hand, the increase in the savings amortization leads to increase of cumulative expenses Z, that reduces balance profit W_b, cumulative profit tax N_3 and, finally, net profit W_r and reducing efficiency of the project.

On the other hand, the amortization directly influences the parameter NPV aside its increase. Thus, the size change of amortization Am influences the NPV in three directions at the same time. Therefore during the use of the nonlinear method the efficiency of the project will increase, as negative influence of amortization on expenses will be compensated, first, by the positive influence of reduction of the wealth tax, and, secondly, direct influence of amortization on NPV.

Due to the fact that the size increase of amortization Am will lead to increase in general expenses *Z*, it will be reflected in the price of production P_k aside its increase, that, in turn, will influence the demand again. Therefore the problem of how much the parameter NPV will finally change after a change in the amortization charge method will depend on elasticity of demand for an exact production, a competitive position of the enterprise in the market.

An account of considered factors during the preparation of this project and during its realization makes the management of its efficiency possible. Thus it is necessary to consider interaction of factors and change of production demand.

In conclusion we can say that the offered classification of investment projects modeling methods and the lead comparative analysis allows the choosing of toolkit which will correspond to purposes of the user. The investment project efficiency factors, allocated on the basis of the net profit calculation algorithm enable to operate the project efficiency at the stage of planning and during its realization.

Bibliography

1. Zimin, I. A. Real investment / I. A. Zimin. M. : Tandem, 2000. 304 p. (in Russian).

2. Starik, D. E. The estimation of investment projects efficiency / D. E. Starik // Financy. 2006. № 10. C. 70–72. (in Russian).

3. The Designer and solver of discrete optimum control tasks ("Karma") : The Computer program : The certificate on the state registration in Rospatent № 2008614387 from 11.09.2008 / Legal owners: A. V. Medvedev, P. N. Pobedash, A. V. Smoljaninov, M. A. Gorbunov. (in Russian).

A. Yu. Vorozheikin, T. N. Gonchar, I. A. Panfilov, E. A. Sopov, S. A. Sopov
Siberian State Aerospace University named after academician M. F. Reshetnev, Russia, Krasnoyarsk

A MODIFIED PROBABILISTIC GENETIC ALGORITHM FOR THE SOLUTION OF COMPLEX CONSTRAINED OPTIMIZATION PROBLEMS*

A new algorithm for the solution of complex constrained optimization problems based on the probabilistic genetic algorithm with optimal solution prediction is proposed. The efficiency investigation results in comparison with standard genetic algorithm are presented.

Keywords: probabilistic genetic algorithm, constrained optimization.

The necessity to develop complex system models appears in different fields of science and technology such as mathematics, economics, medicine, spacecraft control and so on. In the process of modeling there emerge many optimization problems which are multiextremal, multiobjective and have implicit formalized functions, complex feasible area structure, many types of variables etc. There is no possibility to solve such problems using classical optimization procedures thus we have to design and implement more effective and universal methods such as genetic algorithms (GAs). GAs proved their efficiency in the process of solving of many complex optimization problems [1; 2].

The GAs efficiency depends on fine tuning and control of their parameters. If an untrained user sets arbitrary parameters values, the GA efficiency may vary from very low to very high. The recent trends in a field of GAs are adaptive GAs based on complex hybrid structures and efficient GAs with reduced parameters set.

A known approach to GA parameters set reduction is probabilistic genetic algorithms (pGAs) [3; 4]. The essential difference between pGA and standard GA is that pGA has no crossover operator and new solutions are generated according to statistical information about search space. Thus, collecting and processing such kind of information, pGAs can adapt to the problems they solve.

PGAs demonstrated their efficiency with many complex optimization problems and their further investigation and improvement are promising. There are actual problems of the pGA's features analysis for wide-range optimization problems

* This work is supported by State Programs "Scientific and teaching staff of innovative Russia" (project NK-136P/3), by Grant of the President of Russia to the young PhD for 2009-2010 (MK-2160.2009.9).

and parameters set reduction without efficiency loss. This article is devoted to pGAs efficiency for complex constrained optimization problems investigation.

Probabilistic genetic algorithm for non-constrained optimization problems. During its run the GA collects and processes some statistical information about search space, but the statistics is absent in an explicit form. PGA uses the following form of statistics representation – the probabilities vector of the current population:

$$P^k = (p_1^k, ..., p_n^k),\ p_i^k = P(x_i^k = 1),\ i = \overline{1,n},$$

here p_i^k is the probability of unit value for *i*-th bit in solution X^k, *k* is iteration number.

The general scheme of pGA is:

1. Random generation of the initial population.

2. Selection of *r* individuals (called parents) on the basis of their fitness. Evaluate the probabilities vector as:

$$\overline{P} = (p_1, \ldots, p_n),$$

$$p_j = P\{x_j = 1\} = \frac{1}{r}\sum_{i=1}^{r} x_j^i,\ j = \overline{1,n},$$

here *n* is chromosome length, x_j^i is *j*-th gene of *i* th individual.

3. Form a new population (called offspring) according to the distribution $\overline{P}$.

4. Apply mutation operator to the offspring.

5. Form a new population from parents and offspring.

6. Repeat 2–5 steps.

As it was previously mentioned, during pGA run the algorithm collects the statistics about null and unit values distribution in the population. The experimental results show that the probability vector components converge to the corresponding values of the optimal solution vector as shown in figure 1.

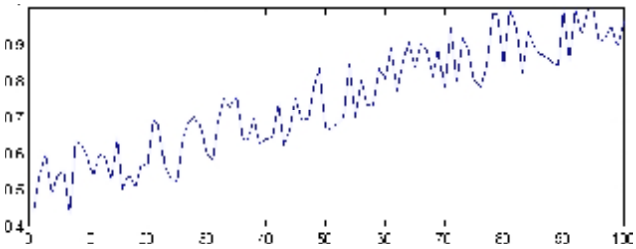

Fig. 1. The values change for *j*-th component of the probability vector

As it is shown in figure 1, the given *j*-th component value of $\overline{P}$ converges to unit. It means that the value of *j*-th gene of optimal solution most probably is equal to unit (for binary representation). One can use this feature to predict the optimal solution.

The following prediction algorithm was proposed in [3; 4]:

1. Choose the certain scheme of pGA for the given problem, set the iteration number $i = 1, ..., I$ and the number of independent algorithm runs $k = 1, ..., K$.

2. Collect the statistic $(p_j)_i^k, j = 1, ..., n$. Average $(p_j)_i$ over *k*. Determine the tendency for p_j change.

3. Set $x_j^{\text{opt}} = 1$, if $\sum_{i=1}^{I}\left((p_j)_i - 0.5\right) > 0$, else $x_j^{\text{opt}} = 0$.

The main idea is that the more often probability value is greater than 0.5, the higher the probability of optimal solution unit value.

In practical problems there may be such situations when pGA collects not enough information at the beginning and *j*-th gene value is equal to unit (or zero) for almost every solution. At the final stage pGA can find a much better solution with inverted values of *j*-th gene and it means that the probability vector values will change their convergence direction (fig. 2). But the above mentioned prediction algorithm will give us the primary value, because the *j*-th value of probability vector was greater than 0.5 (or less than 0.5 for zero values) for a long time.

Thus one can use the following prediction algorithm modification:

1. Set the prediction step *K*.

2. Every *K* iteration use the given statistics $\overline{P}_i$, $i = \overline{1, N_K}$, $N_K = t \cdot K$, $t \in \{1, 2, K\}$ to evaluate the probability vector change: $\Delta\overline{P}_i = \overline{P}_i - \overline{P}_{i-1}$.

3. Set the weights for every iteration according to its number: $\sigma_i = 2i / N_K (N_K + 1)$, $i = 1,\ K, N_K$.

4. Evaluate the probability vector weighted change as:

$$\Delta\tilde{P} = \left(\Delta\tilde{p}_j\right) = \sum_{i=1}^{N_K} \sigma_i \cdot \Delta\overline{P}_i .$$

5. Set the optimal solution: $\overline{X}^{\text{opt}} = \left(x_j^{\text{opt}}\right)$, where $x_j^{\text{opt}} = 1$ if $\Delta\tilde{p}_j \geq 0$, and $x_j^{\text{opt}} = 0$ otherwise.

6. Add the optimal solution in the current population and continue pGA run.

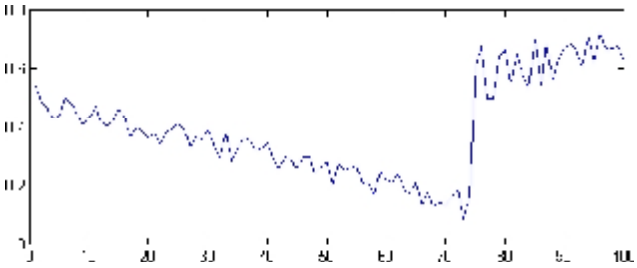

Fig. 2. The situation when prediction can be wrong

The main idea of the given algorithm is that the probability values on the later iterations have the greater weights as the algorithm collects more information about search space. The weights have values such that $\sigma_{i+1} > \sigma_i$ и $\sum_{i=1}^{N_K} \sigma_i = 1$.

Genetic algorithms for constrained optimization problems. In general GAs and pGAs select an individual in accordance with its fitness value, but there is no optimization constrains control. There are many possible methods to solve this problem.

Let the following constrained optimization problem be solved:

$$f(x) \to \text{extr},$$

$$\begin{cases} g_j(x) \leq 0,\ j = \overline{1,r}, \\ h_j(x) = 0,\ j = \overline{r+1,m}. \end{cases}$$

In general, the individual *x* fitness is evaluated as:

$$\text{fitness}(x) = f(x) + \delta \cdot \lambda(t) \cdot \sum_{j=1}^{m} f_j^{\beta}(x),$$

where *t* is iteration number; $\delta = 1$ for the minimization problem; $\delta = -1$ for the maximization problem; $f_j(x)$ is the penalty value for *j*-th constrain break; β is the real number.

The penalty functions $f_j(x)$ are evaluated as:

$$f_j(x) = \begin{cases} \max\{0, g_j(x)\}, j = \overline{1,r} \\ |h_j(x)|, j = \overline{r+1,m} \end{cases}.$$

The following penalty methods are known: the "death" penalty, the static penalty, the dynamic penalty, the adaptive penalty and hybrid methods of the individuals "cure".

As the authors analyzed every penalty method, the further investigation was limited by the dynamic and adaptive penalty methods as other methods have a number of disadvantages.

In particular, the "death" penalty eliminates every unfeasible solution even if it can have the important information for new feasible solutions. The static penalty contains a large set of parameters that should be well tuned – a non-optimal set of parameters can lead to unfeasible solutions. The "cure" method involves local optimization procedures on every iteration of GA, thus such methods use much more computational recourses.

The dynamic penalty. The method uses the previously mentioned penalty functions and defines $\lambda(t)$ in a following way:

$$\lambda(t) = (C \cdot t)^{\alpha}.$$

The fitness of x individual on the t-th iteration is evaluated as:

$$\text{fitness}(x) = f(x) + \delta \cdot (C \cdot t)^{\alpha} \cdot \sum_{i=1}^{m} f_i^{\beta}(x).$$

The values C, α, β are set according to a certain problem. The recommended values are $C = 0.5, \alpha = \beta = 2$ (obtained experimentally).

The adaptive penalty uses the same penalty functions, but $\lambda(t)$ is evaluated as:

$$\lambda(t+1) = \begin{cases} \beta_1 \cdot \lambda(t), \text{ if } \bar{b}^i \in D, \\ \text{for } t-k+1 \le i \le t \\ \beta_2 \cdot \lambda(t), \text{ if } \bar{b}^i \notin D, \\ \text{for } t-k+1 \le i \le t, \\ \lambda(t), \text{ otherwise}, \end{cases}$$

where $\bar{b}^i$ is the best solution in the i-th population, $\beta_1 \in (0,1)$, $\beta_2 > 1$ and $\beta_1\beta_2 \neq 1$. The penalty decreases on the $(t+1)$ step, if the best individual was feasible during the last k iterations. Otherwise, if it was unfeasible, the penalty increases.

The method uses three parameters: β_1, β_2, k. The adaptive penalty method uses both kinds of information: if the solution is unfeasible and if the previous solutions was unfeasible [5].

Probabilistic genetic algorithms for constrained optimization problems. The general scheme of pGA is the same as for the penalty method. The main difference is in the fitness function definition. Thus, one can extend the optimal solution prediction method for the constrained optimization problems. It is appropriate to use the modified prediction procedure as the objective function surface with penalty can have a lot of local optima and the general prediction algorithms can lead to a local solution instead of a global one.

GA and pGA computational efficiency investigation for constrained optimization problems. We compare the algorithms efficiency on a set of test problems of single objective constrained optimization. The objective functions and constrains are linear and non-linear functions of several variables. A part of test problems set is presented in table 1 [6].

We investigate "the best-efficiency" and "the worst-efficiency" parameters set for both algorithms to determine how parameters influence the efficiency in a wide range. The better results for "the worst-efficiency" parameters set give us better effectiveness for arbitrary parameters chosen by an untrained user.

As GA and pGA are stochastic procedures, we average characteristics of algorithms with every unique parameter set over 100 independent runs.

To estimate algorithms efficiency we will use the following criteria:

– the rate of runs (%), where the exact optimal solution was computed;

– the average iteration number (*N*), on which the exact optimal solution was computed for the first time.

At the first stage we define the constrains control method that gives the best efficiency with the given test problems set. We have resumed that the standard GA with "the best-efficiency" parameters shows the best results with the dynamic penalty for the whole test problems set. The standard GA with "he worst-efficiency" parameters shows the best results with the dynamic penalty only for 60 % cases (problems). On the average, the dynamic penalty is more effective than the adaptive penalty in 60 % cases.

PGA both with "the worst-efficiency" and "the best-efficiency" parameters shows the best results with the dynamic penalty for the whole test problems set.

Thus, we have determined that the dynamic penalty is more effective than the adaptive penalty for both GA and pGA algorithms.

At the second stage we compare the efficiency of standard GA and pGA with dynamic penalty. For "the best-efficiency" parameters set the standard GA is more effective than pGA in 67 % cases. But in cases when pGA yelds to GA, their efficiency differs insignificantly. For "the worst-efficiency" and "average-efficiency" pGA is more effective than GA in 100 % and 67 % cases respectively. The computational results of comparison are given in table 2.

At the third stage we compare the efficiency of standard GA and pGA with dynamic penalty. For "the best-efficiency" parameters set the standard GA is more effective than pGA in 67 % cases. But in cases pGA yelds to GA, their efficiency differs insignificantly. For "the worst-efficiency" and "average-efficiency" pGA is more effective than GA in 100 % and 67 % cases respectively. The computational results of comparison are given in table 2.

At the forth stage we compare the efficiency of standard GA and pGA with optimal solution prediction. For «the best-efficiency» parameters set the pGA with optimal solution prediction is more effective than the standard GA in 60 % cases, "the worst-efficiency" pGA is more effective in 67 % cases. Moreover, the pGA with optimal solution prediction finds the optimal solution at much earlier iteration. The comparison of computational results are in table 3.

The results of the investigation shows that the pGA algorithm is more preferable than the standard GA, because it is more effective on the average and it has smaller number of parameters. The pGA with optimal solution prediction allows to compute optimal solution at much earlier iterations.

Bibliography

1. Holland, J. H. Adaptation in natural and artificial systems / J. H. Holland. Ann Arbor, MI : University of Michigan Press, 1975.

2. Goldberg. D. E. Genetic algorithms in search, optimization, and machine learning / D. E. Goldberg. Reading, MA : Addison-Wesley, 1989.

3. Сопов Е. А. Вероятностный генетический алгоритм и его исследование / Е. А. Сопов // VII Королевские чтения. Т. 5. Самара : Изд-во Самар. науч. центра Рос. Акад. наук, 2003. С. 38–39.

4. Сопов Е. А. О вероятностном генетическом алгоритме. Современные техника и технологии. В 2 т. Т. 2 / Е. А. Сопов // Томск : Изд-во Том. политехн. ун-та, 2004. С. 197–199.

5. Michalewicz, Z. Genetic algorithms, numerical optimization and constraints / Z. Michalewicz // Proc. of the Sixth Intern. Conf. on Genetic Algorithms and their Applications. Pittsburgh, PA, 1995.

6. Whitley, D. Building Better Test Functions / D. Whitley // Proc. of the Sixth Intern. Conf. on Genetic Algorithms and their Applications. Pittsburgh, PA, 1995.

Table 1

The test problems for constrained optimization

The problem statement	The exact solution
$z = x^2 + y^2 \to \max$ $\begin{cases} y \le 7 + \sin(2 \cdot x) \\ y \ge 1 - \sin(2 \cdot x) \\ x \in [0, 4] \end{cases}$	$x = 4$ $y = 7.989358247$ $z^* = 79.82984520$
$z = 5 \cdot x + 0.5 \cdot y \to \max$ $\begin{cases} y \le -2 \cdot x + 5 \\ y \ge x - 1.5 \\ y \le 2 \cdot x + 1 \\ x \ge 0 \\ y \ge 0 \end{cases}$	$x = \frac{13}{6} = 2.16666$ $y = \frac{2}{3} = 0.66666$ $z^* = \frac{67}{6} = 11.16666667$
$z(x, y) = 2000x + 2400y \to \max$ $\begin{cases} x \ge 0 \\ y \ge 0 \\ \frac{x}{120} + \frac{y}{110} \le 1 \\ 4x + y \le 320 \\ x + y \le 110 \\ \frac{x}{340} + \frac{y}{120} \le 1 \\ x + 2y \le 160 \\ x + 4y \le 280 \end{cases}$	$x = 50$ $y = 55$ $z_{\text{opt}} = 232,000$
$z = 20 + e - 20\exp\left(-0.2\sqrt{\sum_{i=1}^{N} \frac{x_i^2}{N}}\right) - \exp\left(\sum_{i=1}^{N} \frac{\cos(2\pi \cdot x_i)}{N}\right) \to \min$ $N = 4$ $\begin{cases} 2x_1 - 3x_2 + 4x_3 \le 10 \\ 4x_2 - 5x_3 + x_4 \le 1 \\ 10x_1 + 7.5x_3 - 8.4x_4 \le 3.5 \\ -3.1x_1 + 21.7x_2 - 36.4x_4 \le 16.2 \end{cases}$	$x_i = 0, i = \overline{1, N}$ $z_{\text{opt}} = 0$
$z = \sum_{i=1}^{N} \left(0.1 \cdot x_i^2 - 4\cos(0.8x_i) + 4\right) \to \min$ $N = 2$ $\begin{cases} x_1^2 + 9x_2^2 \le 36 \\ 9x_1^2 + x_2^2 \le 36 \end{cases}$	$x_i = 0, i = \overline{1, N}$ $z_{\text{opt}} = 0$

Table 2

The GA and pGA with the dynamic penalty efficiency comparison for constrained optimization problem

The problem	The best-efficiency parameters				The best-efficiency parameters				The average-efficiency parameters			
	GA		pGA		GA		pGA		GA		pGA	
	%	N	%	N	%	N	%	N	%	N	%	N
Linear problem 1	76	31.05	64	32.81	12	16.83	14	14.14	41.78	27.67	40.22	26.2
Linear problem 2	100	472.04	100	446.9	0	0	0	0	61.19	357.38	61.56	375.58
Non-linear problem 1	58	32.28	66	32.39	8	25	24	18.25	34.22	28.95	41.33	26.49
Non-linear problem 2	100	21.22	98	15.02	44	9.93	68	9.12	76.07	14.49	83.56	12.24
Non-linear problem 3	20	18.8	68	29.94	0	0	22	13.64	7.41	19.45	40.89	24.68
Non-linear problem 4	94	35.77	92	36.09	52	43.42	48	42.96	72.81	33.07	72.44	31.86
Rastrigin function	100	54.4	100	33.4	5	10	100	76.92	56.85	58.32	100	48.22
Ackley (and)	100	2.76	100	4.51	95	61.21	100	69.71	99.63	23.02	100	32.92
Ackley (or)	100	1.94	100	3.79	100	63.04	100	42.24	100	12.57	100	13.41
The number of wins	7	6	6	3	2	2	7	6	3	4	7	5
The rate of wins	53.85	66.67					77.78	75			70	55.56
The number of double wins	4		2		0		5		1		3	
The rate of double wins	66.67						100				75	

Table 3

The GA and pGA with optimal solution prediction efficiency comparison for constrained optimization problem

The problem	The best-efficiency parameters				The best-efficiency parameters				The average-efficiency parameters			
	GA		pGA (prediction)		GA		pGA (prediction)		GA		pGA (prediction)	
	%	N	%	N	%	N	%	N	%	N	%	N
Linear problem 1	76	31.05	48	34.29	12	16.83	22	24.64	41.78	27.67	39.33	32.8
Linear problem 2	100	472.04	100	438.4	0	0	2	8	61.19	357.38	60	350.82
Non-linear problem 1	58	32.28	46	23.96	8	25	0	0	34.22	28.95	21.78	19.49
Non-linear problem 2	100	21.22	66	18.12	44	9.93	28	14.21	76.07	14.49	45.11	16.91
Non-linear problem 3	20	18.8	34	29.47	0	0	0	0	7.41	19.45	17.33	21.45
Non-linear problem 4	94	35.77	100	38.92	52	43.42	60	41.77	72.81	33.07	84.22	30.84
Rastrigin function	100	54.4	100	32.36	5	10	98	55.22	56.85	58.32	99.78	47.82
Ackley (and)	100	2.76	100	3.06	95	61.21	100	50.57	99.63	23.02	100	29.74
Ackley (or)	100	1.94	100	3.47	100	63.04	100	56.65	100	12.57	100	15.58
The number of wins	7	4	5	6	3	4	6	4	5	5	5	4
The rate of wins	58.33			60		50	66.67	50	50	55.56	50	
The number of double wins	2		3		2		4		1		2	
The rate of double wins			60				66.67				66.67	