A real coded genetic algorithm for solving integer and mixed integer optimization problems

Article *in* Applied Mathematics and Computation · June 2009

DOI: 10.1016/j.amc.2009.02.044 · Source: DBLP

4 authors:

Kusum Deep
Indian Institute of Technology Roorkee
115 PUBLICATIONS **1,289** CITATIONS
SEE PROFILE

Krishna Pratap Singh
Indian Institute of Information Technology Al…
13 PUBLICATIONS **239** CITATIONS
SEE PROFILE

M. L. Kansal
Indian Institute of Technology Roorkee
58 PUBLICATIONS **446** CITATIONS
SEE PROFILE

Chander Mohan
Ambala College of Engineering and Applied R…
97 PUBLICATIONS **735** CITATIONS
SEE PROFILE

Applied Mathematics and Computation 212 (2009) 505–518

Contents lists available at ScienceDirect

Applied Mathematics and Computation

journal homepage: www.elsevier.com/locate/amc

A real coded genetic algorithm for solving integer and mixed integer optimization problems

Kusum Deep [a,*], Krishna Pratap Singh [a], M.L. Kansal [b], C. Mohan [c]

[a] Department of Mathematics, Indian Institute of Technology, Roorkee-247667, Uttarakhand, India
[b] Department of Water Resources Development and Management, Indian Institute of Technology, Roorkee-247667, Uttarakhand, India
[c] Ambala College of Engineering and Applied Research, Ambala, Haryana, India

ARTICLE INFO

Keywords:
Real coded genetic algorithms
Random search based techniques
Constrained optimization
Integer and mixed integer optimization problems

ABSTRACT

In this paper, a real coded genetic algorithm named MI-LXPM is proposed for solving integer and mixed integer constrained optimization problems. The proposed algorithm is a suitably modified and extended version of the real coded genetic algorithm, LXPM, of Deep and Thakur [K. Deep, M. Thakur, A new crossover operator for real coded genetic algorithms, Applied Mathematics and Computation 188 (2007) 895–912; K. Deep, M. Thakur, A new mutation operator for real coded genetic algorithms, Applied Mathematics and Computation 193 (2007) 211–230]. The algorithm incorporates a special truncation procedure to handle integer restrictions on decision variables along with a parameter free penalty approach for handling constraints. Performance of the algorithm is tested on a set of twenty test problems selected from different sources in literature, and compared with the performance of an earlier application of genetic algorithm and also with random search based algorithm, RST2ANU, incorporating annealing concept. The proposed MI-LXPM outperforms both the algorithms in most of the cases which are considered.

1. Introduction

A mixed integer programming problem is an optimization problem, linear or nonlinear, with or without constraints, in which some or all decision variables are restricted to have integer values. Such problems frequently arise in various application fields such as process industry, finance, engineering design, management science, process flow sheets, portfolio selection, batch processing in chemical engineering, and optimal design of gas and water distribution networks. Other areas of application in which such problems also arise are automobile engineering, aircraft design, and VLSI manufacturing.

The general mathematical model of a mixed integer programming problem (MIPP) is:

$$\min \quad f(x,y),$$

subject to:

$$g_j(x,y) \leqslant b_j, \quad j = 1,\ldots,r_1,$$
$$h_j(x,y) = b_j, \quad j = r_1+1,\ldots,r_1+r_2,$$
$$x_i^L \leqslant x_i \leqslant x_i^U, \quad i = 1,\ldots,n_1,$$

* Corresponding author.
E-mail addresses: kusumfma@iitr.ernet.in (K. Deep), kpsinghiitr@gmail.com (K.P. Singh), mlkgkfwt@iitr.ernet.in (M.L. Kansal), chander_mohan2@rediffmail.com (C. Mohan).

0096-3003/$ - see front matter © 2009 Elsevier Inc. All rights reserved.
doi:10.1016/j.amc.2009.02.044

$$y_i^L \leqslant y_i \leqslant y_i^U : integer, \quad i = 1, \ldots, n_2,$$
$$x = [x_1, x_2, \ldots, x_{n1}]^T,$$
$$y = [y_1, y_2, \ldots, y_{n2}]^T.$$

Several classical computational techniques (such as branch and bound technique, cutting planes technique, outer approximation technique etc.), which are reasonably efficient, have been proposed in literature for solving mixed integer programming problems ([3–7]). These techniques are applicable to a particular class of problems. In the case of non-convex problems these techniques may cut-off the global optima.

In the last two decades many stochastic algorithms are developed and suitably updated for mixed integer programming problems. Simulated annealing technique, first proposed by Kirkpatrick et al. [8], has proved a valuable tool in solving real and combinatorial global optimization problems ([9,10]). However, algorithms of this class generally posses the ability to provide near global optimal solutions, but the quality of the obtained solution is not stable and the computational time required is generally large. Other techniques such as Differential Evolution ([11]), Line-up competition algorithm ([12]) and Particle Swarm Optimization ([13]) are also used for integer and mixed integer programming problems.

Controlled random search techniques CRS1 and CRS2 ([14,15]) are stochastic algorithms for global optimization problems in which decision variables may have both integer as well as real values. Mohan and Shanker [16] developed an improved version of CRS2 algorithm which uses quadratic approximation in place of simplex approach adopted in CRS2 version and named it RST2 algorithm. Later on, [17] developed a controlled random search technique, called the RST2ANU algorithm. This algorithm incorporates the simulated annealing concept in RST2 algorithm. RST2ANU algorithm is claimed to be more reliable and efficient than RST2 algorithm, and shown to be effective in solving integer and mixed integer constrained optimization problems as well. Salcedo [18] has used an adaptive controlled random search for such problems.

Genetic algorithms (GAs) are general purpose population based stochastic search techniques which mimic the principles of natural selection and genetics laid down by Charles Darwin. The concept of GA was introduced by Holland [19]. This approach was first used to solve optimization problem by De-Jong [20]. A detailed implementation of GA may be found in [21].

In a GA, a population of potential solutions, termed as chromosomes (individuals), is evolved over successive generations using a set of genetic operators called selection, crossover and mutation operators. First of all, based on some criteria, every chromosome is assigned a fitness value, and then a selection operator is applied to choose relatively 'fit' chromosome to be part of the reproduction process. In reproduction process new individuals are created using crossover and mutation operators. Crossover operator blends the genetic information between chromosomes to explore the search space, whereas mutation operator is used to maintain adequate diversity in the population of chromosomes to avoid premature convergence.

The way the variables are coded is clearly essential for GAs' efficiency. Real coded genetic algorithms (RCGAs), which use real numbers for encoding, have fast convergence towards optima than binary and gray coded GAs ([22]). Also, RCGAs overcomes the difficulty of "Hamming Cliff" as in binary coded GAs. In the case of integer and mixed integer programming problems many applications of GAs are available in literature, some of these use binary coded representation ([23–26]) and some use real coded representation ([27–30]). Most of the above approaches use round off of real variable to deal with integer restriction of decision variables. Also, they may differ from each other in the terms coding (binary or real), crossover operator, mutation operator, selection technique and constraint handling approach used in their algorithm. Till date there is no single combination of crossover operator, mutation operator, selection technique and constraint handling approach which is a completely robust GA for solving integer and mixed integer nonlinear programming problems.

The above works motivate us to develop an efficient algorithm for integer and mixed integer nonlinear programming problems. Hence, we have suitably modified and extended the recently developed real coded genetic algorithm, LXPM by Deep and Thakur [1,2], to handle integer restrictions on some or all decision variables. Also, a truncation procedure is incorporated for those variables which have integer restriction. Moreover, a parameter free constraint handling technique is incorporated into LXPM algorithm for handling of constraints. This new version is called MI-LXPM algorithm. The proposed algorithm creates more randomness for efficient handling of integer restrictions on decision variables and increases the possibility to obtain the global optimal solution.

The paper is organized as follows: The proposed MI-LXPM algorithm is described in Section 2. Laplace crossover, Power mutation, tournament selection technique, truncation procedure for integer restrictions and constraint handling techniques are discussed in some details in sub Sections 2.1–2.4 and 2.5, respectively. The algorithm is finally outlined in sub Section 2.6. It is applied to a set of 20 test problems in Section 3 and its performance is compared with that of AXNUM and RST2ANU algorithms. Discussion on the numerical results follows in Section 4. Conclusions, based on the present study, are finally drawn in Section 5.

2. MI-LXPM algorithm

MI-LXPM algorithm is an extension of LXPM algorithm, which is efficient to solve integer and mixed integer constrained optimization problems. In MI-LXPM, Laplace crossover and Power mutation are modified and extended for integer decision variables. Moreover, a special truncation procedure for satisfaction of integer restriction on decision variables and a 'parameter free' penalty approach for constraint handling are used in MI-LXPM algorithm. More details of these operators are defined in subsequent subsections.

2.1. Laplace crossover

Laplace crossover is defined, in original form, in [1]. Herein, we have added another parameter in the Laplace crossover operator to take care of integer decision variables in the optimization problem. Working of the extended Laplace crossover is described below. Two offsprings, $y^1 = (y_1^1, y_2^1, \ldots, y_n^1)$ and $y^2 = (y_1^2, y_2^2, \ldots, y_n^2)$ are generated from two parents, $x^1 = (x_1^1, x_2^1, \ldots, x_n^1)$ and $x^2 = (x_1^2, x_2^2, \ldots, x_n^2)$ in the following way. First, uniform random numbers $u_i, r_i \in [0, 1]$ are generated. Then a random number β_i, which satisfy the Laplace distribution, is generated as:

$$\beta_i = \begin{cases} a - b\log(u_i), & r_i \leqslant 1/2; \\ a + b\log(u_i), & r_i > 1/2, \end{cases}$$

where a is location parameter and $b > 0$ is scaling parameter. If the decision variables have a restriction to be integer then $b = b_{int}$, otherwise $b = b_{real}$, i.e., for integer and real decision variables, scaling parameter (b) is different. With smaller values of b, offsprings are likely to be produced nearer to parents and for larger values of b, offsprings are expected to be produced far from parents. Having computed β_i, the two offsprings are obtained as under:

$$y_i^1 = x_i^1 + \beta_i |x_i^1 - x_i^2|,$$
$$y_i^2 = x_i^2 + \beta_i |x_i^1 - x_i^2|.$$

2.2. Power mutation

Power mutation is defined, in detail, in [2]. It is based on power distribution. We have added another parameter in the Power mutation for integer restriction of decision variables. Working of the extended Power mutation is as follows: A solution x is created in the vicinity of a parent solution $\bar{x}$ in the following manner. First, a random number s which follows the power distribution, $s = (s_1)^p$, where s_1 is a uniform random number between 0 and 1, are created. p is called the index of mutation. It governs the strength of perturbation of power mutation. $p = p_{real}$ or $p = p_{int}$ depending on integer or real restriction on the decision variable. In other words for integer decision variables, value of p is p_{int} and for real decision variables p is p_{real}. Having determined s a muted solution is created as:

$$x = \begin{cases} \bar{x} - s(\bar{x} - x^l), & t < r; \\ \bar{x} + s(x^u - \bar{x}), & t \geqslant r. \end{cases}$$

where $t = \frac{\bar{x} - x^l}{x^u - x^l}$, x^l and x^u being the lower and upper bounds on the value of the decision variable and r a uniformly distributed random number between 0 and 1.

2.3. Selection technique

Genetic algorithms use a selection technique to select individuals from the population to insert individual into mating pool. Individuals from the mating pool are used to generate new offspring, with the resulting offspring forming the basis of the next generation. A selection technique in a GAs is simply a process that favors the selection of better individuals in the population for the mating pool.

Goldberg and Deb [31] have shown that the tournament selection has better or equivalent convergence and computational time complexity properties when compared to any other reproduction operator that exists in the literature. So, in this algorithm, tournament selection operator is used as reproduction operator. In the tournament selection, tournaments are played between k solutions (k is tournament size) and the better solution is chosen and placed in the mating pool. k other solutions are picked again and another slot in the mating pool is filled with the better solution. If carried out systematically, each solution can be made to participate in exactly k tournaments. The best solution in a population will win all the k tournaments, there by making k copies of it in the new population. Using a similar argument, the worst solution will lose in all the k tournaments and will be eliminated from the population. The user specifies the size of the tournament set as a percentage of the total population. In this study, tournament selection operator with tournament size three is used.

2.4. Truncation procedure for integer restrictions

In order to ensure that, after crossover and mutation operations have been performed, the integer restrictions are satisfied, the following truncation procedure is applied. Namely, for $\forall i \in I$,x_i is truncated to integer value $\bar{x}_i$ by the rule:

- if x_i is integer then $\bar{x}_i = x_i$, otherwise,
- $\bar{x}_i$ is equal to either $[x_i]$ or $[x_i] + 1$ each with probability 0.5, ($[x_i]$ is the integer part of x_i).

This ensures greater randomness in the set of solutions being generated and avoids the possibility of the same integer values being generated, whenever a real value lying between the same two consecutive integers is truncated.

2.5. Constraint handling approach

Constraint handling in optimization problems is a real challenge. Parameter free, penalty function approach based on feasibility approach proposed by Deb [32] is used in this study. Fitness value, fitness(X_i) of an ith individual is evaluated as:

$$\text{fitness}(X_i) = \begin{cases} f(X_i), & \text{if } X_i \text{ feasible;} \\ f_{worst} + \sum_{j=1}^{m} |\phi_j(X_i)|, & \text{otherwise;} \end{cases}$$

where, f_{worst} is the objective function value of the worst feasible solution currently available in the population. Thus, the fitness of an infeasible solution not only depends on the amount of constraint violation, but also on the population of solutions at hand. However, the fitness of a feasible solution is always fixed and is equal to its objective function value. $\phi_j(X_i)$ refers to value of the left hand side of the inequality constraints (equality constraint are also transformed to inequality constraints using a tolerance). If there are no feasible solutions in the population, then f_{worst} is set zero. It is important to note that such a constraint handling scheme without the need of a penalty parameter is possible because GAs use a population of solutions in every iteration and comparison of solutions is possible using the tournament selection operator. For the same reason, such schemes cannot be used with classical point-by-point search and optimization methods. Two individual solutions now compared using the following rules:

(1) A feasible solution is always preferred over an infeasible one.
(2) Between two feasible solutions, the one having better objective function is preferred.
(3) Between two infeasible solutions, the one having smaller constraint violation is preferred.

The use of constraint violation in the comparisons aim to push infeasible solutions towards the feasible region (In a real life optimization problem, the constraints are often non-commensurable, i.e., they are expressed in different units. Therefore, constraints are normalized to avoid any sort of bias).

2.6. Computational steps of MI-LXPM

Computational steps of the proposed MI-LXPM algorithm are:

(1) Generate a suitably large initial set of random points within the domain prescribed by the bounds on variable i.e., points satisfying $x_i^L \leqslant x_i \leqslant x_i^U, i = 1, 2, \ldots n$, for variables which are to have real values and $y_i^L \leqslant y_i \leqslant y_i^U$, y_i integer for variables which are to have integer values.
(2) Check the stopping criteria. If satisfied stop; else goto 3.
(3) Apply tournament selection procedure on initial (old) population to make mating pool.
(4) Apply laplace crossover and power mutation to all individuals in mating pool, with probability of crossover (P_c) and probability of mutation (P_m), respectively, to make new population.
(5) Apply integer restrictions on decision variables where necessary and evaluate their fitness values.
(6) Increase generation++; old population $\leftarrow$ new population; goto 2.

3. Solution of test problems

MI-LXPM algorithm, developed in the previous section, is used to solve a set of 20 test problems taken from different sources in the literature. These are listed in Appendix. These include integer and mixed integer constrained optimization problems. All (except 16) are nonlinear. The number of unknown decision variables in these problems varies from 2 to 100. The results are presented in Table 1.

Performance of MI-LXPM algorithm is compared with the earlier RCGA (we call it AXNUM algorithm), which has different crossover and mutation operators (Arithmetic crossover and Non-uniform mutation, [27]). AXNUM algorithm also uses tournament selection operator. It uses x_i always as $[x_i]$ or $[x_i] + 1$ for satisfaction of integer restrictions on decisions variables. Solutions of these test problems with AXNUM are given in Table 1. It is also compared with RST2ANU algorithm of [17]. Solutions of the problems with RST2ANU algorithm are also given in Table 1.

Each problem is executed 100 runs with all the three algorithms (MI-LXPM, AXNUM and RST2ANU algorithms). Each run is initiated using a different set of initial population. A run is considered a success if achieved value of the objective function is within 1% of the known optimal value (in case the optimal value of the objective is zero, a run is considered success if the achieved absolute value of the objective function is less than 0.01). For each problem, the percentage of success (obtained as the ratio of the number of successful runs to total number of runs), the average number of function evaluations in the case of successful runs and the average time in seconds used by the algorithm in achieving the optimal solution in the case of the successful runs are also listed.

Table 1
Results obtained by using MI-LXPM, RST2ANU and AXNUM algorithms.

	MI-LXPM			RST2ANU			AXNUM		
Problem	ps	ave	t	ps	ave	t	ps	ave	t
1	84	172	0.03489	47	173	0.00229	86	1728	0.04250
2	85	64	0.05940	57	657	0.05211	67	82	0.09825
3	43	18608	0.38344	04	221129	0.19340	35	65303	0.38677
4	95	10933	0.64642	02	1489713	172.31	82	45228	0.22643
5	100	671	0.00234	75	2673	0.0076	95	13820	0.06245
6	100	84	0.00015	100	108	0.00512	100	432	0.00188
7	59	7447	0.64459	00	-	-	45	16077	0.64304
8	41	3571	0.82012	15	180859	4.34473	03	1950	1.39033
9	100	100	0.00032	100	189	0.01030	100	4946	0.01691
10	93	258	0.04908	100	545	0.01924	33	700	2.04736
11	100	171	0.00630	100	2500	0.01095	97	863	0.03319
12	71	299979	3.27762	29	6445	0.02431	19	380115	3.88332
13	99	77	0.00598	100	35	0.00343	91	456	0.05253
14	100	78	0.00061	100	214	0.00861	100	1444	0.01749
15	92	2437	0.39190	19	3337	0.02821	09	267177	1.96167
16a	100	1075	0.02609	100	1114	1.39881	100	2950	0.4252
16b	100	1073	0.02578	100	1189	1.49686	100	3016	0.03889
17a	100	600	0.05452	100	2804	18.4187	100	600	0.02194
17b	100	600	0.03535	100	1011	1.3728	100	600	0.0194
18	100	250	0.00139	100	697	0.01850	100	256	0.00218

ps = Percentage of the successful runs to total runs, ave = average number of function evaluations of successful runs, t = average time in seconds used by the algorithm in achieving the optimal solution in case of successful runs.

4. Discussion on the results

In the MI-LXPM algorithm, like other genetic algorithms, finding appropriate value of parameters is the most important and difficult task. Difficulty in parameter fine tuning increases in the case of RCGAs, since the number of parameters involved in RCGAs are more than in binary GAs. For a given test suit, an extensive computational exercise has to be carried out to determine the most optimum parameters setting for MI-LXPM. The most efficient parameter setting found by our experiments were as follows:

Crossover probability $(p_c) = 0.8$; mutation probability $(p_m) = 0.005$; $a = 0$; $b_{real} = 0.15$; $b_{int} = 0.35$; $p_{real} = 10$ and $p_{int} = 4$.

In AXNUM algorithm value of parameters are $p_c = 0.7$ and $p_m = 0.001$. In RST2ANU parameter setting is taken same as reported in [17]. Also, the population size is taken ten times the number of decision variables, except in problem 16a, 16b,17a and 17b where population size is taken three times of number of variables.

Results in Table 1 show that, in case of 10 problems MI-LXPM algorithm provides 100% success. Moreover, only in 3 problems its success rate is less than 50%. In case of AXNUM algorithm, 100% success rate is achieved in 8 problems, but in 6 problems success rate is less than 50%. However, in case of RST2ANU algorithm, 100% success rate is achieved in 11 problems but in 7 problems success rate is less than 50%. Also in the case of problem-4, all the 100 trials failed to achieve optimal solution. MI-LXPM algorithm also required less number of average function evaluations than AXNUM and RST2ANU algorithm in 16 problems. In two problems (problem-17a and problem-17b), AXNUM and MI-LXPM algorithm, both used equal average function evaluations. However, in two problems (problem-12 and problem 13), RST2ANU algorithm used less function evaluations than MI-LXPM and AXNUM algorithm. In the case of 10 problems MI-LXPM algorithm used less computational time than AXNUM and RST2ANU algorithm, while in 7 problems RST2ANU algorithm required less time than MI-LXPM and AXNUM algorithm. Only in one case AXNUM algorithm used less computational time to other algorithms.

In order to get a better insight into the relative performance of MI-LXPM, AXNUM and RST2ANU algorithms, the value of a performance index (PI), proposed by Bharti [33], is calculated in respect of these three algorithms. Mohan and Nguyen [17] have also used this performance index for comparison of the relative performance of algorithms developed by them. This index gives prescribed weighted importance to the rate of success, the computational time and the number of function evaluations. For the computational algorithms under comparison the value of performance index PI_j for the jth algorithm is computed as:

$$PI_j = \frac{1}{N}\sum_{i=1}^{N}(k_1\alpha_1^i + k_2\alpha_2^i + k_3\alpha_3^i)$$

Here $\alpha_1^i = \frac{Sr^i}{Tr^i}$,

$$\alpha_2^i = \begin{cases} \frac{Mt^i}{At^i}, & \text{if } Sr^i \geqslant 0, \\ 0, & \text{if } Sr^i = 0, \end{cases} \quad \text{and } \alpha_3^i = \begin{cases} \frac{Mf^i}{Af^i}, & \text{if } Sr^i \geqslant 0, \\ 0, & \text{if } Sr^i = 0. \end{cases}$$

where, $i = 1, 2, \ldots, N$.

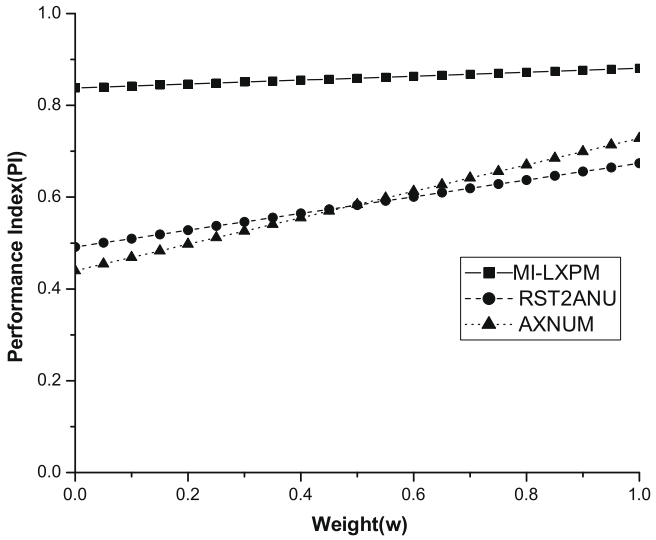

Fig. 1. Performance index of MI-LXPM, RST2ANU and AXNUM when $k_1 = w; k_2 = k_3 = (1 - w)/2$.

Also, Tr^i denotes the total number of times the ith problem is solved,and Sr^i the number of times ith problem is solved successfully. At^i is the average of the time required by the jth algorithm in obtaining the optimal solution of ith problem in case of successful runs, and Mt^i is minimum of the average time required by various algorithms under comparison in obtaining the optimal solution of ith problem. Similarly, Af^i is the average number of function evaluations used by the jth algorithm in obtaining the optimal solution of ith problem in the case of the successful runs, and Mf^i the minimum of the average number of function evaluations of successful runs used of the algorithms under comparison in obtaining the optimal solution of ith problem. Also N is the total number of problems on which the performance of algorithms has been tested.

Further, k_1, k_2 and k_3 are nonnegative constraints such that $k_1 + k_2 + k_3 = 1$ (these are in fact the weights assigned by the user to the percentage of success, the average execution time of successful runs and the average number of function evaluations used in successful run, respectively). Larger the value of PI_j, better is the performance of the algorithm. In order to analyze the relative performance of MI-LXPM, AXNUM and RST2ANU algorithms, we assigned equal weights to two of these terms at a time so that, PI_j became a function of a single variable. The cases considered were:

(1) $k_1 = w, k_2 = k_3 = (1 - w)/2, 0 \leqslant w \leqslant 1$,
(2) $k_2 = w, k_1 = k_3 = (1 - w)/2, 0 \leqslant w \leqslant 1$,

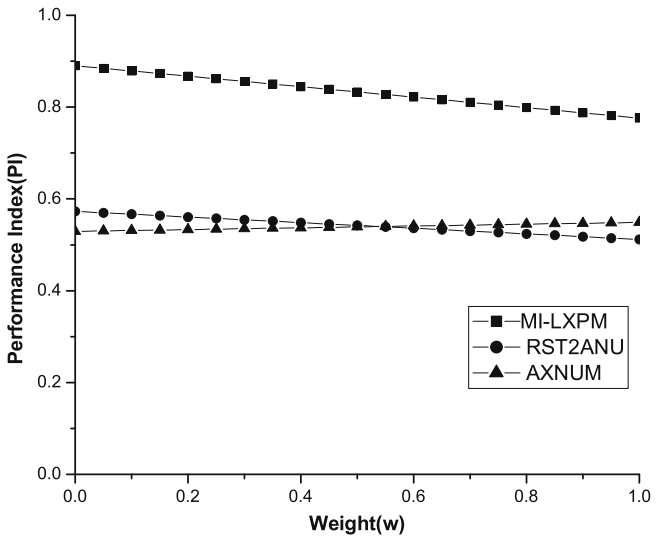

Fig. 2. Performance index of MI-LXPM, RST2ANU and AXNUM when $k_2 = w; k_1 = k_3 = (1 - w)/2$.

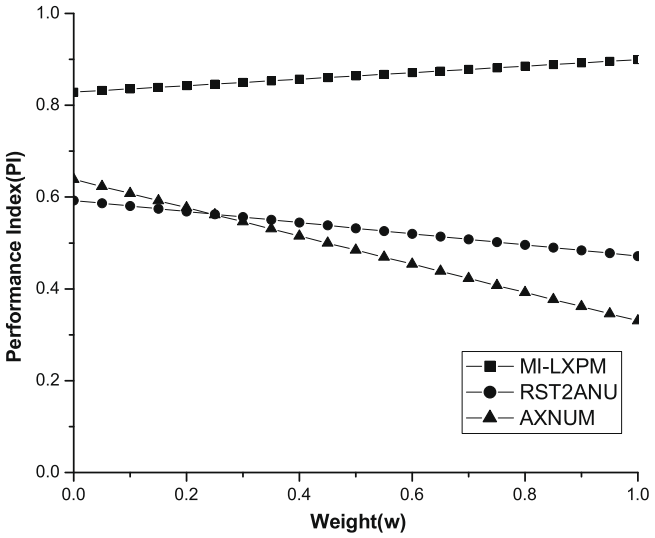

Fig. 3. Performance index of MI-LXPM, RST2ANU and AXNUM when $k_3 = w; k_1 = k_2 = (1 - w)/2$.

(3) $k_3 = w, k_1 = k_2 = (1 - w)/2, 0 \leqslant w \leqslant 1$.

The graphs of PI_j, corresponding to each of these three cases, are shown in Fig. 1–3, respectively. In Fig. 1, weight assigned to the percentage of success is $k_1 = w$, and for average time of successful run (k_2) and average function evaluation of successful run (k_3) are $k_2 = k_3 = (1 - w)/2$. Performance Index values of all the three algorithms, at each value of w between 0 and 1, show that MI-LXPM algorithm is better than RST2ANU and AXNUM algorithms. In Fig. 2, weights are assigned as $k_2 = w, k_1 = k_3 = (1 - w)/2$. PI values of all the three algorithm with respect to weight show that MI-LXPM algorithm is better than other algorithms. Similarly, in Fig. 3, for weights $k_3 = w, k_1 = k_2 = (1 - w)/2$ graph shows that PI value of MI-LXPM algorithm is better than RST2ANU and AXNUM algorithms. On the basis of these three graphs, it is observed that MI-LXPM outperforms AXNUM and RST2ANU algorithms.

5. Conclusions

In this paper, a real coded genetic algorithm MI-LXPM is proposed for solution of constrained, integer and mixed integer optimization problems. In this algorithm a special truncation procedure is incorporated to handle integer restriction on the decision variables and "parameter free" penalty approach is used for the constraints of the optimization problems.

The performance of the proposed MI-LXPM algorithm is compared with AXNUM and RST2ANU algorithm on a set of 20 test problems. Our results show that the proposed MI-LXPM algorithm outperforms AXNUM and RST2ANU algorithm in most of the cases. One of the important advantages of using the proposed MI-LXPM algorithm over RST2ANU algorithm is that unlike their algorithm one need not start with an initial array of feasible points (In the case of constrained optimization problems, search of feasible points is itself a big problem). During its working the algorithm automatically ensures gradual shift towards feasibility of newly generated points. In the RST2ANU algorithm feasibility of points has to be ensured at each stage which results in a large number of newly generated points being discarded because of infeasibility. It is now proposed a modified version of MI-LXPM algorithm which also takes advantage of the annealing concept. In future the proposed MI-LXPM algorithm may be compared with other stochastic approaches like PSO and DE.

Acknowledgement

One of the authors (Krishna Pratap Singh) would like to thank Council for Scientific and Industrial Research (CSIR), New Delhi, India, for providing him the financial support vide Grant number 09/143(0504)/2004-EMR-I.

Appendix

Problem-1

This problem is taken from [34] and is also given in [5,10,25].

$$\begin{aligned}
\min \quad & f(x,y) = 2x + y, \\
\text{subject to}: \quad & \\
& 1.25 - x^2 - y \leqslant 0, \\
& x + y \leqslant 1.6, \\
& 0 \leqslant x \leqslant 1.6, \\
& y \in \{0,1\}.
\end{aligned}$$

The global optimal is $(x,y,f) = (0.5, 1; 2)$.

Problem-2

This problem is taken from [25]. This is a modified form of problem in [34 and 10].

$$\begin{aligned}
\min \quad & f(x,y) = -y + 2x - \ln(x/2), \\
\text{subject to}: \quad & \\
& -x - \ln(x/2) + y \leqslant 0, \\
& 0.5 \leqslant x \leqslant 1.5, \\
& y \in \{0,1\}.
\end{aligned}$$

The global optimal is $(x,y;f) = (1.375, 1; 2.124)$.

Problem-3

This example is taken from [5]. It is also given in [10 and 25].

$$\begin{aligned}
\min \quad & f(x,y) = -0.7y + 5(x_1 - 0.5)^2 + 0.8, \\
\text{subject to}: \quad & \\
& -\exp(x_1 - 0.2) - x_2 \leqslant 0, \\
& x_2 + 1.1y \leqslant -1.0, \\
& x_1 - 1.2y \leqslant 0.2, \\
& 0.2 \leqslant x_1 \leqslant 1.0, \\
& -2.22554 \leqslant x_2 \leqslant -1.0, \\
& y \in \{0,1\}.
\end{aligned}$$

The global optimal is $(x_1, x_2, y; f) = (0.94194, -2.1, 1; 1.07654)$.

Problem-4

This problem is taken from [27].

$$\begin{aligned}
\min \quad & f(x) = (x_1 - 10)^3 + (x_2 - 20)^3, \\
\text{subject to}: \quad & \\
& (x_1 - 5)^2 + (x_2 - 5)^2 - 100 \geqslant 0.0, \\
& -(x_1 - 6)^2 - (x_2 - 5)^2 + 82.81 \geqslant 0.0, \\
& 13 \leqslant x_1 \leqslant 100, \\
& 0 \leqslant x_2 \leqslant 100.
\end{aligned}$$

The known global optimal solution is $(x_1, x_2, f) = (14.095, 0.84296; -6961.81381)$.

Problem-5

This problem is taken from [35].

$$\begin{aligned}
\min \quad & f(x) = x_1^2 + x_1 x_2 + 2x_2^2 - 6x_1 - 2x_2 - 12x_3, \\
\text{subject to}: \quad & \\
& 2x_1^2 + x_2^2 \leqslant 15.0, \\
& -x_1 + 2x_2 + x_3 \leqslant 3.0, \\
& 0 \leqslant x_i \leqslant 10; \textit{integer} \quad i = 1, \ldots, 3.
\end{aligned}$$

The best known optimal solution is $(x_1, x_2, x_3; f) = (2, 0, 5; -68)$.

Problem-6

This example represents a quadratic capital budgeting problem, taken from [34]. It is also given in Ref. [10]. It has four binary variables and features bilinear terms in objective function:

$$\begin{aligned}
\min \quad & f(x) = (x_1 + 2x_2 + 3x_3 - x_4)(2x_1 + 5x_2 + 3x_3 - 6x_4), \\
\text{subject to}: \quad & \\
& x_1 + 2x_2 + x_3 + x_4 \geqslant 4.0, \\
& x = \{0,1\}^4.
\end{aligned}$$

The global optimal solution is $(x_1, x_2, x_3, x_4; f) = (0, 0, 1, 1; -6)$.

Problem-7

This problem is taken from [25]. It is also given (but with equality constraints) in Refs. [34] and [10].

$$\begin{aligned}\min \quad & f(y_1, v_1, v_2) = 7.5y_1 + 5.5(1 - y_1) + 7v_1 + 6v_2 \\ & \quad + 50\frac{y_1/(2y_1 - 1)}{0.9[1 - \exp(-0.5v_1)]} \\ & \quad + 50\frac{1 - (y_1/(2y_1 - 1))}{0.8[1 - \exp(-0.4v_2)]}\end{aligned}$$

subject to :

$$\begin{aligned}& 0.9[1 - \exp(-0.5v_1)] - 2y_1 \leqslant 0, \\ & 0.8[1 - \exp(-0.4v_2)] - 2(1 - y_1) \leqslant 0, \\ & v_1 \leqslant 10y_1, \\ & v_2 \leqslant 10(1 - y_1), \\ & v_1, v_2 \geqslant 0, \\ & y_1 \in \{0, 1\}.\end{aligned}$$

The global optimal is $(y_1, v_1, v_2; f) = (1, 3.514237, 0; 99.245209)$.

Problem-8

This problem is taken from [25]. It is also given in [5,36;10].

$$\begin{aligned}\min \quad & f(x, y) = (y_1 - 1)^2 + (y_2 - 1)^2 + (y_3 - 1)^2, \\ & \quad - \ln(y_4 + 1) + (x_1 - 1)^2 + (x_2 - 2)^2 + (x_3 - 3)^2,\end{aligned}$$

subject to :

$$\begin{aligned}& y_1 + y_2 + y_3 + x_1 + x_2 + x_3 \leqslant 5.0, \\ & y_3^2 + x_1^2 + x_2^2 + x_3^2 \leqslant 5.5, \\ & y_1 + x_1 \leqslant 1.2, \\ & y_2 + x_2 \leqslant 1.8, \\ & y_3 + x_3 \leqslant 2.5, \\ & y_4 + x_1 \leqslant 1.2, \\ & y_2^2 + x_2^2 \leqslant 1.64, \\ & y_3^2 + x_3^2 \leqslant 4.25, \\ & y_2^2 + x_3^2 \leqslant 4.64, \\ & x_1, x_2, x_3 \geqslant 0, \\ & y_1, y_2, y_3, y_4 \in \{0, 1\}.\end{aligned}$$

The global optimal solution is:$(x_1, x_2, x_3, y_1, y_2, y_3, y_4; f) = (0.2, 1.280624, 1.954483, 1, 0, 0, 1; 3.557463)$.

Problem-9

This problem is reported in Refs. [10 and 25].

$$\begin{aligned}\max \quad & f(x, y) = -5.357854x_1^2 - 0.835689y_1x_3 - 37.29329y_1 + 40792.141, \\ \text{subject to :} \quad & \\ & a_1 + a_2y_2x_3 + a_3y_1x_2 - a_4x_1x_3 \leqslant 92.0, \\ & a_5 + a_6y_2x_3 + a_7y_1y_2 + a_8x_1^2 \leqslant 110.0, \\ & a_9 + a_{10}x_1x_3 + a_{11}y_1x_1 + a_{12}x_1x_2 \leqslant 25.0, \\ & 27 \leqslant x_1, x_2, x_3 \leqslant 45, \\ & y_1 \in \{78, 79, \ldots, 102\}, \\ & y_2 \in \{33, 34, \ldots, 45\}.\end{aligned}$$

The global optimal is $(y_1, x_1, x_3; f) = (78, 27; 32217.4)$ and it is obtained with various different feasible combination of (y_2, x_2).

Problem-10

This problem is taken from [37]. It was also studied by Cardoso et al. [10].

$$
\begin{aligned}
\max \quad & f(y) = r_1 r_2 r_3, \\
& r_1 = 1 - 0.1^{y_1} 0.2^{y_2} 0.15^{y_3}, \\
& r_2 = 1 - 0.05^{y_4} 0.2^{y_5} 0.15^{y_6}, \\
& r_3 = 1 - 0.02^{y_7} 0.06^{y_8}, \\
\text{subject to}: & \\
& y_1 + y_2 + y_3 \geqslant 1, \\
& y_4 + y_5 + y_6 \geqslant 1, \\
& y_7 + y_8 \geqslant 1, \\
& 3y_1 + y_2 + 2y_3 + 3y_4 + 2y_5 y_6 + 3y_7 + 2y_8 \leqslant 10, \\
& y \in \{0,1\}^8.
\end{aligned}
$$

The global optimal solution is $(y; f) = (0, 1, 1, 1, 0, 1, 1, 0; 0.94347)$.

Problem-11

This problem is taken from [4] and is also given in Ref. [35].

$$
\begin{aligned}
\min \quad & f(x) = x_1^2 + x_2^2 + x_3^2 + x_4^2 + x_5^2, \\
\text{subject to}: & \\
& x_1 + 2x_2 + x_4 \geqslant 4.0, \\
& x_2 + 2x_3 \geqslant 3.0, \\
& x_1 + 2x_5 \geqslant 5.0, \\
& x_1 + 2x_2 + 2x_3 \leqslant 6.0, \\
& 2x_1 + x_3 \leqslant 4.0, \\
& x_1 + 4x_5 \leqslant 13.0, \\
& 0 \leqslant x_i \leqslant 3 \quad i = 1, 2, \ldots, 5; \textit{integer}.
\end{aligned}
$$

The global optimal solution is $(x_1, x_2, x_3, x_4, x_5; f) = (1, 1, 1, 1, 2; 8)$.

Problem-12

This problem is taken from [4] and is also reported in Ref. [35].

$$
\begin{aligned}
\min \quad & f(x) = x_1 x_7 + 3x_2 x_6 + x_3 x_5 + 7x_4, \\
\text{subject to}: & \\
& x_1 + x_2 + x_3 \geqslant 6.0, \\
& x_4 + x_5 + 6x_6 \geqslant 8.0, \\
& x_1 x_6 + x_2 + 3x_5 \geqslant 7.0, \\
& 4x_2 x_7 + 3x_4 x_5 \geqslant 25.0, \\
& 3x_1 + 2x_3 + x_5 \geqslant 7.0, \\
& 3x_1 x_3 + 6x_4 + 4x_5 \leqslant 20.0, \\
& 4x_1 + 2x_3 + x_6 x_7 \leqslant 15.0, \\
& 0 \leqslant x_1, x_2, x_3 \leqslant 4, \\
& 0 \leqslant x_4, x_5, x_6 \leqslant 2, \\
& 0 \leqslant x_7 \leqslant 6, \\
& x_i; i = 1, 2, \ldots, 7 \textit{ integers}.
\end{aligned}
$$

The known global optimal solution is $(x_1, x_2, x_3, x_4, x_5, x_6, x_7; f) = (0, 2, 4, 0, 2, 1, 4; 14)$.

Problem-13

This problem is taken from [38] and is also given in Ref. [35].

$$
\begin{aligned}
\min \quad & f(x) = \exp(-x_1) + x_1^2 - x_1 x_2 - 3x_2^2 - 6x_2 + 4x_1, \\
\text{subject to}: & \\
& 2x_1 + x_2 \leqslant 8.0, \\
& -x_1 + x_2 \leqslant 2.0, \\
& 0 \leqslant x_1, x_2 \leqslant 3, \\
& x_1, x_2 \textit{ integers}.
\end{aligned}
$$

The known global optimal solution is $(x_1, x_2, f) = (1, 3; -42.632)$.

Problem-14

This problem is taken from [39] and is also given in Ref. [17].

$$\min \quad f(x) = \sum_{i=1}^{9} [\exp(-\frac{(u_i - x_2)^{x_3}}{x_1}) - 0.01i]^2,$$

$$where, u_i = 25 + (-50log(0.01i))^{2/3},$$

subject to :

$$0.1 \leqslant x_1 \leqslant 100.0,$$
$$0.0 \leqslant x_2 \leqslant 25.6,$$
$$0.0 \leqslant x_3 \leqslant 5.0.$$

Mohan and Nguyen [17] have considered this problem as a mixed integer programming problem in which x_1, x_2 are restricted to have integer values and x_3 can have both integer and continuous values. The known global optimal solution is $(x_1, x_2, x_3; f) = (50, 25, 1.5; 0.0)$.

Problem-15

This problem is taken from [40] and is also given in Ref. [17].

$$\min \quad f(x) = x_1^2 + x_2^2 + 3x_3^2 + 4x_4^2 + 2x_5^2 - 8x_1 - 2x_2 - 3x_3 - x_4 - 2x_5,$$

subject to :

$$x_1 + x_2 + x_3 + x_4 + x_5 \leqslant 400,$$
$$x_1 + 2x_2 + 2x_3 + x_4 + 6x_5 \leqslant 800,$$
$$2x_1 + x_2 + 6x_3 \leqslant 200,$$
$$x_3 + x_4 + 5x_5 \leqslant 200,$$
$$x_1 + x_2 + x_3 + x_4 + x_5 \geqslant 55,$$
$$x_1 + x_2 + x_3 + x_4 \geqslant 48,$$
$$x_2 + x_4 + x_5 \geqslant 34,$$
$$6x_1 + 7x_5 \geqslant 104,$$
$$0 \leqslant x_i \leqslant 99; \ integer \ i = 1, \ldots, 5.$$

The known optimal solution is $(x_1, x_2, x_3, x_4, x_5; f) = (16, 22, 5, 5, 7; 807)$.

Problem-16

This problem is taken from Conley [40]. It was also studied by Mohan and Nguyen [17].

$$\begin{aligned}\max \quad f(x) = {} & 215x_1 + 116x_2 + 670x_3 + 924x_4 + 510x_5 + 600x_6 + 424x_7 \\ & + 942x_8 + 43x_9 + 369x_{10} + 408x_{11} + 52x_{12} + 319x_{13} + 214x_{14} \\ & + 851x_{15} + 394x_{16} + 88x_{17} + 124x_{18} + 17x_{19} + 779x_{20} + 278x_{21} \\ & + 258x_{22} + 271x_{23} + 281x_{24} + 326x_{25} + 819x_{26} + 485x_{27} + 454x_{28} \\ & + 297x_{29} + 53x_{30} + 136x_{31} + 796x_{32} + 114x_{33} + 43x_{34} + 80x_{35} \\ & + 268x_{36} + 179x_{37} + 78x_{38} + 105x_{39} + 281x_{40}\end{aligned}$$

subject to :

$$\begin{aligned}& 9x_1 + 11x_2 + 6x_3 + x_4 + 7x_5 + 9x_6 + 10x_7, \\ & \quad + 3x_8 + 11x_9 + 11x_{10} + 2x_{11} + x_{12} + 16x_{13} + 18x_{14} \\ & \quad + 2x_{15} + x_{16} + x_{17} + 2x_{18} + 3x_{19} + 4x_{20} + 7x_{21} \\ & \quad + 6x_{22} + 2x_{23} + 2x_{24} + x_{25} + 2x_{26} + x_{27} + 8x_{28} \\ & \quad + 10x_{29} + 2x_{30} + x_{31} + 9x_{32} + x_{33} + 9x_{34} + 2x_{35} \\ & \quad + 4x_{36} + 10x_{37} + 8x_{38} + 6x_{39} + x_{40} \leqslant 25,000 \\ & 5x_1 + 3x_2 + 2x_3 + 7x_4 + 7x_5 + 3x_6 + 6x_7 \\ & \quad + 2x_8 + 15x_9 + 8x_{10} + 16x_{11} + x_{12} + 2x_{13} + 2x_{14} \\ & \quad + 7x_{15} + 7x_{16} + 2x_{17} + 2x_{18} + 4x_{19} + 3x_{20} + 2x_{21} \\ & \quad + 13x_{22} + 8x_{23} + 2x_{24} + 3x_{25} + 4x_{26} + 3x_{27} + 2x_{28} \\ & \quad + x_{29} + 10x_{30} + 6x_{31} + 3x_{32} + 4x_{33} + x_{34} + 8x_{35} \\ & \quad + 6x_{36} + 3x_{37} + 4x_{38} + 6x_{39} + 2x_{40} \leqslant 25,000 \\ & 3x_1 + 4x_2 + 6x_3 + 2x_4 + 2x_5 + 3x_6 + 7x_7 \\ & \quad + 10x_8 + 3x_9 + 7x_{10} + 2x_{11} + 16x_{12} + 3x_{13} + 3x_{14} \\ & \quad + 9x_{15} + 8x_{16} + 9x_{17} + 7x_{18} + 6x_{19} + 16x_{20} + 12x_{21} \\ & \quad + x_{22} + 3x_{23} + 14x_{24} + 7x_{25} + 13x_{26} + 6x_{27} + 16x_{28}\end{aligned}$$

$$
\begin{aligned}
&+ 3x_{29} + 2x_{30} + x_{31} + 2x_{32} + 8x_{33} + 3x_{34} + 2x_{35}\\
&+ 7x_{36} + x_{37} + 2x_{38} + 6x_{39} + 5x_{40} \leqslant 25,000\\
&10 \leqslant x_i \leqslant 99; \quad i = 1, 2, \ldots, 20,\\
&20 \leqslant x_i \leqslant 99; \quad i = 21, 22, \ldots, 40.
\end{aligned}
$$

The known optimal solution is

$$
\begin{pmatrix}
48 & 73 & 16 & 86 & 49 & 99 & 94 & 79 & 98 & 86\\
94 & 33 & 95 & 80 & 53 & 86 & 87 & 50 & 39 & 78\\
47 & 72 & 97 & 98 & 73 & 86 & 99 & 81 & 77 & 95\\
28 & 95 & 58 & 23 & 55 & 70 & 35 & 82 & 32 & 94
\end{pmatrix}
$$

with $f_{max} = 1,030,361$. This is a LPP having 40 decision variables. We have considered this problem as a linear integer problem (problem 16a) as well as mixed linear integer problem (problem 16b) with $x_i, i = 1, 3, \ldots, 39$ as an integer variables and the rest as real variables.

Problem-17

This problem is taken from [40]. It was also studied by Mohan and Nguyen [17].

$$
\begin{aligned}
\max \quad f(x) = {} & 50x_1 + 150x_2 + 100x_3 + 92x_4 + 55x_5 + 12x_6 + 11x_7\\
& + 10x_8 + 8x_9 + 3x_{10} + 114x_{11} + 90x_{12} + 87x_{13} + 91x_{14}\\
& + 58x_{15} + 16x_{16} + 19x_{17} + 22x_{18} + 21x_{19} + 32x_{20} + 53x_{21}\\
& + 56x_{22} + 118x_{23} + 192x_{24} + 52x_{25} + 204x_{26} + 250x_{27} + 295x_{28}\\
& + 82x_{29} + 30x_{30} + 29x_{31}^2 - 2x_{32}^2 + 9x_{33}^2 + 94x_{34} + 15x_{35}^{35}\\
& + 17x_{36}^2 - 15x_{37} - 2x_{38} + x_{39} + 3x_{40}^4 + 52x_{41} + 57x_{42}^2\\
& - x_{43}^2 + 12x_{44} + 21x_{45} + 6x_{46} + 7x_{47} - x_{48} + x_{49}\\
& + x_{50} + 119x_{51} + 82x_{52} + 75x_{53} + 18x_{54} + 16x_{55} + 12x_{56}\\
& + 6x_{57} + 7x_{58} + 3x_{59} + 6x_{60} + 12x_{61} + 13x_{62} + 18x_{63}\\
& + 7x_{64} + 3x_{65} + 19x_{66} + 22x_{67} + 3x_{68} + 12x_{69} + 9x_{70}\\
& + 18x_{71} + 19x_{72} + 12x_{73} + 8x_{74} + 5x_{75} + 2x_{76} + 16x_{77}\\
& + 17x_{78} + 11x_{79} + 12x_{80} + 9x_{81} + 12x_{82} + 11x_{83} + 14x_{84}\\
& + 16x_{85} + 3x_{86} + 9x_{87} + 10x_{88} + 3x_{89} + x_{90} + 12x_{91}\\
& + 3x_{92} + 12x_{93} - 3x_{94}^2 - x_{95} + 6x_{96} + 7x_{97} + 4x_{98}\\
& + x_{99} + 2x_{100}
\end{aligned}
$$

subject to :

$$
\begin{aligned}
&\sum_{i=1}^{100} x_i \leqslant 7500,\\
&\sum_{i=1}^{50} 10x_i + \sum_{i=1}^{100} x_i \leqslant 42,000,\\
&0 \leqslant x_i \leqslant 99; \quad i = 1, 2, \ldots, 100.
\end{aligned}
$$

This is a nonlinear optimization problem with one hundred decision variables. The global optimal solution of this problem is achieved at

$$
\begin{pmatrix}
51 & 10 & 90 & 85 & 35 & 36 & 75 & 98 & 99 & 30\\
56 & 23 & 10 & 56 & 98 & 94 & 63 & 8 & 27 & 92\\
10 & 66 & 69 & 10 & 39 & 38 & 49 & 8 & 95 & 96\\
86 & 14 & 1 & 55 & 98 & 64 & 8 & 1 & 18 & 99\\
84 & 78 & 4 & 19 & 85 & 33 & 59 & 95 & 57 & 48\\
37 & 95 & 62 & 82 & 62 & 62 & 87 & 38 & 95 & 14\\
91 & 21 & 72 & 85 & 68 & 69 & 30 & 30 & 85 & 93\\
73 & 19 & 26 & 62 & 94 & 59 & 53 & 11 & 0 & 1\\
2 & 26 & 43 & 50 & 42 & 93 & 27 & 71 & 61 & 93\\
44 & 94 & 15 & 92 & 8 & 18 & 42 & 27 & 66 & 49
\end{pmatrix}
$$

with $f_{max} = 303062432$. We have considered this problem as an all integer problem (Problem 17a) as well as an mixed integer problem (Problem 17b with odd numbered $x_i, i = 1, 3, 5 \ldots, 99$ as integer variables).

Problem-18

This problem is taken from [41]. It is also given in Ref. [27].

$$\max R(m,r) = \prod_{j=1}^{t}\{1-(1-r_j)^{m_j}\},$$

subject to :

$$g_1(m) = \sum_{j=1}^{4} v_j \cdot m_j^2 \leqslant v_Q,$$

$$g_2(m,r) = \sum_{j=1}^{4} C(r_j) \cdot (m_j + \exp(m_j/4)) \leqslant c_Q,$$

$$g_3(m) = \sum_{j=1}^{4} w_j \cdot (m_j \cdot \exp(m_j/4)) \leqslant w_Q,$$

$$1 \leqslant m_j \leqslant 10: \quad \textit{integer}; \quad j = 1,2,\ldots t$$

$$0.5 \leqslant r_j \leqslant 1 - 10^{-6},$$

where, v_j is the product of weight and volume per element at stage j, w_j is the weight of each component at the stage j, and C_{r_j} is the cost of each component with reliability r_j at stage j as follows:

$$C(r_j) = \alpha_j \cdot \left(\frac{-T}{\ln(r_j)}\right)^{\beta_j}$$

where, α_j and β_j are constants representing the physical characteristic of each component at stage j and T is the operating time during which the component must not fail. The known optimal solution is $R(m,r) = 0.999955$, $m = [5,5,4,6]$ and $r = [0.899845, 0.887909, 0.948990]$. The design data for this problem is given below.

No. of subsys.	4			
c_Q	400.0			
w_Q	500.0			
v_Q	250.0			
Oper. time (T)	100.0 h			
Subsys.	$10^5 \cdot \alpha_j$	β_j	v_j	w_j
1	1.0	1.5	1	6
2	2.3	1.5	2	6
3	0.3	1.5	3	8
4	2.3	1.5	2	7

References

[1] K. Deep, M. Thakur, A new crossover operator for real coded genetic algorithms, Applied Mathematics and Computation 188 (2007) 895–912.
[2] K. Deep, M. Thakur, A new mutation operator for real coded genetic algorithms, Applied Mathematics and Computation 193 (2007) 211–230.
[3] M.W. Cooper, Survey of methods for nonlinear integer programming, Management Science 27 (1981) 353–361.
[4] H.M. Salkin, Integer Programming, Eddison Wesley Publishing Com., Amsterdam, 1975.
[5] C.A. Floudas, Nonlinear Mixed-integer Optimization. Fundamentals and Applications, Oxford University Press, New York, USA, 1995.
[6] I.E. Grossmann, Review of non-linear mixed integer and disjunctive programming techniques, Optimization and Engineering 3 (2002) 227–252.
[7] H. Marchand, A. Martin, R. Weismantel, Cutting planes in integer and mixed integer programming, Discrete Applied Mathematics 123 (2002) 397–446.
[8] S. Kirkpatrick, C.D. Gelatt, M. Vecchi, Optimization by simulated annealing, Science 220 (1983) 671–680.
[9] A. Sonilah, Simulated anneling for manufacturing systems layout design, European Journal of Operational Research 82 (1995) 592–614.
[10] M.F. Cardoso, R.L. Salcedo, S.F. Azevedo, D. Barbosa, A simulated annealing approach to the solution of minlp problems, Computers and Chemical Engineering 21 (1997) 1349–1364.
[11] B.V. Babu, R. Angira, A differential evolution approach for global optimization of minlp problems, in: Proceedings of Fourth Asia Pacific Conference on Simulated Evolution and Learning, Singapore, 2002, pp. 880–884.
[12] L. Yan, K. Shen, S. Hu, Solving mixed integer nonlinear programming problems with line-up competition algorithm, Computers and Chemical Engineering 28 (2004) 2647–2657.
[13] L. Yiqing, Y. Xigang, L. Yongjian, An improved pso algorithm for solving non-convex nlp/minlp problems with equality constraints, Computers and Chemical Engineering 31 (2007) 153–162.
[14] W.L. Price, Global optimization by controlled random search, Journal of Optimization: Theory and Applications 40 (1983) 333–348.
[15] W.L. Price, Global optimization algorithms for cad workstation, Journal of Optimization: Theory and Application 55 (1987) 133–146.
[16] C. Mohan, K. Shanker, A controlled random search technique for global optimization using quadratic approximation, Asia-Pacific Journal of Operation Research 11 (1994) 93–101.
[17] C. Mohan, H.T. Nguyen, A controlled random search technique incorporating the simulating annealing concept for solving integer and mixed integer global optimization problems, Computational Optimization and Applications 14 (1999) 103–132.

[18] R.L. Salcedo, Solving nonconvex nonlinear programming and mixed-integer non-linear programming problems with adaptive random search, Industrial & Engineering Chemistry Research 31 (1992) 262–273.
[19] J.H. Holland, Adaptation in Natural and Artificial systems, University of Michigan Press, Ann Arbor, 1975.
[20] K.A. De-Jong, An Analysis of the Behavior of a Class of Genetic Adaptive Systems, Ph.D. Thesis, University of Michigan, 1975.
[21] D.E. Goldberg, Genetic Algorithms in Search, Optimization and Machine Learning, Addison-Wesley, New York, USA, 1989.
[22] K. Deb, Multi-Objective Optimization using Evolutionary Algorithms, John Wiley and Sons, 2001.
[23] B.K.S. Cheung, A. Langevin, H. Delmaire, Coupling genetic algorithm with a grid search method to solve mixed integer nonlinear programming problems, Computers and Mathematics with Applications 32 (1997) 13–23.
[24] Y.C. Luo, M. Guignard, C.H. Chen, A hybrid approach for integer programming combining genetic algorithms, linear programming and ordinal optimization, Journal of Intelligent Manufacturing 12 (2001) 509–519.
[25] L. Costa, P. Oliveria, Evolutionary algorithms approach to the solution jof mixed integer non-linear programming problems, Computers and Chemical Engineering 21 (2001) 257–266.
[26] Z. Hua, F. Huang, An efficient genetic algorithm approach to large scale mixed integer programming problems, Applied Mathematics and Computation 174 (2006) 897–907.
[27] Y.-X. Li, M. Gen, Nonlinear mixed integer programming problems using genetic algorithm and penalty function, in: Proceeding of the IEEE International Conference on Systems, Man and Cybernatics, vol. 4, 1996, pp. 2677–2682.
[28] T. Yokota, M. Gen, Y.X. Li, Genetic algorithm for nonlinear mixed integer programming problems and its applications, Computers and Industrial Engineering 30 (1996) 905–917.
[29] A.K. Maiti, A.K. Bhunia, M. Maiti, An application of real coded genetic algorithm (rcga) for mixed integer non-linear programming in two storage multi-item inventory model with discount policy, Applied Mathematics and Computation 183 (2006) 903–915.
[30] A. Ponsich, C.A. Pantel, S. Domenech, L. Pibouleau, Mixed integer nonlinear programming optimization strategies for batch plant design problems, Industrial & Engineering Chemistry Research 46 (2007) 854–863.
[31] D.E. Goldberg, K. Deb, A comparison of selection schemes used in genetic algorithms, in: Foundations of Genetic Algorithms 1, FOGA-1, vol. 1, 1991, pp. 69–93.
[32] K. Deb, An efficient constraint handling method for genetic algorithms, Computer Methods in Applied Mechanics and Engineering 186 (2000) 311–338.
[33] Bharti, Controlled Random Search Techniques and Their Applications, Ph.D. Thesis, Department of Mathematics, University of Roorkee, India, 1994.
[34] G.R. Kocis, I.E. Grossmann, Global optimazation of nonconvex mixed-integer nonlinear programming (minlp) problems in process synthesis, Industrial & Engineering Chemistry Research 27 (1988) 1407–1421.
[35] H.T. Nguyen, Some Global Optimization Techniques and Their Use in Solving Optimization Problems in Crisp and Fuzzy Environments, Ph.D. Thesis, Department of Mathematics, University of Roorkee, Roorkee, India, 1996.
[36] X. Yuan, S. Zhang, L. Pibouleau, S. Domenech, une methode d'optimization nonlineaire en variables mixtes pour la conception de proceds, RAIRO Operations Research 22 (1988) 131–146.
[37] O. Berman, N. Ashrafi, Optimization models for reliability of modular software systems, IEEE Transactions on Software Engineering 19 (1993) 11–19.
[38] M.S. Bazaraa, H.D. Sherah, C.M. Shetty, Nonlinear Programming: Theory and Algorithms, second ed., John Wiley and Sons, Asia, 2004.
[39] D.M. Himmelblau, Applied Nonlinear Programing, McGraw Hill, New York, USA, 1972.
[40] W. Conley, Computer Optimization Techniques, Petrocelli Books, Newjersy, USA, 1984.
[41] A.K. Dhingra, Optimal apportionment of reliability and redundancy in series systems under multiple objectives, IEEE Transactions on Reliability 41 (1992) 576–582.