UDC 519.925.51
DOI: 10.20535/SRIT.2308-8893.2020.3.07

MULTICRITERIA CONDITIONAL OPTIMIZATION BASED ON GENETIC ALGORITHMS

V. SINEGLAZOV, K. RIAZANOVSKIY, O. CHUMACHENCKO

Abstract. This article takes on solving the problem of multicriteria conditional optimization. This problem is one of the most key tasks of the current time and has its application in many areas. Reuse of various existing algorithms for solving unconstrained optimization is proposed. Different methods of multicriteria unconditional optimization are reviewed. The advantages and disadvantages of each algorithm are analyzed. The algorithms modified to take into account the constraints. Additional algorithms of transition from solving an unconditional optimization problem to a conditional optimization problem are developed. A genetic algorithm SPEA2 was used to test the developed algorithms. Examples of solving the problem at hand using the aforementioned algorithms are presented. A comparative analysis of the final results was conducted.

Keywords: multicriteria optimization, conditional optimization, genetic algorithms, repairing algorithm, SPEA2, Pareto optimization.

INTRODUCTION

There are a lot of applications in existence that require solving a problem of multicriteria optimization (MCO). One of the fields used for this goal is artificial intelligence (AI) studies. An example of this is a process of neural networks "learning" [1].

At this point in time there are a lot of algorithms for solving multicriteria optimization problems. The majority of them are geared towards solving unconditional optimization problems. One of the most effective ways for this is the usage of the genetic algorithms (GA). Those stochastic methods, with a condition of a lot of individuals and learning epochs, allow for reaching good solutions. Because of modern computational methods it's possible to effectively parallelize the evolutionary algorithms. It allows for easy usage in solving a broad specter of different problems: pattern discovery, signal processing, neural network learning, etc.

The majority of GA are oriented towards solving unconditional problems of MCO, but actual, real problems always have a plethora of constraints that need to be considered, and that results in solving a problem of conditional optimization. This consideration of constraints also requires a modification of the classic GA.

This article is dedicated to development of GA modifications to allow for consideration of constraints and solving problems of conditional MCO.

PROBLEM STATEMENT

A set of allowed vector values of X variables is a limited and an enclosed set (eq. 1).

$$D_X = \{X : G(X) \geq 0\} \subseteq \{X\} = R^n,$$

where $G(X)$ is some limitative vector-function; R^n is an n-dimensional arithmetic space, where n equals the power of a set $X - |X|$.

The target vector-function $F(X) = (f_1(X), f_2(X), ..., f_{|X|}(X))$ with values in a space of targets $\{F\}$ is described in a range D_X. A problem of minimizing $F(X)$ in this range, which means minimizing each of the individual target functions $f_1(X), f_2(X), ..., f_{|X|}(X)$ (optimality criteria) (eq. 2).

$$\min_{X \in D_X} F(X) = F(X^*) = F^*,$$

where vector X^* is the solution for a problem at hand.

REVIEW

The solution to the problem of conditional optimization is based on the solution to the problem of unconditional optimization, so it is worth considering methods of unconditional optimization. At this point in time there are two main classes of algorithms: genetic and swarm algorithms. The first class has an advantage in computational difficulty and the amount of learning epochs [11], for this reason genetic algorithms are used in this study.

There are different kinds of GA for solving multicriteria optimization problems. They can be divided in two following groups [5]: lexicographical selection, alternating criterial functions algorithms and algorithms that use Pareto dominance.

The first kind includes a lexicographic tournament selection algorithm and its numerous modifications. This algorithm is also sometimes called “naïve”

It is reasonably fast and simple, but it requires classification of criteria in regards to their importance.

The second kind includes altering objective functions algorithms, which realizes VEGA algorithm (Vector Evaluated Genetic Algorithm) [16]. It is similar to “naïve” algorithms in terms of evaluating of fitness being reliant on corresponding values of various target functions.

This type of GA also includes predator-prey algorithm. It’s more complicated in terms of computing than lexicographic selection, but the results are more accurate. It’s empirically proven that with the number of generations rising, the population of prey is moving towards the Pareto front [14]. A drawback of this algorithm is a chance of losing optimal solutions. There is a number of modifications of this algorithm to remove the drawbacks [9].

The third type includes agent ranging algorithms that is based of Pareto dominance. One of the realizations is a well-known NGSA algorithm (Non-dominated Sorting Genetic Algorithm) and its modification, NGSA-II [10].

The third kind also includes the Pareto-power algorithms, that utilize agent power and it's wimpiness.

Pareto-power algorithm realizes SPEA (Strength Pareto Evolutionary Algorithm), which evolved into SPEA-2 [4].

The latest agent ranging algorithms based on Pareto-dominance allow for receiving decent approximation of the Pareto front, leaving the level covering and rarefaction untouched, which gives more possible choices of the appropriate solution.

SPEA-2 method is unique and advantageous in the following:

- it includes all of the listed approaches in one algorithm;
- fitness of each individual of the population in this method is decided only in relation to the individuals of the outside set, with no correlation to dominance of individuals in the population;
- despite "the best" individuals of the previous generations being stored in an outside set, all of them are legible for the selection process;
- to prevent premature convergence, SPEA uses a special mechanism for creating niches, where dividing on the basis of fitness is done not because of the distance between individuals, but by Pareto-dominance.

Because of reasons stated above, this study utilizes SPEA2 algorithm.

The provided evolutionary algorithms are good at managing unconditional problems of multicriteria optimization, but during solving problems with constraints, the results are not always decent enough [2, 3]:

- it is possible for them to not include the point of conditional maximum;
- the resulting points may be separated in the search area;
- a part of the solutions may lay beyond the margins of the allowed area.

Based on this, the conclusion would be that the evolutionary algorithms aren't fit enough for solving problems with constraints and require some modifications to be made, those modifications taking into account the specifics of the conditional optimization problem.

Going forward to the problem of conditional optimization, it's possible to single out the following methods of solving it [12].

The first method is based on translating the problem of a conditional optimization to the problem of an unconditional optimization:

- penal functions algorithm [13]. For GA, special algorithms for this method, there is a method of "lethal" penalties, a method of statistical penalties and a method of dynamic penalties;
- algorithm based on sliding admittance [12].

The second method includes algorithms that do not use reduction of the problem down to a problem of the unconditional optimization:

- each restriction is transformed into a separate target function [6–8];
- the procedure of "repairing" using the local search [17, 18, 20];
- Orvosh-Davis reduction method [15];
- modified complex algorithm [19].

There are also hybrid methods. For example, method of the behavioral memory, "repairing" + lethal penalties [12].

The quality of the chosen solutions is dependent on choosing the period of multicriteria optimization algorithm, where the procedure of "repairing" is used.

PROBLEM SOLUTION

To solve the problem of a conditional MCO, the following algorithms were developed and researched:

1. Removal of unfit solutions after the algorithm completed.
2. Removal of unfit solutions after each generation of the population.
3. Taking into account the amount of violations by individuals in Pareto-dominance.
4. Transforming the constraints into additional criteria and solving a multicriteria problem with additional criteria.
5. Solving a multicriteria optimization problem without taking the constraints into account and "repairing" the unfit solutions after the algorithm completed.
6. Transforming the constraints into criteria, solving the multicriteria problem with additional criteria and "repairing" the unfit solutions after the algorithm completed.
7. Solving a multicriteria optimization problem without taking the constraints into account and "repairing" the unfit solutions after each iteration.
8. Transforming the constraints into criteria, solving the multicriteria problem with additional criteria and "repairing" the unfit solutions after each iteration.

For more extensive research and comparison between the aforementioned algorithms, the following is a detailed description of each algorithm.

1. Removal of unfit solutions after the algorithm completed.

This method is the simplest and the easiest. After all the generations of populations, the deletion of unfit individuals follows. But because of the simplicity the amount of fit solutions is not high.

2. Removal of unfit solutions after each generation of the population.

This method is a modification of the former one. The deletion of unfit individuals after every generation is assumed, but, with the usage of the selection operator, a new population is generated with the same size. This way a new population is only generated using the feasible solutions. With large amount of iterations, the amount of fit end-results will be close to the starting population size.

3. Taking into account the amount of constraints violated by individuals in Pareto-dominance.

After every iteration, before computing the fitness-function each individual of the population is checked for fitting in the constraints. If the individual is unfit, it's marked not feasible, and the amount of constraints that it's not fit for is saved.

Then, the modification of Pareto-comparison during the formation of fitness-function follows. The change is, initially the comparison between two individuals their fitness is checked. If one of them violates the constraints, and the other one does not, the former is considered dominated by the latter, without taking into

account it's values of criteria vector. If both of them are unfit, the dominated one is chosen on the basis of the amount of constraints it's unfit for. If both are feasible, Pareto-comparison between their criteria vectors follows.

In the process of generation of generations, the amount of individuals that are unfit is reduced, and in large amounts of generations are geared towards 0. But after the learning process, there may be unfit individuals left, so they are marked as unfeasible and simply removed from the end solution.

4. Transforming the constraints into additional criteria and solving a multicriteria problem with additional criteria.

In this method, the transformation of constraints into additional criteria is used. This way, the problem of multicriteria optimization is transformed and takes the following form:

The initial problem: Target functions — $F(X) \to \text{opt}$, constraints — $G(X) \le B$.

The transformed problem: Target functions — $F(X) \to \text{opt}$, $(G(X) - B)^2 \to \text{opt}$.

Next, the basic SPEA2 algorithm is used.

5. Solving a multicriteria optimization problem without taking into account constraints and "repairing" the unfit solutions after the algorithm completed

This method uses the unfit solution "repairing" algorithm. The "repairing" is conducted using the Pareto local search method [17, 18, 20]. In this method of solving the conditional multicriteria problem utilizes learning all of the populations without taking into account the constraints, and "repairing" all points that are unfit after.

6. Transformation of constraints into criteria, solving the multicriteria problem with multiple additional criteria and "repairing" the unfit solutions after the algorithm completed.

This method also utilizes "repairing" the points at the end, but the learning functions not only for the target criteria, but also with the transforming the constraints into criteria, as it is described in method 4.

7. Solving a multicriteria optimization problem without taking the constraints into account and "repairing" the unfit solutions after each iteration.

This method utilizes learning only on the target functions, but the "repairing" is used after each generation.

8. Transforming the constraints into criteria, solving the multicriteria problem with additional criteria and "repairing" the unfit solutions after each iteration.

The last method uses the "repairing" procedure after every iteration, the constraints are converted into additional criteria.

RESULTS

To analyze the results and to compare the a for ementioned methods, the following conditional multicriteria optimization problem was solved:

$$\text{Minimize} = \begin{cases} f_1(x,y) = 2 + (x-2)^2 + (y-1)^2, \\ f_2(x,y) = 9x - (y-1)^2. \end{cases}$$

$$\text{Constraints} = \begin{cases} g_1(x,y) = x^2 + y^2 \le 225; \\ g_2(x,y) = x - 3y + 10 \le 0. \end{cases}$$

$$\text{Variables constraints} = \begin{cases} -20 \le x; \\ y \le 20. \end{cases}$$

Real Pareto front looks as shown on Fig. 1.

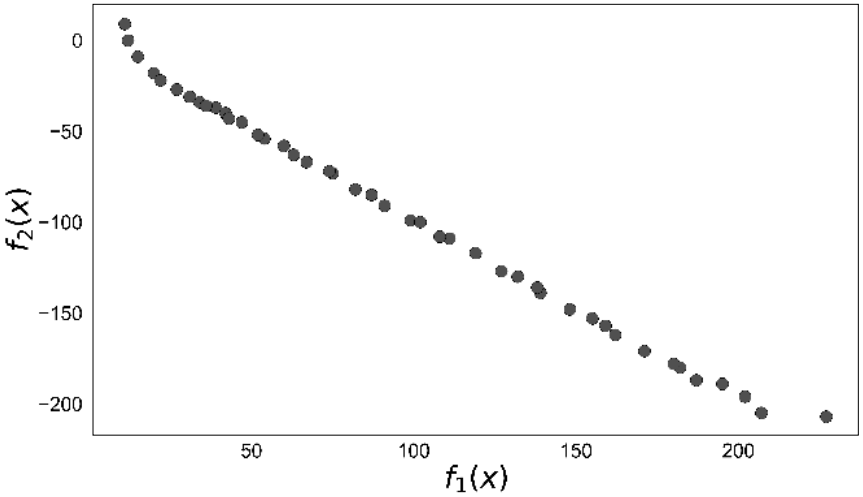

Fig. 1. Real Pareto front of the solved problem

Firstly, consider the detailed results and examples of SPEA2 algorithm working using the 3rd method, *Taking into account the amount of constraints violated by individuals in Pareto-dominance,* to solve the current problem. The amount of individuals in the population is 5, the amount of generations – 20. For each variable memory allocated is 5 bits. Integer numbers are coded into bits, and then into Grey code.

The following is the work of one iteration of the SPEA2 algorithm. For some of the following generations there will be a table of values of some individuals.

Initially, the following random population was generated:

```
10100 11111                    01110 11001
11101 00001                    00010 01100
11111 00101
```

Firstly, each iteration is allocated a "parent" for new populations, using a selection operator, that is presented in SPEA2 with a binary tournament.

Binary tournament #1.

Two individuals are consecutively, yet randomly chosen:

11111 00101 и 11111 00101. In this case, the individuals chosen are the same, so there's no better one among them. This individual will be the first "parent".

Binary tournament #2.

The chosen individuals: 11111 00101 и 00010 01100.

The first individual is dominated by the second, so it is chosen as the second "parent".

Using the operator "crossover" we get two descendants: 01111 01100 and 10110 00101.

This way, repeating the aforementioned algorithm, 6 more descendants are generated and, with a certain probability, mutated. Then, the values of the target functions and constraints are calculated. The intermediary results are in Table 1.

Table 1. Results of the 1st generation offspring

Variables		Objectives		Constraints	
10001	00111	102	54	125	35
10100	11101	127	-85	160	-22
01000	01110	51	-45	26	2
10101	11101	132	-76	169	-21
11011	00101	43	-43	20	20
00100	00101	252	-142	185	9

Those are added to the current population. The result is a sum of 11 individuals. Next step is to calculate the values of fitness-functions, which are lower than 1. Going forward, if the size of the population is lower than the preassigned value, the dominated elements with the best fitness-function are added, else – the worst individuals are removed. The new population is shown in Table 2.

Table 2. Details of the 2nd population

Gray binary x	x	**Gray binary** y	y	$f_1(x,y)$	$f_2(x,y)$	$R(i)$	$D(i)$	$F(i)$	$g_1(x,y)$	$g_2(x,y)$	**Passes**
10100	4	11111	11	106	-64	0	0,013	0,013	137	-19	True
01110	-9	11001	7	159	-117	0	0,013	0,013	130	-10	True
11101	2	00001	-9	102	-82	8	0,015	8,01	85	39	False
00010	-17	01100	-2	372	-162	9	0,003	9,003	293	-1	False
11111	1	00101	-4	28	-16	6	0,009	6,01	17	23	False

After the operations and generation of the new generations, as well as SPEA2 algorithm usage are concluded, the resulting individuals are only non-dominated. The examples of populations on 5th and on the last 20th generations are shown in Table 3 and Table 4 respectively.

Table 3. Details of the 5th population

Gray binary x	x	**Gray binary** y	y	$f_1(x,y)$	$f_2(x,y)$	$R(i)$	$D(i)$	$F(i)$	$g_1(x,y)$	$g_2(x,y)$	**Passes**
11110	0	11100	13	150	-144	0	0,026	0,026	169	-29	True
10100	4	11111	11	106	-64	0	0,5	0,5	137	-19	True
10100	4	11110	10	87	-45	0	0,034	0,034	116	-16	True
10100	4	11101	12	127	-86	0	0,031	0,031	160	-22	True
10100	4	11111	11	106	-64	0	0,5	0,5	137	-19	True

Table 4. Details of the 20th population

Gray binary x	x	**Gray binary** y	y	$f_1(x,y)$	$f_2(x,y)$	$R(i)$	$D(i)$	$F(i)$	$g_1(x,y)$	$g_2(x,y)$	**Passes**
11010	-1	01011	3	15	-13	0	0,081	0,081	10	0	True
11000	-4	11001	7	74	-72	0	0,046	0,045	65	-15	True
11010	-1	11001	7	47	-45	0	0,038	0,038	50	-12	True
11101	2	11101	12	123	-103	0	0,015	0,014	148	-24	True
11000	-4	11101	12	159	-157	0	0,008	0,008	160	-30	True

It should be noted that, after using this method, every individual fits. A graphical illustration of the real Pareto front and 5 points, the trained individuals, are on Fig. 2.

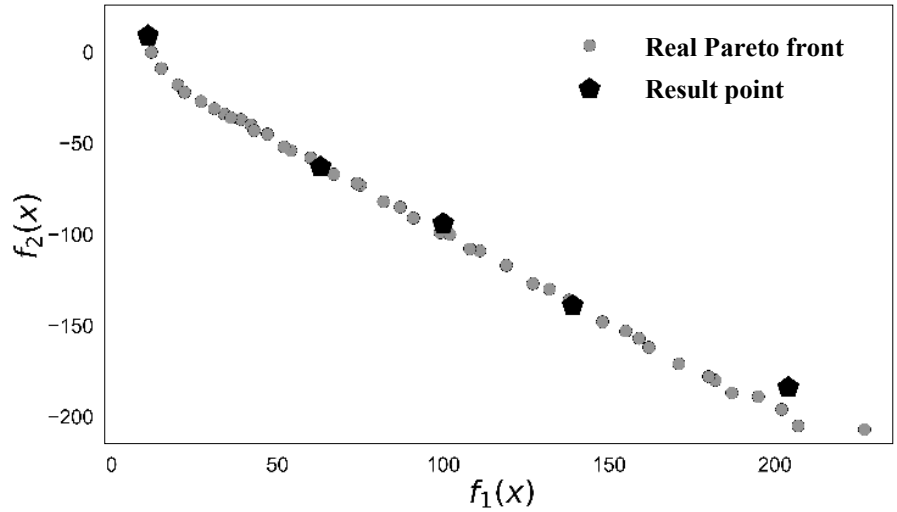

Fig. 2. Real Pareto front and five result individuals

As it is shown, even a small amount of iterations allows for great resulting solutions to a multicriteria optimization problem.

The following is an example of work and the results of each described method on the problem. Each algorithm initially used 100 individuals.

1. Removal of unfit solutions after the algorithm completed.

After learning a population of 100 individuals, only 27 fitted. Fig. 3 shows all of the individuals, and Fig. 4 shows only the fit ones.

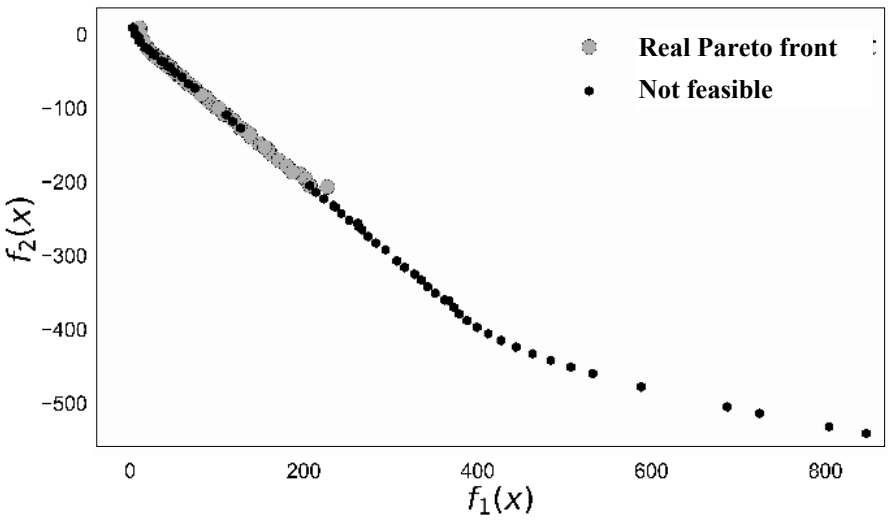

Fig. 3. Unsatisfactory individuals for the 1st method

2. Removal of unfit solutions after every generation of the population.

In the end, 43 unique solutions are correct. Fig. 5 is visualization.

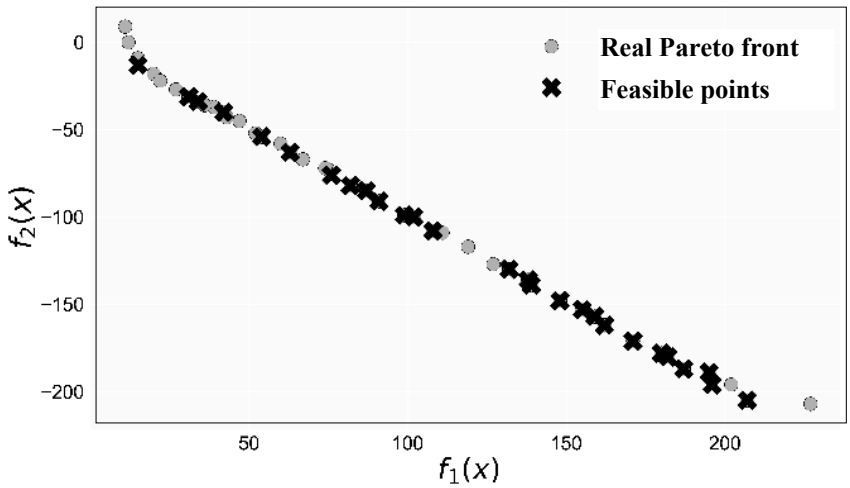

Fig. 4. Resulting individuals satisfying constraints for the 1st method

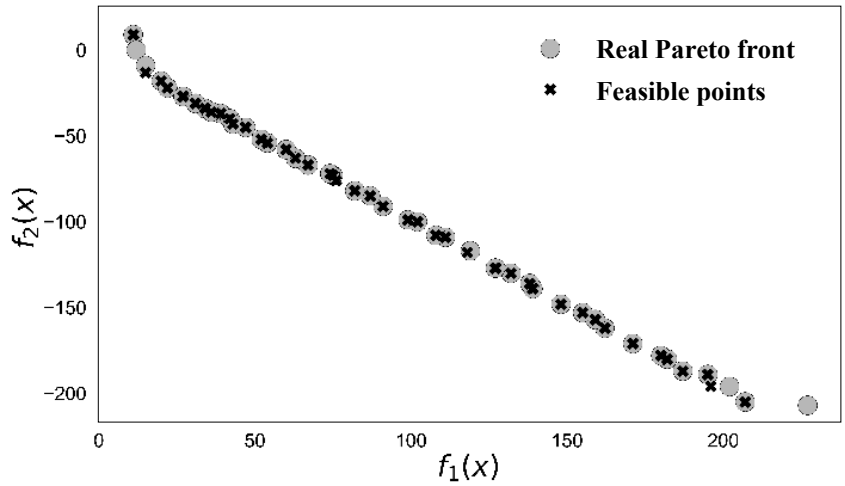

Fig. 5. Resulting individuals for the 2nd method

3. Taking into account the amount of violations by individuals in Pareto-dominance.

This method allows us to get a population with all points being fit, but only 42 of them are unique. The data is shown on Fig. 6.

4. Transforming the constraints into additional criteria and solving a multicriteria problem with additional criteria.

A simple transformation of constraints into criteria shows bad results. Only 19 non-dominated points under 4 criteria (2 real + 2 constraints) have passed. Of them, only 7 are non-dominated by two target criteria. In the end, 93 results were removed. Fig. 7 visualizes the full population, and Fig. 8 shows the end results.

5. Solving a multicriteria optimization problem without taking the constraints into account and "repairing" the unfit solutions after the algorithm completed.

In this case the "repairing" procedure allows us to get the fit points, but all of them are not better than the original set of fit points 9 unique results were acquired. Fig. 9 and Fig. 10 contains the visualization.

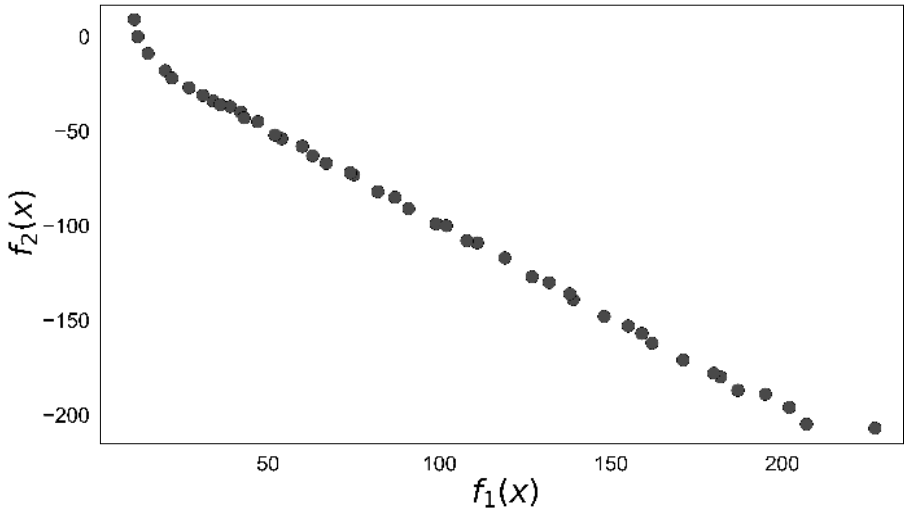

Fig. 6. Resulting population for the 3rd method

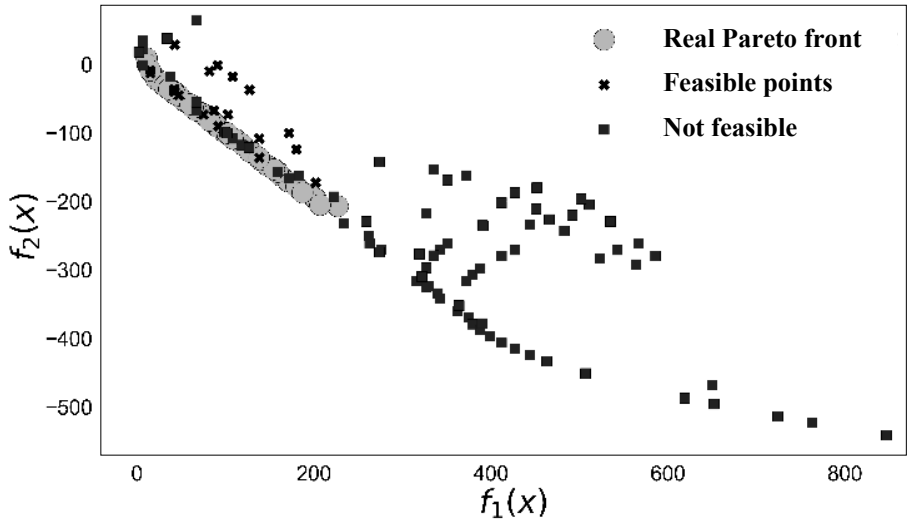

Fig. 7. Resulting population for the 4th method

6. Solving a multicriteria optimization problem without taking the constraints into account and "repairing" the unfit solutions after the algorithm completed.

In this case the "repairing" procedure allows us to get the fit points, but all of them are not better than the original set of fit points. 9 unique results were acquired. Fig. 9 and Fig. 10 contains the visualization.

7. Transforming the constraints into criteria, solving the multicriteria problem with additional criteria and "repairing" the unfit solutions after the algorithm completed.

In this case of an additional transformation into criteria the procedure of "repairing" allows for better results: 34 cured solutions, 11 of them are unique,

19 satisfactory solutions and overall result is 12 solutions. Fig. 11 shows the unfit individuals, and Fig. 12 shows the end results.

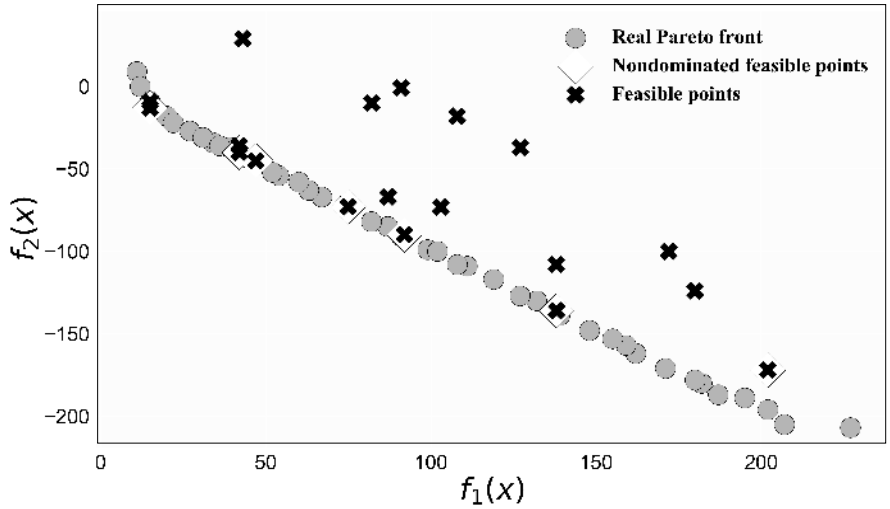

Fig. 8. Resulting non-dominated individuals for the 4th method

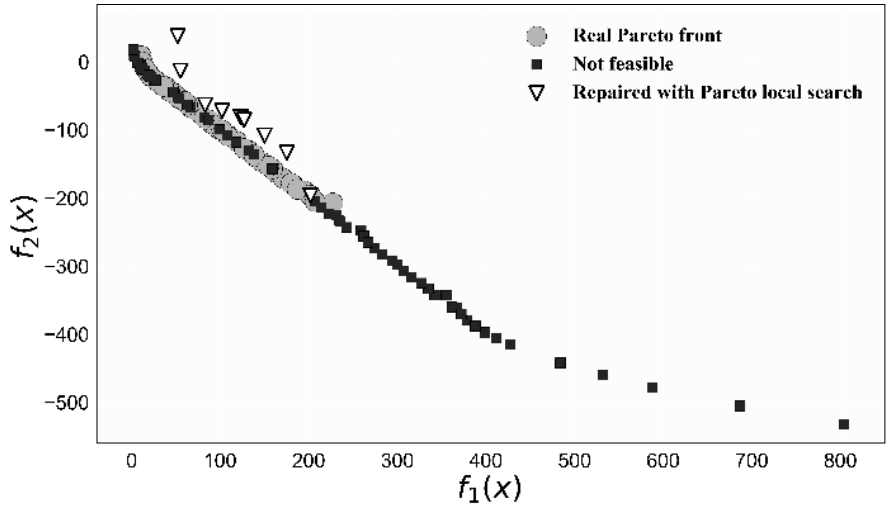

Fig. 9. Not feasible and repaired individuals for the 5th method

In this case of an additional transformation into criteria the procedure of "repairing" allows for better results: 34 cured solutions, 11 of them are unique, 19 satisfactory solutions and overall result is 12 solutions. Fig. 11 shows the unfit individuals, and Fig. 12 shows the end results.

8. Solving a multicriteria optimization problem without taking the constraints into account and "repairing" the unfit solutions after each iteration.

Learning without taking constraints into account and "repairing" after each iteration allows for decent results: 42 unique end solution. Fig. 13 is a visualization.

9. Solving a multicriteria optimization problem without taking the constraints into account and "repairing" the unfit solutions after each iteration.

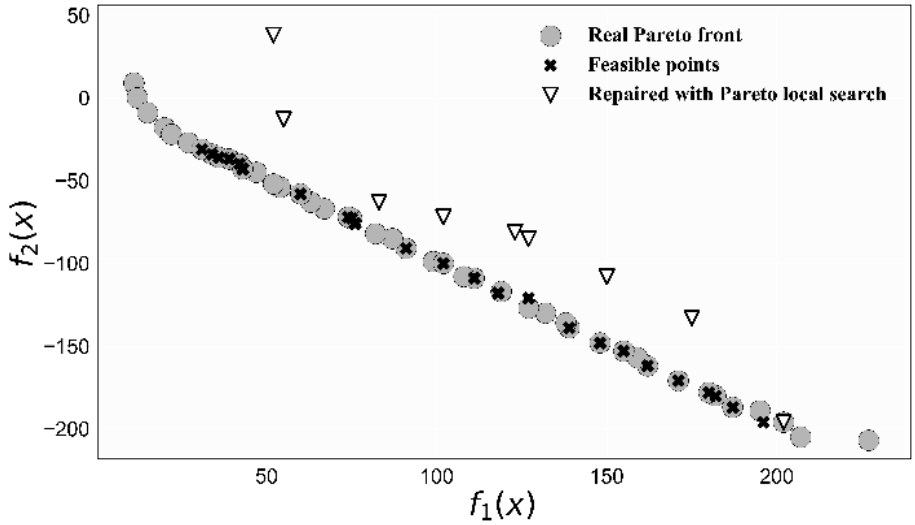

Fig. 10. Resulting individuals for the 5th method

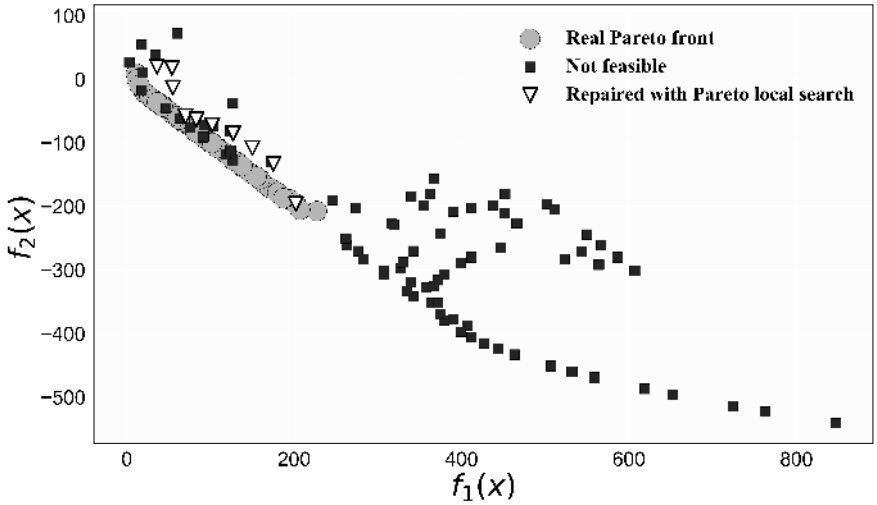

Fig. 11. Not feasible and repaired individuals for the 6th method

Learning without taking constraints into account and "repairing" after each iteration allows for decent results: 42 unique end solution. Fig. 13 is a visualization.

10.Solving a multicriteria optimization problem without taking the constraints into account and "repairing" the unfit solutions after each iteration.

Learning without taking constraints into account and "repairing" after each iteration allows for decent results: 42 unique end solution. Fig. 13 is a visualization.

11.Transforming the constraints into criteria, solving the multicriteria problem with additional criteria and "repairing" the unfit solutions after each iteration.

This method allows to get the whole population non-dominated by 4 criteria, but by the target criteria only 24 individuals are non-dominated. (Fig. 14)

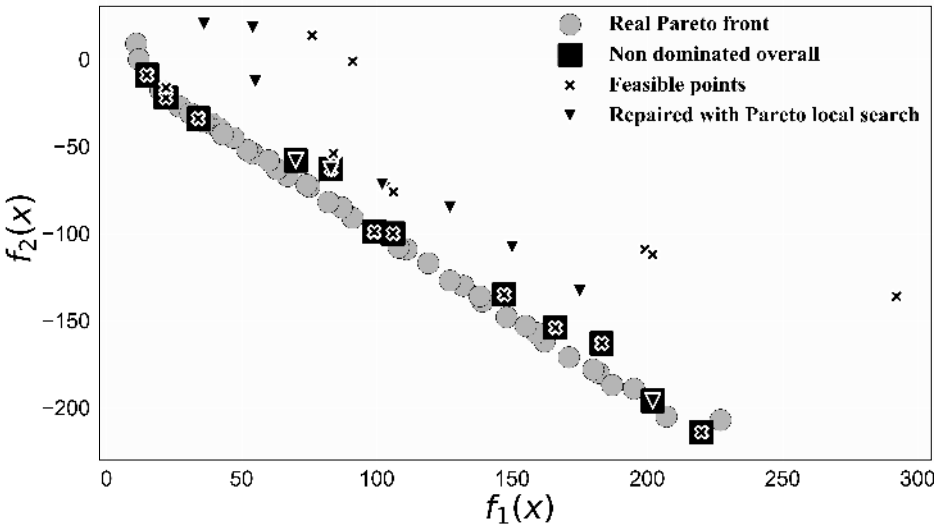

Fig. 12. Non-dominated resulting individuals for the 6th method

These results suggest that of the methods used, the best one is also one of the simplest: removal unfit individuals after each iteration. This method also does not require major SPEA2 modifications.

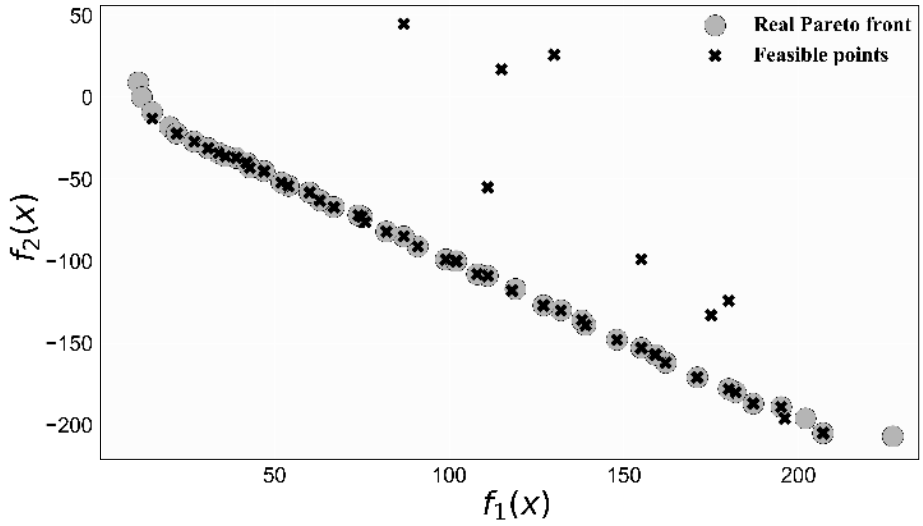

Fig. 13. Resulting non-dominated individuals for the 7th method

The same results were achieved using a method of saving the amount of violated constraints for further dominance checking during the computing of the fitness-function. This method requires some modifications to SPEA2, but those are minor and do not complicate the algorithm too much.

Another method got results, similar to the aforementioned ones. It's a method of "repairing" the individuals after each iteration. It requires much more intermediary computations that complicate the algorithm quite a bit, but the results are better.

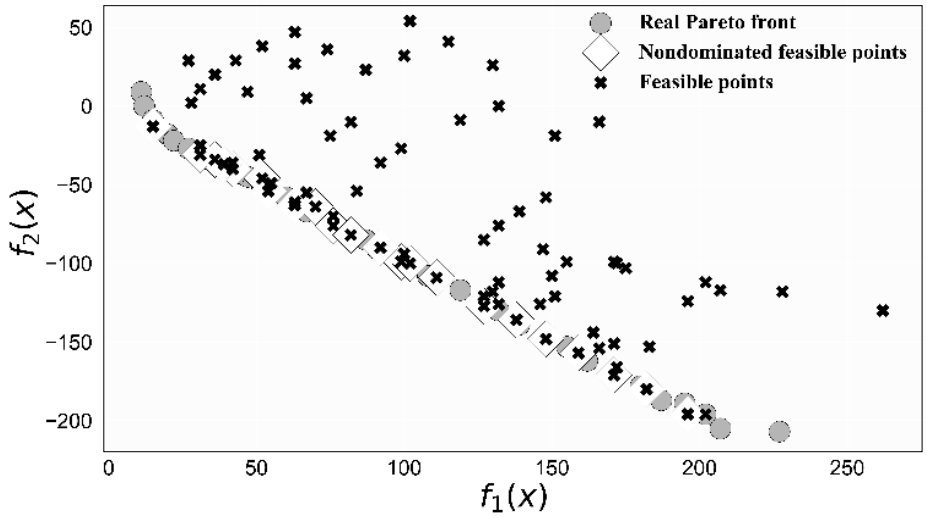

Fig. 14. Resulting individuals for the 8th method

By the means of all gathered data, we are able to come to a conclusion that transforming the constraints into criteria does not give an advantage, and the "repairing" method requires more computational time that does not give high-quality results.

CONCLUSION

The results showed that using the SPEA2 algorithm allows to accurately solve the problem of multicriteria optimization. The methods considered above showed that by modifying SPEA2 it is possible to easily switch from solving the unconditional optimization problem to solving the conditional optimization problem.

The result analysis allows for a following conclusion: the best method for the problem at hand is a simple method of removing the unfit individuals after each iteration as well as the algorithm with repairing after each iteration. Of 100 initial individuals, 43 of the resulting ones were unique and formed a resulting Pareto front of solving the conditional multicriteria optimization problem.

REFERENCES

1. A.P. Braga, R.H. Takahashi, M.A. Costa, and R. Teixeira, "Multi-Objective Algorithms for Neural Networks Learning", *Multi-Objective Machine Learning. Studies in Computational Intelligence*, vol. 16. Berlin, Heidelberg: Springer, 2016. Available: https://doi.org/10.1007/3-540-33019-4_7
2. Carlos Artemio Coello Coello, "An Empirical Study of Evolutionary Techniques for Multiobjective Optimization in Engineering Design", *PhD thesis*. Department of Computer Science, Tulane University, New Orleans, LA, April 1996.
3. C. A. Coello Coello, *A comprehensive survey of evolutionary-based multiobjective optimization techniques*. Laboratorio Nacional de Informatica Avanzada, Veracruz, Mexico, 1998. Available: https://doi.org/10.1007/BF03325101
4. E. Zitzler, M. Laumanns, and L. Thiele, "Spea2: Improving the strength pareto evolutionary algorithm for multiobjective optimization", *Evolutionary Methods for De-*

sign Optimization and Control with Applications to Industrial Problems. International Center for Numerical Methods in Engineering, Athens, Greece, 2001, pp. 95–100.

5. S.V. Groshev, A.P. Karpenko, and V.A. Martynyuk, "The effectiveness of population-based Pareto-approximation algorithms. Experimental comparison", *on-line journal "Naukovedenie",* 8(4), 2016. doi: 10.15862/67EVN416.
6. A.V. Gumennikova, "Hybrid adaptive search algorithm for solving problems of conditional multi-criteria optimization", *Siberian Journal of Science and Technology*, iss. 5, pp. 70–76, 2004.
7. A.V. Gumennikova, "On the evolutionary approach to solving multicriteria problems of conditional optimization", in *VIII international scientific-practical conference "System analysis in the project and management"*, St. Petersburg, 2004, pp. 72–76.
8. A.V. Gumennikova, "Solving multicriteria problems of conditional and unconditional optimization using genetic algorithms MultiobjectiveGA v.1.0", *Computer curriculum and innovation*, no. 8, p. 16, 2005.
9. K. Deb and B.R.N. Uday, "Investigating Predator–Prey algorithms for multiobjective optimization", *KanGAL*, Kanpur, Indian, Rep. 2005010, Dec. 2005.
10. K. Deb, S. Agrawal, A. Pratap, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II", *IEEE Trans. Evol. Comput.*, vol. 6, no. 2, pp. 182–197, Apr. 2002. doi: 10.1109/4235.996017
11. Karl O. Jones, "Comparison of genetic algorithm and particle swarm optimization", in *International Conference on Computer Systems and Technologies*, 2005.
12. A.P. Karpenko, *Modern search engine optimization algorithms. Algorithms inspired by nature*, Moscow: Publishing House MSTU, 2014.
13. A.F. Kuri-Morales and J. Gutiérrez-García, "Penalty Function Methods for Constrained Optimization with Genetic Algorithms: A Statistical Analysis", *Lecture Notes in Computer Science*, vol. 2313. Berlin, Heidelberg: Springer, 2002. Available: https://doi.org/10.1007/3-540-46016-0_12
14. M. Laumanns, G. Rudolph, and H.P. Schwefel, "A spatial predator-prey approach to multi-objective optimization: A preliminary study", in P*roceedings of the Parallel Problem Solving from Nature*, V, pp. 241–249, 1998. Available: https://doi.org/10.1007/BFb0056867
15. David Orvosh and Lawrence Davis, "Using a Genetic Algorithm to Optimize Problems with Feasibility Constraints", *IEEE Conference on Evolutionary Computation – Proceedings*, vol. 2, pp. 548–553, 1994. Available: https://doi.org/10.1109/ICEC.1994.350001
16. J. Schaffer, "Multiple Objective Optimization with Vector Evaluated Genetic Algorithms", *Proceedings of the First Int. Conference on Genetic Algortihms*, pp. 93–100, 1985.
17. E.S. Semenkin, O.E. Semenkina, and S.P. Korobeinikov, *Optimization of technical systems. Tutorial*. Krasnoyarsk: SIBMP, 1996.
18. O.E. Semenkina and V.V. Zhidkov, *Optimization of management of complex systems by the method of generalized local search*. MAKS Press, 2002.
19. Tomio Umeda and Atsunobu Ichikawa, "A Modified Complex Method for Optimization", *Industrial & Engineering Chemistry Process Design and Development,* 10 (2), pp. 229–236, 1971. doi: 10.1021/i260038a016
20. Yu.I. Zhuravlev and Yu.Yu. Finkelstein, "Local Algorithms for Linear Integer Programming Problems", *Cybernetics problems*, iss. 14, pp. 289–295, 1965.

Received 06.10.2020

INFORMATION ON THE ARTICLE

Victor M. Sineglazov, ORCID: 0000-0002-3297-9060, National Aviation University, Ukraine, e-mail: svm@nau.edu.ua.

Kirill D. Riazanovskiy, ORCID: 0000-0002-8771-8060, National Technical University of Ukraine "Igor Sikorsky Kyiv Polytechnic Institute", Ukraine, e-mail: kir_ryaz@tk.kpi.ua.

Olena I. Chumachenko, ORCID: 0000-0003-3006-7460, National Technical University of Ukraine "Igor Sikorsky Kyiv Polytechnic Institute", Ukraine, e-mail: chumachenko@tk.kpi.ua.

БАГАТОКРИТЕРІАЛЬНА УМОВНА ОПТИМІЗАЦІЯ НА ОСНОВІ ГЕНЕТИЧНИХ АЛГОРИТМІВ / В.М. Синєглазов, К.Д. Рязановський, О.І. Чумаченко

Анотація. Розглянуто проблему багатокритеріальної умовної оптимізації, розв'язання якої натепер є найважливішим завданням для багатьох галузей. Запропоновано повторне використання існуючих алгоритмів розв'язання безумовної оптимізації. Розглянуто різні алгоритми багатокритеріальної безумовної оптимізації. Проаналізовано переваги та недоліки кожного алгоритму. Алгоритми модифіковано для врахування обмежень. Розроблено додаткові алгоритми переходу від розв'язання задачі безумовної оптимізації до задачі умовної оптимізації, для тестування яких використано генетичний алгоритм SPEA2. Наведено приклади вирішення поставленого завдання з використанням згаданих алгоритмів. Виконано порівняльний аналіз остаточних результатів.

Ключові слова: багатокритеріальна оптимізація, умовна оптимізація, генетичний алгоритм, алгоритм лікування, SPEA2, Парето оптимізація.

МНОГОКРИТЕРИАЛЬНАЯ УСЛОВНАЯ ОПТИМИЗАЦИЯ НА ОСНОВЕ ГЕНЕТИЧЕСКИХ АЛГОРИТМОВ / В.М. Синеглазов, К.Д. Рязановский, Е.И. Чумаченко

Аннотация. Рассмотрена проблема многокритериальной условной оптимизации, решение которой является одной из важнейших задач настоящего времени для многих областей. Предложено повторное использование существующих алгоритмов решения безусловной оптимизации. Рассмотрены различные алгоритмы многокритериальной безусловной оптимизации. Проанализированы достоинства и недостатки каждого алгоритма. Алгоритмы модифицированы для учета ограничений. Разработаны дополнительные алгоритмы перехода от решения задачи безусловной оптимизации к задаче условной оптимизации, для тестирования которых использован генетический алгоритм SPEA2. Приведены примеры решения поставленной задачи с использованием упомянутых алгоритмов. Проведен сравнительный анализ окончательных результатов.

Ключевые слова: многокритериальная оптимизация, условная оптимизация, генетический алгоритм, алгоритм лечения, SPEA2, Парето оптимизация.