustable parameters: evaporation rate (ρ), relatively importance of previous search experience (α) and relatively importance of the distance between cities (β).

Pheromone trails are updated after each ant has completed a tour by formula:

$$\tau_{ij}(t+1) = \rho \cdot \tau_{ij}(t) + \Delta\tau_{ij},$$

where ρ is parameter such that $(1 - \rho)$ is evaporation and

$$\Delta\tau_{ij} = \sum_{k=1}^{m} \Delta\tau_{ij}^{k}.$$

Here $\Delta\tau_{ij}^{k}$ is an amount of the pheromone that the ant *k* leaves on the edge *ij* and can be calculated by formula:

$$\Delta\tau_{ij}^{k} = \begin{cases} \dfrac{Q}{L_k}, & \textit{if ant } k \textit{ uses edge } ij \textit{ in its tour} \\ 0, & \textit{otherwise} \end{cases}$$

where Q is a constant, L_k – length of the *k*-th ant tour.

Let $\eta_{ij} = 1/d_{ij}$ (d_{ij} is distance between *i* and *j*) be called a visibility. The probability of choosing the city *j* after *i* is a function of the distance between cities and amount of pheromone on the edge *ij* and can be expressed as follows

$$p_{ij}^{k} = \begin{cases} \dfrac{\left[\tau_{ij}(t)\right]^{\alpha} \cdot \left[\eta_{ij}\right]^{\beta}}{\sum_{k \in \text{allowed}_k} \left[\tau_{ik}(t)\right]^{\alpha} \cdot \left[\eta_{ik}\right]^{\beta}}, & \text{if } j \in \text{allowed}_k, \\ 0, & \text{if } j \notin \text{allowed}_k, \end{cases}$$

where allowed_k is a list of unvisited by *k*-th ant cities.

Some parameters, such as Q and ρ, does not significantly affect the efficiency of the algorithm, so in this paper we will consider only the parameters α and β.

Genetic algorithm. A well-known genetic algorithm (GA) [8] is based on some principals of evolution, but in the GA for the TSP a chromosome is represented as permutation of the n numbers (number of cities). That is why some standard operations have a few changes, but many adjustable parameters remain such as mutation probability, the type of selection, etc.

There are three type of selection in genetic algorithm:

1) Tournament selection (parameter is size of the tournament).

First, we select a random subset of *k* individuals from the population and then select the best solution out of this subset.

2) Fitness proportional selection.

The probability of the *i*-th individual to be selected is proportional to its fitness function value fit_i and is calculated as follows:

$$p_i = \frac{fit_i}{\sum_{j=1}^{m} fit_j},$$

where *m* is a number of individuals.

3) Rank selection (linear or exponential ranking).

Individual *i* has rank less than individual *j* ($R_i < R_j$) if *i*-th fitness function value is less than *j*-th fitness function value ($fit_i < fit_j$).

3.1) Linear ranking.

Probability of the *i*-th individual to be selected is calculated by formula:

$$p_i = \frac{R_i}{\sum_{k=1}^{m} R_k}, \text{ where } \sum_{k=1}^{m} p_k = 1.$$

3.2) Exponential ranking.

All individuals are assigned a weight according to the value of fitness function so that the best individual has weight $\omega_1 = 1$, $(k + 1)$-th individual has weight

$$\omega_{k+1} = \begin{cases} \omega_k, & \textit{if } R_{k+1} = R_k, \\ \omega_k \cdot \lambda, & \text{otherwise}, \end{cases}$$

where $\lambda \in [0;1]$.

In this case probability of the *i*-th individual to be selected is

$$p_i = \frac{\omega_i}{\sum_{j=1}^{m} \omega_j}.$$

Algorithms self-configuration. All bionic algorithms have many adjustable parameters and this is a significant disadvantage because parameter tuning is a difficult task even for the specialists. Nowadays, for elimination of this defect one applies the tuning algorithm parameters during its work or, it can be said, an adaptation [9].

In this study we have investigated the adaptive GA, which had 8 different selection variants - tournament selection with size of the tournament equals 2, 4 or 8, rank selection with a linear ranking, rank selection with exponential

ranking with parameter λ equal to 0,95, 0,8 or 0,5, and fitness proportional selection. Also it had 5 different mutation variants – very low, low, medium, high and very high. Adaptive ACO had 4 different variants of parameter α and also 4 variance of parameter β, in both cases 1, 2, 5 or 10.

Variant of each operator shall be determined separately. Let z be a number of variants of operator. In our situation, for example, $z = 8$ in case of selection operator of adaptive GA. If we take a parameter of the algorithm instead of the operator, and different values of this parameter instead of operator variants, then $z = 4$ in case of parameter α of ACO. So here the essence of adaptation will be described in terms of operators and their variants.

At the beginning of the algorithm probability of selecting all the options of the operator are the same: $p = 1/z$. At each generation, the effectiveness of each operator variant is estimated in the following way:

$$averagefitness_i = \frac{\sum_{j=1}^{n_i} f_{ij}}{n_i}, \qquad i = 1, 2, ..., z$$

where n_i is a number of individuals obtained by the i-th variant of the operator; f_{ij} is the value of the fitness function of the j-th individual obtained by the i-th variant of the operator; $averagefitness_i$ is the average value of the fitness function of individuals that were generated by the i-th variant of the operator.

Probability of the operator variant with the largest value of average fitness (i.e. the most effective) increases by $((z - 1)\cdot K)/(z\cdot N)$, while the probability of all other operator variants decreases by $K/(z\cdot N)$), where N is the number of past generations of the algorithm, K is a constant, usually equals to *2*. In addition, there must be a lower bound of the probability of the operator variant as no one of its probabilities can be equal to zero. If some probability reaches lower bound, this variant stops to give its share in the benefit of the best variant. The sum of probabilities of all variants of the same operator is always equal to 1. Thus, the probability distribution of the operator variant selection is gradually displaced towards the most effective operator variant from less effective variants.

Thereby, when the algorithm has to create the next offspring from the current population, it firstly must configure settings, i.e. form the list of operators with the use of operator probability distributions. Then the algorithm selects parents with the chosen selection operator, produces an offspring with the chosen crossover operator, mutates this offspring with the chosen mutation probability and puts it into an intermediate population. When the intermediate population is complete, the fitness evaluation is executed and the operator rates (probabilities to be chosen) are updated according to the operator's productivity, i. e. the ratio of the average offspring's fitness obtained with this operator and the offspring population average fitness.

Algorithms work analysis. Algorithms performance was compared on two well-known benchmark problems Oliver30 and Eil51. To solve these problems all heuristics have got as much resource as algorithm 3-opt required on average (i. e., 52800 objective function calculations in case of Oliver30 and 342210 in case of Eil51). Results of numerical experiments averaged on 100 runs are presented in table 1.

Figure shows one example of algorithms work on Oliver30 problem in case where the better solution (423.741) was found. Here we can see some advantage in the effectiveness of ACO that may be due to the fact that it begins its work with a solution oriented on the distances between cities, but not with a completely random as other algorithms.

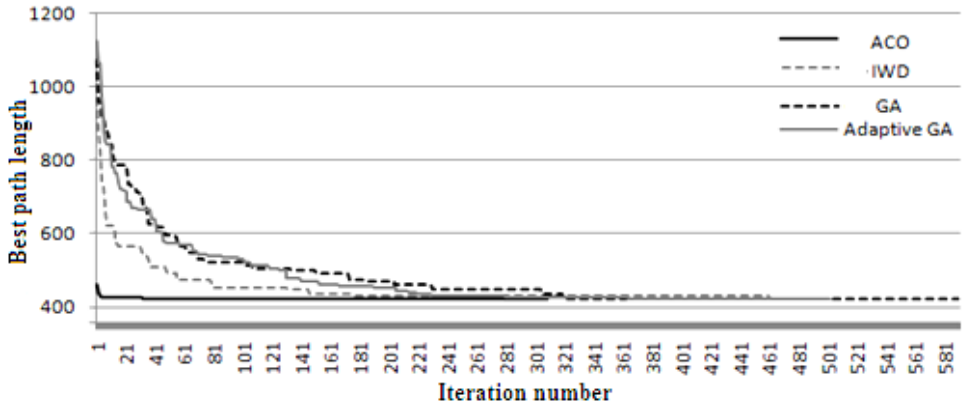
Table 1 shows that adaptive methods lose conventional with the best settings not so much. At the same time, they do not require testing many variants of parameters. Effectiveness of biology inspired algorithms on a specific task varies considerably depending on the settings.

If we solve the problem only once (that is what we actually do with real problems), the effectiveness of the algorithms will be approximately equal to the mean value on this task and not to the best value. Such a comparison on the tasks is shown in table 2, where two lines for each algorithm contain the values of the objective function, found in the best and average settings through all the settings and columns contain averaging over runs, and the standard deviation.

Table 1

Adaptive algorithms comparison with other algorithms on the tasks Oliver30, Eil51

	3-opt	IWDs	GA	Adaptive GA	ACO	Adaptive ACO
Oliver30						
best	423,741	423,741	423,741	423,741	423,741	423,741
average	428,610	425,500	432,356	434,239	423,782	426,428
std deviation	7,4740	3,7022	13,2365	13,6875	0,1675	3,7653
Eil51						
best	428,872	428,872	431,953	429,118	429,484	429,484
average	438,598	437,279	447,943	449,938	432,732	433,936
std deviation	5,0160	5,57137	7,5848	9,94143	3,11622	2,88955



Examples of the ACO, IWDs, GA and adaptive GA behavior on task Oliver30

Table 2

Adaptive algorithms comparison with other algorithms with different parameters variants on the tasks Oliver30 and Eil51

	3-opt	IWD		GA		Adaptive GA	ACO		Adaptive ACO
		best	average	best	average		best	average	
Oliver30									
Best run	423.741	423,741	423,965	423,741	424,139	423,741	423,741	428,684	423,741
Average run	434.61	426,413	434,75	431,647	442,982	434,485	424,04	443,771	426,428
Standard deviation	11.931	3,19925	10,1741	11,3784	12,8611	12,7539	0,42986	11,7151	3,7653
Eil51									
Best run	435,58	433,101	450,775	442,672	444,923	440,817	428,872	478,636	429,118
Average run	450,346	442,05	468,64	450,913	460,762	457,286	429,866	496,615	434,634
Standard deviation	8,59725	8,33447	11,6792	7,6221	10,7719	12,0611	0,74094	11,1774	3,38365

Both adaptive algorithms show good results on these problems as they outperform other algorithms with their settings giving average performance. Although on average adaptive GA cannot outperform other algorithms with their best settings and adaptive ACO cannot outperform conventional ACO with the best settings, one has to remark that we don't know beforehand which settings of the algorithm on the given task will be the best. There are 16 variants of ACO and 24 variants of GA and IWDs that means much extra efforts for the determination of these "best" algorithms before they could win adaptive GA. One can use the part of these efforts to improve results of the adaptive GA or adaptive ACO.

Besides, if we deal with real-world problems, possible situation is the best settings absence. It means that there are different best settings in the different steps of problem solving. In such cases self-configuring algorithms bring much more advantages.

Conclusion. In this paper we compared performance of some heuristic algorithms of combinatorial optimization, such as 3-opt algorithm, intelligent water drops algorithm, conventional genetic algorithm and conventional ant colony algorithm with self-configuring (adaptive) genetic and ant colony algorithms.

Our investigations demonstrate that adaptive algorithms are the effective methods of optimization with the remarkable property, which consists in the fact that the user does not have to adjust parameters but can have competitive results in solution quality.

As a future work plans, it can be comparison with other algorithms, development of adaptive versions of other algorithms and using suggested approach for solving real world problems.

References

1. Semenkin E., Semenkina M. 2012. *Self-Configuring Genetic Algorithm with Modified Uniform Crossover Operator*. Tan Y., Shi Y., Ji Z. (Eds.): Advances in Swarm Intelligence, ICSI 2012, Part 1, LNCS 7331, Springer, Heidelberg, 414–421.

2. Semenkin E., Semenkina M. 2012. *Spacecrafts' Control Systems Effective Variants Choice with Self-Configuring Genetic Algorithm.* In: Ferrier, J.-L., Bernard, A., Gusikhin, O. and Madani, K. (eds), Proceedings of the 9th International Conference on Informatics in Control, Automation and Robotics, vol. 1, p. 84–93.

3. Davendra D. (ed.) 2010. Traveling Salesman Problem, Theory and Applications. InTech.

4. Papadimitriou C. H., Steiglitz K. 1982. Combinatorial Optimization. Algorithms and Complexity. Englewood Cliffs, NJ: Prentice-Hall.

5. Lin S., Kernigan B.W. 1973. *An Effective Heuristic Algorithm for the Traveling-Salesman Problem*. Operations Res. 21, p. 498–516.

6. Shah-Hosseini, H. 2007. Problem solving by intelligent water drops. *Proc. of IEEE Congresson Evolutionary Computation*, Swissotel The Stamford, Singapore, p. 3226–3231.

7. Dorigo M., Gambardella L. M. 1997. *Ant Colony System: A Cooperative Learning Approach to the Traveling Salesman Problem.* IEEE Transactions on Evolutionary Computation, p. 53–66.

8. Eiben A. E., Smith J. E. 2003. Introduction to evolutionary computing. Springer-Verlag, Berlin, Heidelberg.

9. Schaefer R., Cotta C., Kołodziej J., Rudolph G. 2010. *Parallel Problem Solving from Nature*. PPSN XI 11th International Conference, Kraków, Poland, p. 11–15.

УДК 004.93

EMOTION RECOGNITION AND SPEAKER IDENTIFICATION FROM SPEECH

M. Yu. Sidorov[1], S. G. Zablotskiy[1], W. Minker[1], E. C. Semenkin[2]

[1]Ulm University
43, Albert-Einstein-Allee, Ulm, 89081, Germany
E-mail: maxim.sidorov@uni-ulm.de, sergey.zablotskiy@uni-ulm.de, wolfgang.minker@uni-ulm.de
[2] Siberian State Aerospace University named after academician M. F. Reshetnev
31, Krasnoyarsky Rabochy Av., Krasnoyarsk, 660014, Russian Federation
E-mail: eugenesemenkin@yandex.ru

The performance of spoken dialogue systems (SDS) is not perfect yet, especially for some languages. Emotion recognition from speech (ER) is a technique which can improve the SDS behavior by finding critical points in the human-machine interaction and changing a dialogue strategy. Inclusion of the speaker specific information, by conducting the speaker identification procedure (SI) at the set up of ER task could also be used in order to improve the dialogue quality. Choosing of both appropriate speech signal features and machine learning algorithms for the ER and SI remain a complex and challenging problem. More than 50 machine learning algorithms were applied in the study for ER and SI tasks, using 9 multi-language corpora (Russian, English, German, and Japanese) of both acted and non-acted emotional utterance recordings. The study provides the results of evaluation as well as their analysis and future directions.

Keywords: emotion recognition from speech, speaker identification from speech, machine learning algorithms, speaker adaptive emotion recognition from speech.

РАСПОЗНАВАНИЕ ЭМОЦИЙ И ИДЕНТИФИКАЦИЯ СПИКЕРА ПО РЕЧЕВЫМ СИГНАЛАМ

М. Ю. Сидоров[1], С. Г. Заблоцкий[1], В. Минкер[1], Е. С. Семенкин[2]

[1]Университет города Ульма
Германия, 89081, Ульм, Аллея Альберта Эйнштейна 43. E-mail: maxim.sidorov@uni-ulm.de, sergey.zablotskiy@uni-ulm.de, wolfgang.minker@uni-ulm.de
[2]Сибирский государственный аэрокосмический университет имени академика М. Ф. Решетнева
Российская Федерация, 660014, Красноярск, просп. им. газ. «Красноярский рабочий» 31
E-mail: eugenesemenkin@yandex.ru

Производительность диалоговых систем, основанных на естественном языке, по-прежнему находится на достаточно низком уровне, особенно для некоторых языков. Распознавание эмоций на основе речевого сигнала представляет собой подход, способный улучшить качество работы таких систем посредством определения критических точек в диалоге между человеком и компьютером и последующей адаптации диалога. Использование процедуры идентификации улучшает качество распознавания эмоций на основе речевого сигнала пользователя, так как становится возможным построение моделей эмоций конкретного человека. Выбор подходящих параметров речевых сигналов и алгоритма моделирования для задач идентификации говорящего и распо-