Genetic Algorithm Designed for Solving Linear or Nonlinear Mixed-Integer Constrained Optimization Problems

Hemant Jalota and Manoj Thakur

Abstract Genetic algorithms (GA) initially were not developed to handle the integer restriction or discrete values to the design variables. In the recent years, researchers has focused their work on developing/modifying GAs for handling integer/discrete variables. We have modified the BEX-PM algorithm developed by Thakur et al. [1] to solve the nonlinear constrained mixed-integer optimization problems. Twenty test problems have been used to conduct a comparative study to test the effectiveness of proposed algorithm (MI-BEXPM) with other similar algorithms (viz. MILXPM, RST2ANU, and AXNUM) in this class available in the literature. The efficacy of the results acquired through MI-BEXPM is compared with other algorithms on two well-known criteria. The performance of MI-BEXPM is also analysed and compared for solving real-life mixed-integer optimization problems with other methods available in the literature. It is found that MI-BEXPM is significantly superior to the algorithms considered in this work.

Keywords Real-coded genetic algorithms · Bounded exponential crossover Power mutation · Mixed-integer optimization · Constraint optimization

1 Introduction

Mixed-integer nonlinear programming problem (MINLP) is the important class of nonlinear optimization problems. A MINLP is an optimization problem where the objective functions and constraints are nonlinear functions of the decision variables with some of the decision variables having integer restriction. If objective functions as well as constraints are linear function, then the corresponding problem is called a

H. Jalota (✉) · M. Thakur
Indian Institute of Technology Mandi, Mandi 175001, Himachal Pradesh, India
e-mail: hemant.jalota4@gmail.com

M. Thakur
e-mail: manojpma@gmail.com

M.S. Reddy et al. (eds.), *International Proceedings on Advances in Soft Computing, Intelligent Systems and Applications*, Advances in Intelligent Systems and Computing 628, https://doi.org/10.1007/978-981-10-5272-9_27

mixed-integer linear programming problem (MILP). A MINLP having the all the variable as integer is called integer nonlinear programming problem (INLP).

In last few decades, several population-based heuristic algorithms have been designed. These algorithms try to search the global optimal solution of general nonconvex optimization problems. Several variants of these algorithms have been suggested to handle constraints and for solving mixed-integer optimization problems. Two of the major classes of algorithms where the research had been focused are evolutionary algorithms and swarm-based algorithms. Evolutionary algorithms try to mimic the process of natural evolution. Some of the algorithms which belong to the class of evolutionary algorithms are genetic algorithms, genetic programming, evolutionary strategies, evolutionary programming.

Swarm intelligence is based on group behaviour of simple individuals (swarms) in which individuals independently may not show intelligence but as group they show intelligent behaviour. Swarm intelligence is intelligent behaviour shown by a system which emerges due to cooperative interaction of components of the system to achieve a goal which may not be achievable by individual efforts. Ant colony optimization, particle swarm optimization, and artificial bee colony algorithm are among some of the most popular swarm intelligence-based techniques used to solve optimization problems.

Evolutionary algorithms and swarm intelligence have been quite successful in solving many engineering applications having highly nonlinear, nonconvex, nondifferentiable, and multimodal models, and a variety of modifications have been proposed to tackle MINLPs. Many algorithms based on EAs have been effectively used to find the solution of MINLP problem [2, 3].

Lin [4] designed a mimetic algorithm combined with an evolutionary Lagrange method for solving MINLPs. Lin et al. [5] proposed a hybrid differential evolution method to solve MINLPs. The method works with two phases called accelerated phase and migrating phase used to maintain exploration and exploitation. Later a modified coevolutionary hybrid differential evolution for MINLPs was suggested by Lin et al. [6]. Yan et al. [7] introduce a memory-based lineup competition algorithm having cooperation and bi-level competition mechanism for exploration and exploitation. Xiong et al. [8] introduced a hybrid genetic algorithm for finding a globally compromise solution of a mixed-discrete fuzzy nonlinear programming. Cheung et al. [9] developed a hybrid algorithm which combines genetic algorithm and grid search to solve MINLP.

Cardoso et al. [10] presented a modified simulated annealing (M-SIMPSA). This method uses combination of simulate annealing and Nelder and Mead simplex method in the inner loop and Metropolis algorithm ([11–13]) in the outer loop. Some of the other population-based algorithms used to solve MINLP are simulated annealing technique [10, 14], tabu search method [15], multistart scatter search [16], and particle swarm optimization [17].

Among the above-discussed methods, genetic algorithms (GAs) [3, 9, 18, 19] are the most successful. GA is an iterative process that works with a set of the solutions (population) which are modified by genetic operators to guide the solution towards the optimum solution in the search space. Crossover and mutation are one of the essential operators of GA. Crossover helps in exploring the promising zones of the search space, using the information from chromosomes (solutions), and mutation assists in avoiding premature convergence by preserving sufficient disparity within the population. This work extends the recently developed RCGA, BEXPM by Thakur et al. [1], named as "MI-BEXPM". The performance of MI-BEXPM is analysed and compared with other algorithms on the basis of twenty test problems as well as real-life problems.

The rest of paper is organized as the following: Sect. 2 proposed the algorithm to find the solution of the mixed-integer optimization problem. The experimental setup used in current study is detailed in Sect. 3. Analysis of the results for twenty test problems and discussion is given in Sect. 4. The efficiency of MI-BEXPM for solving real-life mixed-integer optimization problems is analysed and compared with other algorithms in Sect. 5. Finally, conclusions from the comparative study are drawn in Sect. 6.

2 Proposed GA

GA belongs to a class of population-based iterative algorithms, which tries to find the near global optimal solution of an optimization problem. The search in GA is governed by three main genetic operators, viz. selection, crossover, and mutation. These operators are applied iteratively to direct the search during the evolution of the population during the search process. We will discuss about the operators used in this study in the following subsections.

2.1 Selection

Selection operator works on the principle of the survival of fittest. It is used to discard the inferior individuals from the population and filter relatively better fit individuals to participate in the biological evolution process. After applying selection operator, an intermediate pool of the population (mating pool) is constructed. Lot of selection techniques have been proposed in the literature. Some of the popular selection operators are ranking [20], roulette wheel [21], stochastic uniform sampling (SUS) [22], and tournament [20] which are widely used selection operators. We have employed tournament selection operator in this work. Tournament selection selects a subset of the population randomly and conducts a fitness-based competition among the chosen solutions. The cardinality of this subset is called tournament size. The winner of the tournament becomes a part of mating pool. The process is repeated until the cardinality of this mating pool becomes equal to the population size.

2.2 Crossover

Crossover operator in GA mimics the process of chromosomal crossover in biology to produce the recombinant chromosomes. Individuals from mating pool are randomly chosen to participate in the crossover process with certain probability called crossover probability (p_c). Here, we used BEX crossover [1] to produce a pair of offspring solutions within the variable bounds from a pair of parent solutions lying within the variable bounds. BEX crossover is a parent-centric operator and has one scale parameter λ. For small (large) values of λ, offspring produced are spread near (away) from the parents. Also for a fixed value of λ, the spread of child chromosome is proportional to the that of the parent solutions. Steps to generate child chromosomes C_1 and C_2 using parent chromosomes P_1 and P_2 via BEX crossover operator are as follows:

1. Randomly choose two parent chromosomes P_1 and P_2 from the mating pool (population after employing GA operator).
2. Randomly generate a uniform number $r_c \in (0,1)$. If r_c is less than prescribed crossover rate p_c, then crossover is applied to P_1 and P_2, otherwise it will pass as such for mutation.
3. If $r_c < p_c$, then C_1 and C_2 is produced using Eqs. (1) and (2)

$$C_1 = P_1 + \gamma_1 |P_2 - P_1| \tag{1}$$

$$C_2 = P_2 + \gamma_2 |P_2 - P_1| \tag{2}$$

where

$$\gamma_j = \begin{cases} \lambda\, ln\left\{\exp\left(\frac{B_l - P_j}{\lambda|P_2 - P_1|}\right) + u\left(1 - \exp\left(\frac{B_l - P_j}{\lambda|P_2 - P_1|}\right)\right)\right\} & \text{if}\, p \le 0.5 \\ -\lambda\, ln\left\{1 - u\left(1 - \exp\left(\frac{B_u - P_j}{\lambda|P_2 - P_1|}\right)\right)\right\} & \text{if}\, p > 0.5 \end{cases}$$

for $j \in \{1,2\}$, $u, p \in (0,1)$ are random numbers following uniform distribution, $\lambda > 0$ is a scaling parameter, $B_l = \{B_l^1, B_l^2, \ldots, B_l^n\}$ and $B_u = \{B_u^1, B_u^2, \ldots, B_u^n\}$ are lower and upper bound of the decision variable.

2.3 Mutation

It is inspired from the biological mutation in which changes in a DNA sequence occurs by altering one or more gene values of the chromosome. Mutation operator in GA is applied for minimizing the possibility to stick into the local or suboptimal solution. It gives a small random perturbation to explore the neighbourhood of the current solution. Not all the individuals go through the mutation phase. It is applied with a relatively small probability (p_m) called probability of mutation and tries to give a random drift to solution to be in a promising zones of the search space. Here, power mutation [23] is being used for this purpose. The search power of mutation is controlled by index parameter (p). Larger (smaller) the value of p has higher (smaller) possibility to introduce perturbation in the muted solution. The probability of generating mutated solution on either side is proportional to its relative position from the variable bounds of the decision variable. The muted solution (x_i^{k+1}) from the current solution (x_i^k) is produced as follows:

$$x_j^{k+1} = \begin{cases} x_j^k - t_j\left(x_j^k - L_j\right) & \text{if } \frac{x_j^k - L_j}{U_j - L_i} \leq s \\ x_j^k + t_j\left(U_j - x_j^k\right) & \text{Otherwise} \end{cases}$$

Here, k refers to the current generation, $s \in (0,1)$, t_j $(j \in \{1,2,\ldots,n_v\}$; $n_v = \#$ of decision variables) are random numbers which follow uniform distribution and power distribution, respectively.

2.4 Truncation Technique

In this work, truncation technique based on floor and ceiling function is applied to each $x_i \in I$ (here, I refer to the set of variables having integer restrictions). It helps in maintaining the randomness within the newly generated population and reduces the chance of producing similar integer value for same real values that lies within two similar successive integer values [18, 24].

$$x_i^{k+1} = \begin{cases} \lfloor x_i^k \rfloor & \text{if } p \leq 0.5 \\ \lceil x_i^k \rceil & \text{Otherwise} \end{cases}$$

where $p \in (0,1)$ is a uniform random number.

2.5 Constraint Handling Technique

Due to non-requirement of penalty parameter unlike other penalty constraint handling techniques, parameter-free penalty (PFP) method suggested by Deb [25] is applied to handling constraints. This approach can be easily embedded and executed while evaluating the fitness function during the search process. The fitness function using PFP is evaluated as follows

$$\Theta(z) = \begin{cases} G(z); & \text{if } z \text{ is feasible} \\ G_{\max}(z) + \sum_{k=1}^{r_1} (\Gamma_k(z)) + \sum_{k=1}^{r_2} |\zeta_j|; & \text{otherwise} \end{cases}$$

where $G_{\max}(z)$, Γ_k, r_1, ζ_j, and r_2 are the worst feasible value, inequality constraint, # of inequality constraints, equality constraint, and # of equality constraints, respectively.

3 Experimental Setup

The test bed selected for the comparative study consists of a number of real and/or integer decision variables having linear and/or nonlinear inequality constraints. The best/optimum solutions reported in the literature are summarized in Table 1.

Table 1 Problems considered for study

Problem	Variables	Objective function	Inequality constant	Global optimum value	References
1	2(1+1+0)	Linear	2(1+1)	2	[3, 10, 18, 26]
2	2(1+1+0)	Nonlinear	1(0+1)	2.124	[3, 10, 18]
3	3(2+1+0)	Quadratic	3(2+1)	1.07654	[3, 10, 18, 26]
4	2(2+0+0)	Cubic	2(0+2)	−6961.81	[18]
5	3(0+3+0)	Quadratic	2(1+1)	−68	[18]
6	4(0+4+0)	Quadratic	1(1+0)	−6	[10, 18]
7	3(2+1+0)	Nonlinear	4(2+2)	99.24521	[3, 10, 18]
8	7(3+4+0)	Nonlinear	9(5+4)	3.557463	[3, 10, 18, 26]
9	5(3+2+0)	Quadratic	3(0+3)	32217.4	[3, 10, 18]
10	8(0+8+0)	Nonlinear	3(3+0)	0.94347	[10, 18]
11	5(0+5+0)	Quadratic	6(6+0)	8	[18]
12	7(0+7+0)	Nonlinear	6(3+3)	14	[18]
13	2(0+2+0)	Nonlinear	2(2+0)	−42.632	[18]
14	3(1+2+0)	Nonlinear	0(0+0)	0	[18, 24]
15	5(0+5+0)	Quadratic	8(8+0)	807	[18, 24]
16	40(0+40+0)	Linear	3(3+0)	1030361	[18, 24]
17	40(20+20+0)	Linear	3(3+0)	1030361	[18, 24]
18	100(0+100+0)	Nonlinear	2(2+0)	3.03E+08	[18, 24]
19	100(50+50+0)	Nonlinear	2(2+0)	3.03E+08	[18, 24]
20	8(4+4+0)	Nonlinear	3(0+3)	0.999955	[18]
Gear train	4(0+4+0)	Nonlinear	0	–	[27]
Reinforced concrete beam	3(1+1+1)	Nonlinear	0	–	[28]
Speed reducer	7(6+1)	Nonlinear	0	–	[29]
Welded beam (A)	4(3+0+1)	Nonlinear	0	–	[30]
Welded beam (B)	6(1+1+4)	Nonlinear	0	–	[31]

Table 2 Parameter settings of MI-BEXPM

Parameter	Values
Crossover rate	0.9
Mutation rate	0.009
Crossover index for real variables	0.35
Mutation index for real variables	10.0
Crossover index for integer variables	0.65
Mutation index for integer variables	4.0
Elitism size	1
Tournament size	2

As discussed in [18], each problem is run 100 times with distinct initial population. Here, successful run is that run whose objective function value lies within 1% range of the reported best/optimal solution. For each problem percentage of successful runs (*ps*), average function evaluations of successful runs (avg) are measured.

$$ps = \frac{T_s * 100}{T_r} \tag{3}$$

$$\text{avg} = \frac{T_f}{T_s} \tag{4}$$

Here, T_s = total # of successful runs, T_r = total runs, and T_f = sum of function evaluations of successful runs.

The parameter settings of MI-BEXPM algorithm used to conduct this experiment are shown in Table 2. Population size = 10 times the total number of decision variable for each considered problem except 16, 17, and 18 problems (chosen to be three times).

4 Results and Discussions

Results observed using MI-BEXPM are compared with MILXPM, RST2ANU, and AXNUM on the basis of ps and avg for twenty problems. Table 3 demonstrates successful runs for each problem corresponding to each algorithm for twenty test problems. MI-BEXPM shows 100% success rate in twelve problems, whereas MILXPM, RST2ANU, and AXNUM show 100% success rate only in ten, eleven, and eight problems, respectively. Moreover, the minimum success rate of MI-BEXPM to solve problem is greater than 50% for each problem, while MILXPM, RST2ANU (unsuccessful to obtain optimal solution within 100 runs for 4th problem), and AXNUM have two, seven, and six problems, respectively, which have less than 50% success rate.

Table 3 Percentage of successful runs and average function evaluations of successful runs—twenty problems

Problem No.	Percentage of successful runs				Average function evaluations of successful runs			
	MI-BEXPM	MILXPM	RST2ANU	AXNUM	MI-BEXPM	MILXPM	RST2ANU	AXNUM
1	85	84	47	86	135	172	173	1728
2	89	85	57	67	100	64	657	82
3	68	43	4	35	813	18608	221129	65303
4	80	95	2	82	221	10933	1489713	45228
5	100	100	75	95	84	671	2673	13820
6	100	100	100	100	80	84	108	432
7	57	59	0	45	916	7447	–	16077
8	71	41	15	3	2064	3571	180859	1950
9	100	100	100	100	100	100	189	4946
10	100	93	100	33	160	258	545	700
11	95	100	100	97	275	171	2500	863
12	87	71	29	19	1159	299979	6445	380115
13	100	99	100	91	40	77	35	456
14	100	100	100	100	70	78	214	1444
15	100	92	19	9	11067	2437	3337	267177
16	100	100	100	100	240	1075	1114	2950
17	100	100	100	100	240	1073	1189	3016
18	100	100	100	100	600	600	2804	600
19	100	100	100	100	600	600	1011	600
20	100	100	100	100	160	250	697	256

MI-BEXPM, MILXPM, and AXNUM show equivalent success rate for Problem 1, but MI-BEXPM has lesser average number of function execution (shown in Table 3), while in problem-4 and problem-7 MILXPM outperforms better in terms of success rate but MI-BEXPM performs better in average number of function evaluation than other algorithms. And for problem-11, MILXPM performs better than other algorithms in terms of both *ps* and avg.

For overall performance comparison of MI-BEXPM for the chosen test suite, performance index (PI) is applied, suggested by Bharti [32] and used by several to compare the performance of the algorithm [18, 23, 24, 33]. It is based on successful run, average number of function evaluations, and average time of execution of successful runs. In the current study, all the algorithms considered to be compared are not run on same machine so it is absurd to consider time. Consequently

$$\mathrm{PI} = \frac{1}{P_n}\sum_{i=1}^{P_n}\left(k_1 s_1^i + k_2 s_2^i\right) \tag{5}$$

where s_1^i and s_2^i is the ratio of T_s to T_r and minimum of average function evaluation among algorithm to the average function evaluation of each algorithm, respectively, for the ith problem. k_j is the weight correspond to each $s_j, j = \{1, 2\}$ such that $\sum_{j=1}^{2} k_j = 1$. Assume $\{k_1 = k\}$, then $k_2 = \{1 - k\}$. From Fig. 1, it can be observed that MI-BEXPM is superior than MILXPM, RST2ANU, and AXNUM in terms of *PI*.

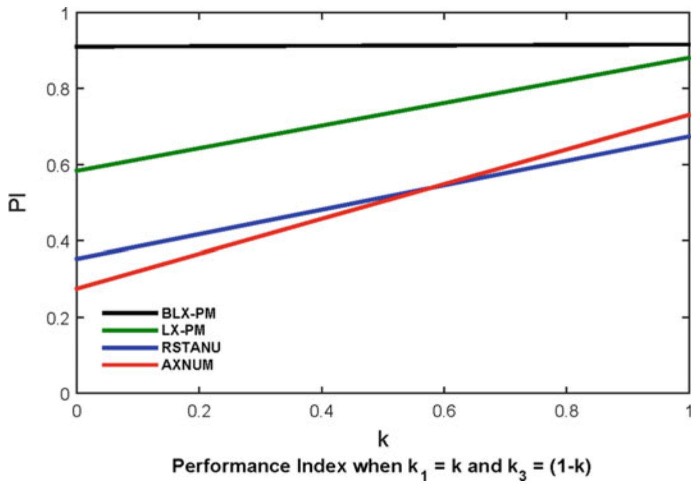

Fig. 1 Performance index

5 Application of MI-BEXPM

In previous section, we have demonstrated the effectiveness of MI-BEXPM for solving benchmark test problems. In this section, we further analyse the performance of MI-BEXPM on a set of mixed-integer real-life problems. The problems considered for this purpose are some of the popular problems available in the literature (stated in Table 1). The parameter setting of MI-BEXPM while solving these problems is kept same as discussed in Sect. 3.

The best results obtained by MI-BEXPM and those reported in the literature are shown in Table 4. From these results, it is observed that result obtained by MI-BEXPM is better than that reported in [27, 34, 35]. Cases where the results are similar to [31, 36–39], it is uses lesser function calls.

Table 5 refers to the results obtained by MI-BEXPM and reported in the literature. From these results, it can be easily observed that solution obtained by MI-BEXPM and reported by Gandomi et al. [30] are superior to rest of the algorithms considered. It is to be noted that the best solution obtained by MI-BEXPM and stated in Gandomi et al. [30] are same, but MI-BEXPM is able to solve this problem with lesser number of function evaluation than Gandomi et al. [30].

Table 4 Results of designing of gear train

Author(s)	β_1	β_2	β_3	β_4	Gear	Minimum	Function evaluation
Sandgren [27]	18	22	45	60	0.146667	5.70E−06	–
Kannan and Kramer [34]	13	15	33	41	0.144124	2.20E−08	–
Zhang and Wang [35]	30	15	52	60	0.14423	2.40E−09	–
Deb and Goyal [31]	19	16	49	43	0.144281	2.70E−12	–
Gandomi et al. [36]	19	16	43	49	0.144281	2.70E−12	5,000
Parsopoulos and Vrahatis [37]	19	16	43	49	0.144281	2.70E−12	100,000
Gandomi et al. [38]	16	19	49	43	0.144281	2.70E−12	2,000
Ali et al. [39]	16	19	43	49	0.144281	2.70E−12	1,500
MI-BEXPM	16	19	43	49	0.144281	2.70E-12	1,273

Table 5 Results of designing of reinforced concrete beam

Attributes	Amir [28]	Yun (GA) [40]	Yun (GA-FL) [40]	Montes and Ocaña [41]	Gandomi et al. [30]	MI-BEXPM
Best	374.200	366.146	364.854	376.298	359.208	359.208
$A(\beta_1)$	7.800	7.200	6.160	–	6.320	6.320
β_2	31	32	35	–	34	34
α_1	7.790	8.045	8.750	–	8.500	8.500
Γ_1	−4.201	−2.878	−3.617	–	−0.224	−0.224
Γ_2	−0.021	−0.022	0	–	0	0
Function evaluation	396	100,000	100,000	30,000	25,000	9,457

Tables 6, 7 and 8 demonstrates the best results obtained by MI-BEXPM and available in the literature. The solution obtained by MI-BEXPM for these problems is better among the feasible solutions considered in this study.

Table 6 Results of designing of speed reducer

Attributes	Ray and Saini [42]	Kuang et al. [29]	Mollinetti et al. [43]	MI-BEXPM	Gandomi et al. [36]	Akhtar et al. [44]	Montes et al. [45]
Best	2732.901†	2876.118†	2894.901†	2996.36	3000.981	3008.08	3025.005
β	17	17	17	17	17	17	17
α_1	3.514185	3.6	3.5	3.50001	3.5015	3.506122	3.506163
α_2	0.700005	0.7	0.7	0.7	0.7	0.700006	0.700831
α_3	7.497343	7.3	7.3	7.30006	7.605	7.549126	7.460181
α_4	7.8346	7.8	7.8	7.8	7.8181	7.85933	7.962143
α_5	2.9018	3.4	2.9	3.35024	3.352	3.365576	3.3629
α_6	5.0022	5	5.286683	5.28669	5.2875	5.289773	5.309
Γ_1	−0.0777	−0.0996	−0.07392	−0.07392	−0.0743	−0.0755	−0.0777
Γ_2	−0.2012	−0.2203	−0.198	−0.198	−0.1983	−0.1994	−0.2013
Γ_3	−0.036	−0.5279	−0.10795	−0.49918	−0.4349	−0.4562	−0.4741
Γ_4	−0.8754	−0.8769	−0.90147	−0.90147	−0.9008	−0.8994	−0.8971
Γ_5	0.5395	−0.0433	0.541785	−2.26E−05	−0.0011	−0.0132	−0.011
Γ_6	0.1805	0.1821	1.30E−07	−3.84E−06	−0.0004	−0.0017	−0.0125
Γ_7	−0.7025	−0.7025	−0.7025	−0.7025	−0.7025	−0.7025	−0.7022
Γ_8	−0.004	−0.0278	0	−2.86E−06	−0.0004	−0.0017	−0.0006
Γ_9	−0.5816	−0.5714	−0.79583	−0.79583	−0.5832	−0.5826	−0.5831
Γ_{10}	−0.166	−0.0411	−0.14384	−0.05133	−0.089	−0.0796	−0.0691
Γ_{11}	−0.0552	−0.0513	−0.198	−0.01085	−0.013	−0.0179	−0.0279
Mean	2758.888	–	2894.901	2996.38	3007.2	3012.12	3088.778
Worst	2780.307	–	–	2996.43	3009.0	3028.28	3078.592
SD	–	–	0.00E+00	0.01586	4.9634	–	–

†Values refer to the infeasible constraints

Table 7 Results of designing of welded beam (A)

Method	α_1	α_2	α_3	α_4	Γ_1	Γ_1	Γ_3	Γ_4	Γ_5	Best
MI-BEXPM	0.240	6.356	8.294	0.2443	0.2070	16.351	0.0038	0.0312	0.2342	2.390
Gandomi et al. [30]	0.201	3.562	9.041	0.2057	9800	−27.368	−0.0042	2210	−0.2355	1.73†

†Values refer to the infeasible constraints

Table 8 Results of designing of welded beam (B)

Method	α_1	α_2	α_3	α_4	α_5	α_6	Γ_1	Γ_2	Γ_3	Γ_4	Best
MI-BEXPM	0.187	1.6848	8.25	0.25	4	Steel	−3.80E+02	−4.02E+02	−0.2343	−0.0159	1.9418
Deb et al. [31]	0.187	1.6849	8.25	0.25	4	Steel	−3.80E+02	−4.02E+02	−0.2346	−0.1621	1.9422
Wang and Yin [46]	0.187	1.6842	8.25	0.25	4	Steel	−3.80E+02	−4.02E+02	−0.2343	2.4199	1.9421†

†Values refer to the infeasible constraints

6 Conclusions

In the present study, BEX-PM GA developed for continuous variable constrained optimization problems is modified to solve mixed-integer variables constraint optimization problems. The new variant of BEX-PM GA is named as MI-BEXPM. The results obtained are compared on a set of twenty mixed-integer constrained optimization benchmark problems mentioned in the literature. It is observed that MI-BEXPM outperforms other algorithms MILXPM, RST2ANU, and AXNUM individually in several problems. The overall performance is found to be better than these algorithms on the basis of performance index used extensively in the literature for such type of comparative studies.

Moreover, the performance of MI-BEXPM has been analysed for solving real-life mixed-integer optimization problems in comparison with other existed methods from the literature. Analysing the obtained results, MI-BEXPM not only performs well for test benchmark problems, also performs well for real-life problems.

From the above study, it can be concluded that MI-BEXPM is a promising algorithm among the class of algorithms considered in this study for solving test problems as well as real-life problems.

References

1. Thakur, M., S.S. Meghwani, and H. Jalota. 2014. A modified real coded genetic algorithm for constrained optimization. *Applied Mathematics and Computation* 235: 292–317.
2. Yokota, T., M. Gen, Y. Li, and C.E. Kim. 1996. A genetic algorithm for interval nonlinear integer programming problem. *Computers and industrial engineering* 31 (3): 913–917.
3. Costa, L., and P. Oliveira. 2001. Evolutionary algorithms approach to the solution of mixed integer non-linear programming problems. *Computers and Chemical Engineering* 25 (2): 257–266.
4. Lin, Y. 2013. Mixed-integer constrained optimization based on memetic algorithm. *Journal of applied research and technology* 11 (2): 242–250.
5. Lin, Y.C., K.S. Hwang, and F.S. Wang. 2004. A mixed-coding scheme of evolutionary algorithms to solve mixed-integer nonlinear programming problems. *Computers and Mathematics with Applications* 47 (8): 1295–1307.
6. Lin, Y.C., K.S. Hwang, and F.S. Wang. 2001. Co-evolutionary hybrid differential evolution for mixed-integer optimization problems. *Engineering Optimization* 33 (6): 663–682.

7. Yan, L., K. Shen, and S. Hu. 2004. Solving mixed integer nonlinear programming problems with line-up competition algorithm. *Computers and Chemical Engineering* 28 (12): 2647–2657.
8. Xiong, Y., and S.S. Rao. 2004. Fuzzy nonlinear programming for mixed-discrete design optimization through hybrid genetic algorithm. *Fuzzy Sets and Systems* 146 (2): 167–186.
9. Cheung, B.S., A. Langevin, and H. Delmaire. 1997. Coupling genetic algorithm with a grid search method to solve mixed integer nonlinear programming problems. *Computers and Mathematics with Applications* 34 (12): 13–23.
10. Cardoso, M.F., R. Salcedo, S.F. de Azevedo, and D. Barbosa. 1997. A simulated annealing approach to the solution of MINLP problems. *Computers and Chemical Engineering* 21 (12): 1349–1364.
11. Metropolis, N., A.W. Rosenbluth, M.N. Rosenbluth, A.H. Teller, and E. Teller. 1953. Equation of state calculations by fast computing machines. *The Journal of Chemical Physics* 21 (6): 1087–1092.
12. Kirkpatrick, S., M. Vecchi, et al. 1983. *Optimization by simulated annealing. science* 220 (4598): 671–680.
13. Kirkpatrick, S. 1984. Optimization by simulated annealing: quantitative studies. *Journal of Statistical Physics* 34 (5–6): 975–986.
14. Wah, B.W., Y. Chen, and T. Wang. 2007. Simulated annealing with asymptotic convergence for nonlinear constrained optimization. *Journal of Global Optimization* 39 (1): 1–37.
15. Exler, O., L.T. Antelo, J.A. Egea, A.A. Alonso, and J.R. Banga. 2008. A tabu search-based algorithm for mixed-integer nonlinear problems and its application to integrated process and control system design. *Computers and Chemical Engineering* 32 (8): 1877–1891.
16. Ugray, Z., L. Lasdon, J.C Plummer, F. Glover, J. Kelly, J., and R. Marti. 2005. A multistart scatter search heuristic for smooth NLP and MINLP problems. In *Metaheuristic optimization via memory and evolution*, 25–57. Springer.
17. dos Santos Coelho, L. 2009. An efficient particle swarm approach for mixed-integer programming in reliability redundancy optimization applications. *Reliability Engineering and System Safety* 94 (4): 830–837.
18. Deep, K., K.P. Singh, M. Kansal, and C. Mohan. 2009. A real coded genetic algorithm for solving integer and mixed integer optimization problems. *Applied Mathematics and Computation* 212 (2): 505–518.
19. Hua, Z., and F. Huang. 2006. An effective genetic algorithm approach to large scale mixed integer programming problems. *Applied Mathematics and Computation* 174 (2): 897–909.
20. Goldberg, D.E., and K. Deb. 1991. A comparative analysis of selection schemes used in genetic algorithms. *Urbana* 51: 61801–2996.
21. da Silva, E.L., H.A. Gil, and J.M. Areiza. 1999. Transmission network expansion planning under an improved genetic algorithm. In *Proceedings of the 21st 1999 IEEE international conference on power industry computer applications*, PICA'99, 315–321. IEEE.
22. Blickle, T., and L. Thiele. 1996. A comparison of selection schemes used in evolutionary algorithms. *Evolutionary Computation* 4 (4): 361–394.
23. Deep, K., and M. Thakur. 2007. A new mutation operator for real coded genetic algorithms. *Applied Mathematics and Computation* 193 (1): 211–230.
24. Mohan, C., and H. Nguyen. 1999. A controlled random search technique incorporating the simulated annealing concept for solving integer and mixed integer global optimization problems. *Computational Optimization and Applications* 14 (1): 103–132.
25. Deb, K. 2000. An efficient constraint handling method for genetic algorithms. *Computer Methods in applied Mechanics and Engineering* 186 (2): 311–338.
26. Floudas, C.A. 1995. *Nonlinear and mixed-integer optimization: fundamentals and applications*. Oxford University Press.
27. Sandgren, E. 1990. Nonlinear integer and discrete programming in mechanical design optimization. *Journal of Mechanical Design* 112 (2): 223–229.
28. Amir, H.M., and T. Hasegawa. 1989. Nonlinear mixed-discrete structural optimization. *Journal of Structural Engineering* 115 (3): 626–646.

29. Ku, K.J., S. Rao, and L. Chen. 1998. Taguchi-aided search method for design optimization of engineering systems. *Engineering Optimization* 30 (1): 1–23.
30. Gandomi, A.H., X.S. Yang, and A.H. Alavi. 2011. Mixed variable structural optimization using firefly algorithm. *Computers and Structures* 89 (23): 2325–2336.
31. Deb, K., and M. Goyal. 1996. A combined genetic adaptive search (GeneAS) for engineering design. *Computer Science and Informatics* 26: 30–45.
32. Bharti. 1994. Controlled random search technique and their applications. PhD thesis, Department of Mathematics, University of Roorkee, Roorkee, India.
33. Deep, K., and M. Thakur. 2007. A new crossover operator for real coded genetic algorithms. *Applied Mathematics and Computation* 188 (1): 895–911.
34. Kannan, B., and S.N. Kramer. 1994. An augmented lagrange multiplier based method for mixed integer discrete continuous optimization and its applications to mechanical design. *Journal of Mechanical Design* 116 (2): 405–411.
35. Zhang, C., and H.P. Wang. 1993. Mixed-discrete nonlinear optimization with simulated annealing. *Engineering Optimization* 21 (4): 277–291.
36. Gandomi, A.H., X.S. Yang, and A.H. Alavi. 2013. Cuckoo search algorithm: a metaheuristic approach to solve structural optimization problems. *Engineering with Computers* 29 (1): 17–35.
37. Parsopoulos, K.E., and M.N. Vrahatis. 2005. United particle swarm optimization for solving constrained engineering optimization problems. In *Advances in natural computation*, 582–591. Springer.
38. Gandomi, A.H., G.J. Yun, X.S. Yang, and S. Talatahari. 2013. Chaos-enhanced accelerated particle swarm optimization. *Communications in Nonlinear Science and Numerical Simulation* 18 (2): 327–340.
39. Ali, M.Z., N.H. Awad, P.N. Suganthan, R.M. Duwairi, and R.G. Reynolds. 2016. A novel hybrid cultural algorithms framework with trajectory-based search for global numerical optimization. *Information Sciences* 334: 219–249.
40. Yun, Y. 2005. Study on adaptive hybrid genetic algorithm and its applications to engineering design problems. Master's thesis, Waseda University.
41. Mezura-Montes, E., B. Hernández-Ocaña. 2009. Modified bacterial foraging optimization for engineering design. In *Intelligent engineering systems through artificial neural networks*. ASME Press.
42. Ray, T., and P. Saini. 2001. Engineering design optimization using a swarm with an intelligent information sharing among individuals. *Engineering Optimization* 33 (6): 735–748.
43. Mollinetti, M.A.F., D.L. Souza, R.L., Pereira, E.K.K. Yasojima, O.N. Teixeira, O.N. 2016. ABC+ES: Combining artificial bee colony algorithm and evolution strategies on engineering design problems and benchmark functions. In *Hybrid intelligent systems*, 53–66. Springer.
44. Akhtar, S., K. Tai, and T. Ray. 2002. A socio-behavioural simulation model for engineering design optimization. *Engineering Optimization* 34 (4): 341–354.
45. Mezura-Montes, E., C.A.C. Coello, and R. Landa-Becerra. 2003. Engineering optimization using simple evolutionary algorithm. In *Proceedings of 15th IEEE international conference on tools with artificial intelligence*, 149–156. IEEE.
46. Wang, J., and Z. Yin. 2008. A ranking selection-based particle swarm optimizer for engineering design optimization problems. *Structural and Multidisciplinary Optimization* 37 (2): 131–147.