International Journal of Computational Intelligence Systems
Vol. **12(2)**; 2019, *pp.* 530–543
DOI: https://doi.org/10.2991/ijcis.d.190402.001; ISSN: 1875-6891; eISSN: 1875-6883
https://www.atlantis-press.com/journals/ijcis/

An Efficient Modified Particle Swarm Optimization Algorithm for Solving Mixed-Integer Nonlinear Programming Problems

Ying Sun[1], Yuelin Gao[2,*]

[1]*School of Computer Science and Information Engineering, Hefei University of Technology, No.193 Tunxi Road, Hefei, 230009, PR China*
[2]*Ningxia Province Key Laboratory of Intelligent Information and Data Processing, North Minzu University, No.204 Wenchang North Street, Yinchuan, 750021, PR China*

ARTICLE INFO

Article History
Received 05 Aug 2018
Accepted 28 Mar 2019

Keywords
Particle swarm optimization
Mixed-integer nonlinear programming
Constrained optimization
Simulated annealing

ABSTRACT

This paper presents an efficient modified particle swarm optimization (EMPSO) algorithm for solving mixed-integer nonlinear programming problems. In the proposed algorithm, a new evolutionary strategies for the discrete variables is introduced, which can solve the problem that the evolutionary strategy of the classical particle swarm optimization algorithm is invalid for the discrete variables. An update strategy under the constraints is proposed to update the optimal position, which effectively utilizes the available information on infeasible solutions to guide particle search. In order to evaluate and analyze the performance of EMPSO, two hybrid particle swarm optimization algorithms with different strategies are also given. The simulation results indicate that, in terms of robustness and convergence speed, EMPSO is better than the other algorithms in solving 14 test problems. A new performance index (NPI) is introduced to fairly compare the other two algorithms, and in most cases the values of the NPI obtained by EMPSO were superior to the other algorithms.

1. INTRODUCTION

In the fields of science and engineering, researchers have a great interest in solving optimization problems consisting of real and discrete (or integer) variables. These are called mixed-integer programming (MIP) problems, which arise from a variety of real-world situations and applications. For example, Alavidoost et al. [1] described a supply-chain network with an MIP problem and solved it by using a genetic algorithm. Willis et al. [2] used an MIP to search in the superstructure of chemical reaction networks. Cobuloglu et al. [3] presented a two-stage stochastic MIP model to maximize the economic and environmental benefits of food and biofuel production. Merchan et al. [4] studied discrete-time MIP models to solve the production scheduling problem. Ku et al. [5] used an MIP to solve job shop scheduling. Benati et al. [6] established an MIP model to solve the feature selection problem in clustering and a heuristic method was given. There are many other studies on MIPs [7–10].

An MIP problem is described as follows:

$$\begin{aligned} &\min\ f(x,y) \\ &\text{s.t.}\quad g_i(x,y) \le 0, i = 1,2,\cdots,n \\ &\qquad \underline{x} \le x \le \overline{x} \\ &\qquad \underline{y} \le y \le \overline{y}, \end{aligned} \tag{1}$$

where $x \in \mathbb{R}^l$ represents the continuous variables, $y \in \mathbb{Z}^m$ represents the discrete or integer variables, and $(\underline{x}, \underline{y})$ and $(\overline{x}, \overline{y})$ are the lower and upper bounds of the corresponding decision vectors, respectively. The objective function is f: $\mathbb{R}^l \times \mathbb{Z}^m \to \mathbb{R}$. The constraint functions are g_i: $\mathbb{R}^l \times \mathbb{Z}^m \to \mathbb{R}$. The feasible region is defined as:

$$\begin{aligned} FR = \{(x,y) \in \mathbb{R}^l \times \mathbb{Z}^m | g_i(x,y) \le 0,\ i = 1,2,\cdots,n; \\ \underline{x} \le x \le \overline{x};\ \underline{y} \le y \le \overline{y}\}. \end{aligned}$$

If any of $f(x,y)$ or $g_i(x,y)$ are nonlinear functions, then the problem is called mixed-integer nonlinear programming (MINLP). When $x \in \{0,1\}$, it is called 0–1 mixed-integer nonlinear programming (0–1 MINLP). If none of the decision variables are continuous variables, the problem is called integer programming (IP).

Both integer and MIP are Non-deterministic Polynomial-hard, and the difficulty to solve these problems increases exponentially with size [11]. The traditional optimization algorithms can only deal with simple optimization problems, and cannot be used to solve complex problems. Therefore, a large number of stochastic algorithms are proposed and applied to solve such problems.

Current studies on mixed-integer optimization problems typically focus on some special models, such as the convex model, 0–1 MIP, and so on. At present, research mainly concentrates on the following four approaches: (1) The integer continuity method [12]. In this approach, the continuous optimization algorithm is first used to calculate the optimal solution without integer constraints. Then, the integer variables are rounded into the feasible domain. So, the approximate optimal solution is found. This method can easily solve the continuous optimization problem; however, this has low precision in the process of converting continuous variables into integer

*Corresponding author. Email: gaoyuelin@263.net

variables, and perhaps may not find the optimal solution for some problems. (2) The deterministic method, which uses the branch boundary method [13] and the cutting-plane approach [14] to solve MIP problems. These methods are effective and efficient for solving small-scale problems but, when the scale of the problem increases and the degree of nonlinearity becomes higher, the deterministic method is usually incapable of solving the problem. (3) Monte Carlo methods [15] comprise a broad class of computational algorithms that rely on repeated random sampling to obtain an approximate solution of an MIP problem. From a theoretical point of view, a Monte Carlo method can find a better solution with enough sampling, but what constitutes "enough" is not easy to measure in actual operation. (4) Evolutionary algorithms have wide applications and many new examples of them have been used to solve MIP problems, including random searching algorithms [16], simulated annealing algorithms [17], and genetic algorithms [18], among others. Hybrid algorithms, which combine these algorithms with a penalty function, are especially useful in solving constrained optimization problems [19, 20], but an exact penalty function is difficult to build. As the decision variables include discrete (integer) variables and continuous variables, a coevolution strategy using multiple algorithms has also been applied for this problem [21, 22].

In this paper, the initial population is randomly generated by mixed-integer and real number coding methods. For the part comprised of the continuous variables, the traditional adaptive particle swarm optimization (PSO) algorithm is used to update the speed and position. The traditional update strategy of the PSO algorithm, proposed for the continuous optimization problem, cannot solve the discrete optimization problem very well. In order to solve this, a new discrete evolution strategy (DS) is adopted to update the position for the part comprised of the discrete or integer variables, such that the integer and continuous parts can be simultaneously evolved.

The constrained optimization problem cannot be solved by the standard PSO algorithm. In order to enable the PSO algorithm to solve such problems, many improved mechanisms has been proposed [23, 24]. Among them, one of the most common ways is by transforming the constrained optimization problem into an unconstrained optimization problem by using a penalty function method, and then solving it by a standard PSO algorithm [25]. Another approach is the feasible solution priority method (the Deb method) [26, 27], which enforces the following criteria: (1) any feasible solution is preferred to any infeasible solution, (2) among two feasible solutions, the one having a better objective function value is preferred, (3) among two infeasible solutions, the one having a smaller constraint violation is preferred. However, it is often not the most effective to restrict the infeasible solution emergence in the individual optimal solution and the global optimal solution. In this way, the available information on infeasible solutions cannot be used, such as the infeasible solution with smaller objective function values or smaller constraint violations. These infeasible solutions may be closer to the global best solution than some of the feasible solutions, so it is necessary to allow some infeasible solutions to be used as guiding positions under a certain probability. Thus, a new update strategy of individual and global best position based on the constraints (IDeb) is proposed in this paper.

The performance of the EMOPSO algorithm is tested with 14 benchmark functions. The experimental results indicate our algorithm outperforms other two hybrid PSO algorithms, and improves the computational accuracy under the acceptable computational time and space occupancy.

The rest of the paper is organized as follows: In Section 2, we present the efficient modified PSO algorithm (EMPSO). Thereafter, the results are provided in Section 3. The paper concludes with Section 4.

2. AN EMPSO ALGORITHM

2.1. PSO Algorithm

The PSO [28] algorithm, first introduced by Eberhart and Kennedy in 1995, is a population-based evolutionary optimization method, inspired by the social behavior of flocking birds. Similar to a genetic algorithm, it is an optimization technique based on iterative steps. The system is initialized with a population of random solutions and then searches for optima by updating generations. The swarm consists of N particles, where each particle is regarded as a point in the n-dimensional search space. Each particle has its own position and velocity, which are denoted as $x = (x_1, x_2, \cdots, x_n)$ and $v = (v_1, v_2, \cdots, v_n)$, respectively. Different position vectors x correspond to different fitness function values, related to the objective function value.

Algorithm 1: Adaptive Particle Swarm Optimization Algorithm (APSO)

- Step 1: Initialize a population of particles X_N, each with a random position vector x_i and velocity vector v_i. Set the self-cognitive coefficient c_1 and the social cognitive coefficient c_2, the maximum generation T_{max}, and the generation number $t := 1$.
- Step 2: Calculate the fitness of all particles in $X(t)$.
- Step 3: Update the position and velocities of the particles, based on the equations.

$$v_{id}^{t+1} = wv_{id}^{t} + c_1 r_1 \left(p_{i_{best}d} - x_{id}^{t}\right) + c_2 r_2 \left(g_{bestd} - x_{id}^{t}\right) \quad (2)$$

$$x_{id}^{t+1} = x_{id}^{t} + v_{id}^{t+1} \quad (3)$$

$$w = w_{max} - \frac{t(w_{max} - w_{min})}{T_{max}}, \quad (4)$$

  where $d = (1, 2, \cdots, n)$, r_1, r_2 are the random numbers between 0 and 1, $p_{i_{best}}$ is the personal best position of the ith particle, g_{best} is the global best position, w_{max} and w_{min} are the maximum and minimum values of the inertia weight, respectively.
- Step 4: Calculate each particle's fitness and update every particle's optimal position and the global best position.
- Step 5: (Termination examination) If the termination criterion is satisfied, then output the global best position and its fitness value. Otherwise, let $t := t + 1$, and return to **Step 2.**

The adaptive PSO algorithm, with the strategy of a linearly decreasing inertia weight, was introduced based on the standard particle swarm algorithm and has good global search capabilities.

2.2. Settings of the Initial Population and Selection of the Optimal Positions

This article modifies the initialization approach of the PSO algorithm, where the particle swarm is initialized through a mixed coding of discrete integers and real numbers. Namely, both the l-dimensional real variables $x \in (\underline{x}, \overline{x})$ and the m-dimensional discrete (integer) variables $y \in (\underline{y}, \overline{y})$ are randomly produced by a computer. These initialized values make up an individual code (a chromosomal):

$$(x, y) = (x_1, x_2, \cdots, x_l, y_1, y_2, \cdots, y_m).$$

If the individual (x, y) satisfies the constraint conditions $g_i(x, y) \leq 0$, $i = 1, 2, \cdots, n$, then it is regarded as a solution of the problem in Eq. 1 and marked as "1." Otherwise, it is marked as "0."

In order to ensure the guidance of the global best position, the population must have at least one feasible solution during the initialization process. That is, the initial position of the first particle must be in the feasible region and is taken as the initial global best position. If it is not a feasible solution, the generation is repeated. The initial position of all other particles can be randomly generated and are taken as their personal best positions.

2.3. Evolutionary Strategies for the Discrete Variables of a Particle

In view of the continuity of flight in the feasible domain, the traditional PSO is applied to solve the continuous optimization problem. In order to deal with the discrete variables of the MINLP more effectively, an improved evolutionary strategy for discrete variables (DS) is proposed. The specific approach is described in the following:

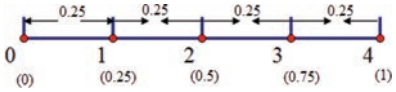
The discrete variables of the ith particle at the tth iteration are $y_i^t = [y_{i1}^t, y_{i2}^t, \cdots, y_{im}^t,]$, $y_{id}^t \in \Omega_d$, where Ω_d = {all possible discrete values that may be taken for the ith dimension} and $d = 1, 2, \cdots, m$. The update process for each discrete variable is discussed below:

1. Determine the number of all possible values for the dth dimension of the discrete (integer) variable as $|\Omega_d|$—for instance, if $1 \leq y_{id}^t \leq 4$, then $\Omega_d = \{1, 2, 3, 4\}$ and $|\Omega_d| = 4$.
2. Calculate the standard spacing for each possible value $\lambda_{id} = \frac{1}{|\Omega_d|}$.
3. As the individual and global optimal solutions both have a guiding role in finding the current solution, if the dth dimension of the global optimal solution equals the kth element of the set Ω_d, then adjust the spacing of $\Omega_d[k]$ to $\lambda'_{id}[k] = c_3 \times \lambda_{id}$, where c_3 is called the social cognitive coefficient (taken as c_3 = 1.5 here). For the ith particle, if the dth dimension of the individual optimal solution equals the lth element of the set Ω_d, then adjust the spacing of $\Omega_d[l]$ to $\lambda'_{id}[l] = c_4 \times \lambda_{id}$, where c_4 is called the self-cognitive coefficient (taken as c_4 = 1.2 here).

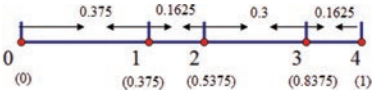
   After updating the spacings, we use the normalization method and rescale the other adjustment spacing values via

$$\lambda'_{id}[j] = \frac{\lambda'_{id}[j]}{\sum_{j=1}^{|\Omega_d|} \lambda'_{id}[j]}, \left(j = 1, 2, \cdots, |\Omega_d|, \text{ except } k \text{ and } l\right).$$

   For example,
   Standard spacing:



   Suppose that the global optimal solution is 1 and the individual optimal solution is 3, then the adjusted spacing is
   Adjusted spacing:



4. Next we follow the update strategy for each discrete variable: a random number uniformly distributed between 0 and 1 is generated by the computer. This number will fall into one of the intervals, the right value of this interval is defined as y_{id}^{t+1}.

   For the example case shown above, if the random number is 0.625, then it will fall in the interval [0.5375, 0.8375], and so $y_{id}^{t+1} = 3$.

   Repeating the above process m times, the new discrete variable vector y_i^{t+1} is obtained.

2.4. Update Strategy for the Optimal Position Based on the Constraints

2.4.1. Update strategy of the personal best position

When the personal best position update is performed, we compare the current position x_i^t with the personal best position $P_{i,best}$ and select the position that is more suitable for guiding as the new personal best position.

Let $f_i = f(x_i^t)$ and $G_i = G(x_i^t)$ express the function value and constraint violation degree of x_i^t, respectively. Likewise, let $f_{i,best} = f(P_{i,best})$ and $G_{i,best} = G(P_{i,best})$ be the function value and constraint violation degree of $P_{i,best}$, respectively.

For the two positions, there will be the following three situations in the comparison process (IDeb):

1. Both of the positions are feasible solutions $(G_i = G_{i,best} = 0)$. In this case, it means that the two positions are in the feasible region, so the one having a better objective function value is preferred and made to be the current personal best position. Namely, when $f_i < f_{i,best}$, let x_i^t replace $P_{i,best}$.

2. One position is a feasible solution, and the other one is an infeasible solution. In order to avoid losing the available information of the infeasible solution, we allow the infeasible solution to be accepted with a certain probability when the function value of the infeasible solution is less than the feasible solution in an early iteration. However, when the algorithm runs to the late stage, it is unacceptable that the infeasible solutions appear in the optimal position. So, the probability value of accepting infeasible solutions will be reduced to zero.

   The probability function for accepting an infeasible solutions is $Pr = Pr_2 - \frac{t(Pr_2 - Pr_1)}{T_{max}}$, where $Pr_1 = 0$ is the probability value for accepting infeasible solutions at the end, and $Pr_2 = 0.5$ is the probability value of accepting infeasible solutions at the beginning. Further, t is the number of the current iteration, and T_{max} is the maximum number of iterations. Obviously, the probability value gradually decreases from 0.5 to 0 as the iteration number increases.

3. Both of the two positions are infeasible solutions $\left(G_i > 0 \text{ and } G_{i,\,best} > 0\right)$. At this point, it is not reasonable to only consider the constraint violation degree and ignore the information of the function value. Knowing how to reasonably balance the relationship between the function value and the constraint violation degree is very important. In this case, similarly to the multi-objective constraint handling method, all of the constraint functions are treated as an objective function.

   When a dominant relationship exists, the solution will be compared by using a multi-objective constraint handling method. Namely, when the current position x_i^t dominates the personal best position $P_{i,\,best}$, let x_i^t replace $P_{i,\,best}$. Otherwise, don't update.

   If $f_i > f_{i,\,best}$ and $G_i < G_{i,\,best}$ or $f_i < f_{i,\,best}$ and $G_i > G_{i,\,best}$, a dominant relationship does not exist. Thus, in order to better measure the relationship between the function value and the constraint violation degree, we define two new parameters. Consider case 1, as an example:

   i. Constraint violation multiple: $Gmul_i = \frac{G_{i,\,best}}{G_i}$.

   ii. Function value optimization multiple: $fmul_i = \frac{f_i}{f_{i,\,best}}$.

In case 1, the function value of $P_{i,\,best}$ is smaller than the function value of x_i^t, but the constraint violation degree of x_i^t is smaller than the constraint violation degree of $P_{i,\,best}$, so how do we know which one is the best solution? We can compare them by using the constraint violation multiple and function value optimization multiple. When $Gmul_i \le fmul_i$, it can be shown that the function value optimization degree of $P_{i,\,best}$ is better than the constraint violation degree of x_i^t. Therefore, it is not necessary to update the personal best position. When $Gmul_i > fmul_i$, it is exactly the opposite case, and we let x_i^t replace $P_{i,\,best}$.

In case 2, the same conclusion can also be obtained.

By merging the above three cases, the pseudocode of the update strategy for the personal best position is as follows:

```
————————————————IDeb————————————————
1) Input:
      x_i^t: the current position;
      P_{i,best}: the personal best position;
      Pr: the probability value of accepting infeasible solutions;
2) If G_i = G_{i,best} = 0 (Both positions are feasible solutions.)
3)       If f_i < f_{i,best}
4)          P_{i,best} ← x_i^t;
5)       End If
6) Elseif G_i > 0, G_{i,best} = 0 (One position is a feasible solution, and
   the other one is an infeasible solution.)
7)       If f_i < f_{i,best}
8)          Randomly generate r ∈ [0,1];
9)          If r < Pr
10)            P_{i,best} ← x_i^t;
11)         End If
12)      End If
13) Elseif G_i = 0, G_{i,best} > 0
14)      If f_i < f_{i,best}
15)         P_{i,best} ← x_i^t;
16)      Else
17)         Randomly generate r ∈ [0,1];
18)         If r > Pr
19)            P_{i,best} ← x_i^t;
20)         End if
21)      End if
22) Elseif G_i > 0, G_{i,best} > 0 (Both positions are infeasible
    solutions.)
23)      Calculate fmul_i and Gmul_i
24)      If f_i < f_{i,best} & G_i < G_{i,best}
25)         P_{i,best} ← x_i^t;
26)      Elseif f_i > f_{i,best} & G_i < G_{i,best}
27)         If Gmul_i > fmul_i
28)            P_{i,best} ← x_i^t;
29)         End If
30)      Elseif f_i < f_{i,best} & G_i > G_{i,best}
31)         If Gmul_i < fmul_i
32)            P_{i,best} ← x_i^t;
33)         End If
34)      End If
35) End If
```

The pseudocode of the update strategy for the personal best position.

2.4.2. Update strategy for the global best position

In the PSO algorithm, the guiding role of the global best position is considered as the social cognition of the population in the optimization process. Here, when the best optimal solution in the feasible solutions for the current population is better than the existing optimal solution, the global best position is updated. If there is no feasible solution in this generation, the global best position is not updated.

2.5. Description of the EMPSO

As previously analyzed, the details of EMPSO can be summarized as follows and the flowchart of EMPSO is given in Figure 1.

- Step 1: Initialize the position and velocity of each particle randomly in the search space:
  - (a) Set the parameters c_1, c_2, c_3, c_4, w_{max}, w_{min}, Pr_1, Pr_2, the maximum generation T_{max}, and the population size N. Also set $t = 0$.
  - (b) Randomly initialize the positions of the population of particles by integer and continuous hybrid coding techniques and the velocities of the population of particles by continuous coding techniques.
- Step 2: Calculate the objective function values from the initial positions of the particles, set the personal best position $P_{i,best}$ of each particle equal to its current position, and set the global best position g_{best}.
- Step 3: Update the continuous variables using Equations (2) and (3), and update the discrete variables by using the new evolutionary strategies described in Section 2.3.
- Step 4: Update the personal best position $P_{i,best}$, for $i = (1, 2, \cdots, m)$, and the global best position g_{best}, by using the update strategies described in Section 2.4.
- Step 5: If the termination criterion is satisfied, then output g_{best} and its fitness value. If not, set $t := t + 1$ and go to **Step 2**.

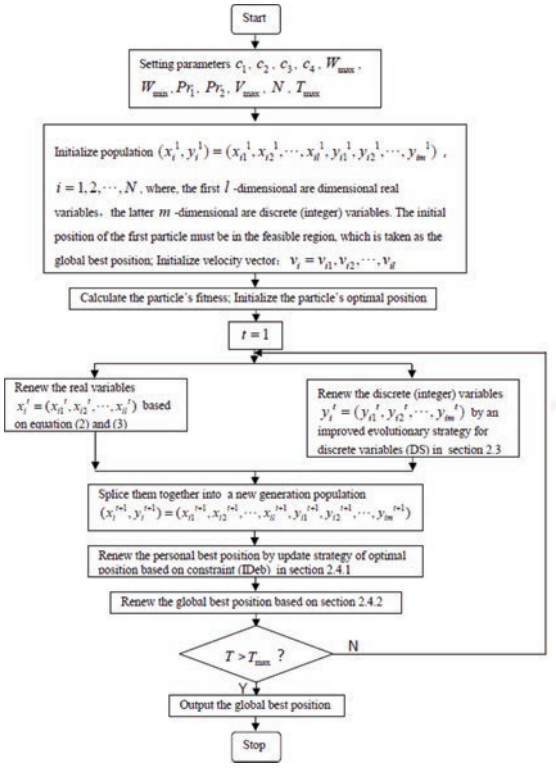


Figure 1 Flowchart of the efficient modified particle swarm optimization (EMPSO) algorithm.

3. NUMERICAL SIMULATIONS

3.1. Test Problems and Experimental Setup

To demonstrate the effectiveness and performance of the EMPSO, 14 test problems selected from published literature, were solved. The salient features of the test problems are shown in Table 1. For the sake of completeness, a summary of these benchmark problems is given in the Appendix A.

Table 1 Salient features of the test problems.

Problem	Total Number of Variables	Number of Integer Variables	Number of Constraints Excluding Bounds on Variables		References
			Inequality	Equality	
1	2	1	2	–	[2, 13]
2	2	1	1	–	[2, 13]
3	3	1	3	–	[2, 13]
4	5	3	3	2	[2]
5	7	4	9	–	[2]
6	5	2	3	–	[2]
7	2	1	2	–	[13]
8	3	2	–	–	[13]
9	3	1	2	–	[2]
10	2	2	2	–	[2]
11	3	3	2	–	[13]
12	5	5	6	–	[13]
13	7	7	7	–	[13]
14	4	4	3	–	[13]

In order to better study the effectiveness of the two improvement strategies (DS and IDeb) of the EMPSO algorithm, the classical evolutionary strategies for discrete variables, INT (update the discrete variables by a continuous evolution strategy, and then round), and Deb (an update strategy based on feasible solution priority, as previously described), were also studied. The experiments compare the EMPSO algorithm with other hybrid PSO algorithms: The INT-Deb-PSO algorithm and the DS-Deb-PSO algorithm. In the strategies mentioned above, common parameters to the three algorithms were set as follows: the self-cognitive coefficient for the continuous variables $c_1 = 1.7$, the social cognitive coefficient for the continuous variables $c_2 = 1.7$, the maximum inertia weight $w_{max} = 0.9$, the minimum inertia weight $w_{min} = 0.5$, and the maximum generation $T_{max} = 1000$. In the EMPSO algorithm, the social cognitive coefficient for the discrete variables was set to $c_3 = 1.5$, the self-cognitive coefficient for the discrete variables to $c_4 = 1.2$, the minimum probability of accepting an infeasible solutions $Pr_1 = 0$, and the maximum probability of accepting an infeasible solutions to $Pr_2 = 0.5$. All programs were coded in Matlab and all executions were made on a Tsinghua Tongfang personal computer with an Intel(R) Core(TM) i5-3570K CPU@3.40 GHz.

3.2. Experimental Results

The 14 test problems were executed in 50 independent runs. Experimental results obtained by the three algorithms are given in Table 2, where "Fun." indicates the problem, "Best" stands for the best optimal value obtained by the algorithm in all runs, "Worst"

Table 2 Results obtained by the three algorithms on 14 benchmark problems.

Fun.	Algorithm	Best	Worst	Median	Mean	Std	Success Rate	Time	Avg. Fun. Eval.
	EMPSO	2.000003	2.000100	2.000047	2.000051	9.54E-10	1	0.09	3307
1	DS-Deb-PSO	2.000005	2.000100	2.000045	2.000050	6.36E-10	1	0.14	5315
	INT-Deb-PSO	2.000001	2.236068	2.000048	2.009489	2.18E-03	0.96	0.04	5189
	EMPSO	2.124468	2.124499	2.124486	2.124484	1.01E-10	1	0.07	2311
2	DS-Deb-PSO	2.124346	2.124499	2.124484	2.124479	5.46E-10	1	0.09	3448
	INT-Deb-PSO	2.124349	2.124499	2.124477	2.124452	2.94E-09	1	0.03	2665
	EMPSO	1.076544	1.076839	1.076550	1.076556	1.67E-09	1	0.17	6406
3	DS-Deb-PSO	1.076544	1.250000	1.076551	1.102258	3.63E-03	0.84	0.24	9151
	INT-Deb-PSO	1.076545	1.350507	1.076551	1.118517	5.89E-03	0.74	0.07	6798
	EMPSO	7.667180	7.667180	7.667180	7.667180	7.24E-30	1	0.00	114
4	DS-Deb-PSO	7.666911	8.740312	7.667180	7.737174	5.46E-10	0.92	0.17	3135
	INT-Deb-PSO	7.667180	8.740215	7.667180	7.803010	1.05E-01	0.84	0.00	200
	EMPSO	4.579613	5.632730	4.579679	4.600734	2.22E-02	0.98	11.54	150992
5	DS-Deb-PSO	4.579632	5.632730	4.579685	4.642862	6.38E-02	0.94	8.50	111262
	INT-Deb-PSO	4.579641	5.780000	4.579695	4.940220	2.84E-01	0.68	1.40	124306
	EMPSO	−32217.427471	−32217.426018	−32217.426489	−32217.426579	1.47E-07	1	2.73	66672
6	DS-Deb-PSO	−32217.427739	−32215.812744	−32217.420663	−32217.320323	9.54E-02	0.8	6.66	159860
	INT-Deb-PSO	−32217.427353	−28674.167359	−32217.426388	−32146.470479	2.51E + 05	0.94	0.08	8219
	EMPSO	−4242.004672	−4242.003706	−4242.004106	−4242.004124	8.14E-08	1	0.62	22022
7	DS-Deb-PSO	−4242.004372	−4241.997047	−4242.003737	−4242.002696	3.91E-06	0.62	0.78	28168
	INT-Deb-PSO	−7189.179772	−1142.244867	−4242.004209	−3737.029756	2.42E + 06	0.72	0.07	7256
	EMPSO	0.000000	0.000001	0.000001	0.000001	7.05E-14	1	1.32	26314
8	DS-Deb-PSO	0.000000	0.000001	0.000000	0.000000	8.66E-14	1	1.40	27814
	INT-Deb-PSO	0.000000	0.000001	0.000001	0.000001	8.22E-14	1	0.17	8350
	EMPSO	−75.134137	−75.134000	−75.134023	−75.134039	1.36E-09	1	0.60	26654
9	DS-Deb-PSO	−75.134150	−75.134000	−75.134033	−75.134043	1.40E-09	1	0.63	27948
	INT-Deb-PSO	−75.134118	15.532925	5.385932	−28.057187	1.91E + 03	0.44	0.23	24141
	EMPSO	−42.632121	−42.632121	−42.632121	−42.632121	5.15E-29	1	0.00	60
10	DS-Deb-PSO	−42.632121	−42.632121	−42.632121	−42.632121	5.15E-29	1	0.00	67
	INT-Deb-PSO	−42.632121	−42.632121	−42.632121	−42.632121	5.15E-29	1	0.00	85
	EMPSO	−68.000000	−68.000000	−68.000000	−68.000000	0	1	0.02	436
11	DS-Deb-PSO	−68.000000	−68.000000	−68.000000	−68.000000	0	1	0.03	593
	INT-Deb-PSO	−90.000000	16.000000	−68.000000	−65.840000	1.59E + 02	0.84	0.01	744
	EMPSO	8.000000	8.000000	8.000000	8.000000	0	1	0.01	119
12	DS-Deb-PSO	8.000000	8.000000	8.000000	8.000000	0	1	0.04	440
	INT-Deb-PSO	8.000000	14.000000	8.000000	8.120000	7.20E-01	0.98	0.00	86
	EMPSO	14.000000	14.000000	14.000000	14.000000	0	1	0.19	1680
13	DS-Deb-PSO	14.000000	14.000000	14.000000	14.000000	0	1	0.22	2023
	INT-Deb-PSO	14.000000	29.000000	14.000000	17.060000	2.12E + 01	0.56	0.04	4155
	EMPSO	−0.974565	−0.974565	−0.974565	−0.974565	4.53E-31	1	0.02	254
14	DS-Deb-PSO	−0.974565	−0.974565	−0.974565	−0.974565	4.53E-31	1	0.03	391
	INT-Deb-PSO	−0.974565	0.000000	−0.974565	−0.948732	2.08E-02	0.96	0.01	672

EMPSO, efficient modified particle swarm optimization.

is the worst optimal value found by the algorithm in all runs, "Median" is the middle value of all obtained optimal values through the algorithm, "Mean" denotes the mean value of all obtained optimal values for the algorithm, "Std" is the standard deviation of all obtained optimal values through the algorithm, and "Success rate" represents the percentage of successful runs to the total runs for the algorithm (a successful run is where the obtained objective function value is within 0.1% of the known optimal value). Furthermore, "time" represents the average computation time (in seconds) used by the algorithm (in the case of successful runs), and "Avg. fun. eval." is average number of objective function evaluations used by the algorithm in successful runs. The success rates of the three algorithms are shown in Figure 2.

In order to reflect the optimization process, the optimizing curves of the three algorithms are plotted, in Figures 3–16, for the 14 test problems. For each image combination, the left image is the entire optimization process for the problem. As the main optimization process of some functions only takes place in a short period of time, the convergence process is not obvious in the left image. Therefore, in the right image, the part that converges fast for each function is locally enlarged.

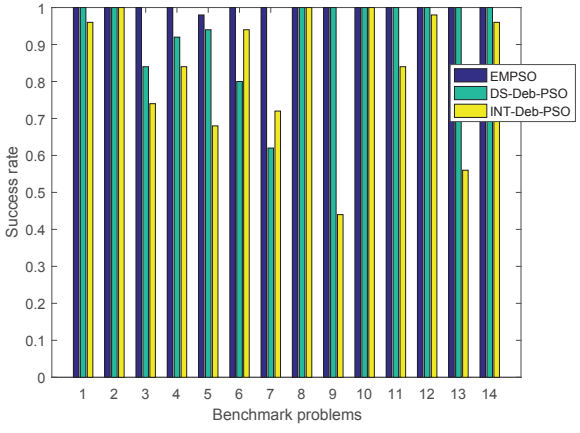


Figure 2 The success rates of three algorithms for 14 benchmark problems.

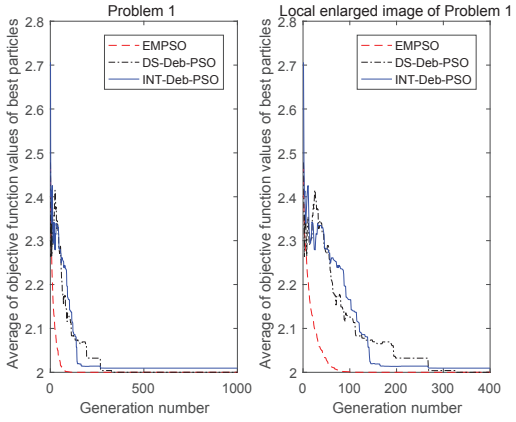


Figure 3 | The entire (left) and locally enlarged (right) optimization curves for test problem 1.

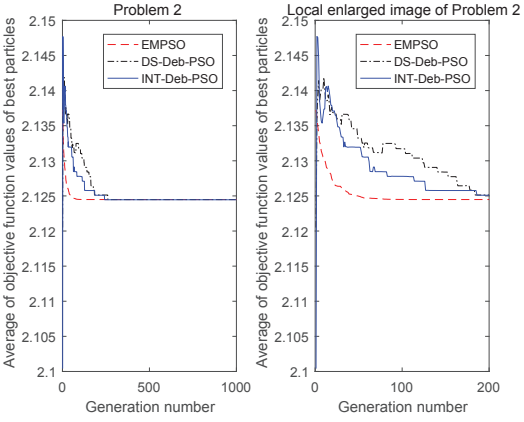


Figure 4 | The entire (left) and locally enlarged (right) optimization curves for test problem 2.

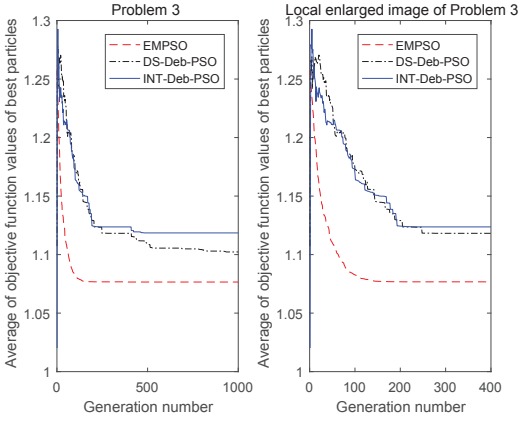


Figure 5 | The entire (left) and locally enlarged (right) optimization curves for test problem 3.

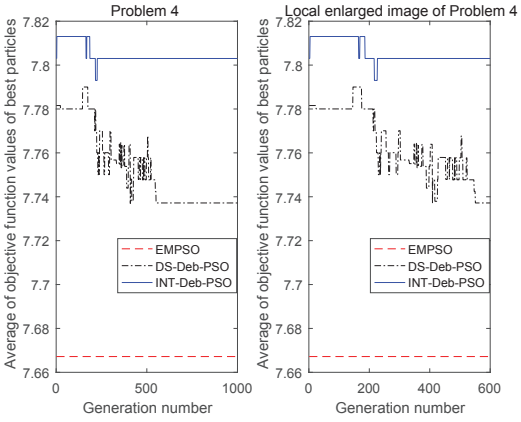


Figure 6 | The entire (left) and locally enlarged (right) optimization curves for test problem 4.

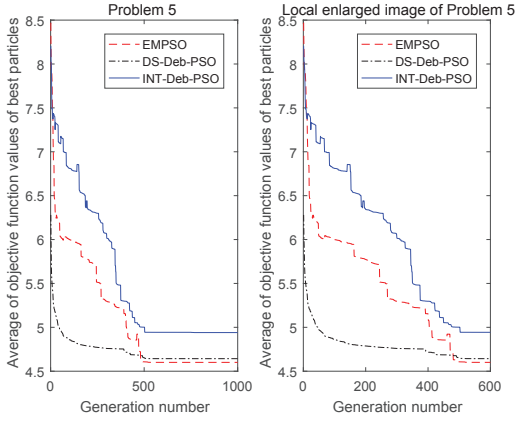


Figure 7 | The entire (left) and locally enlarged (right) optimization curves for test problem 5.

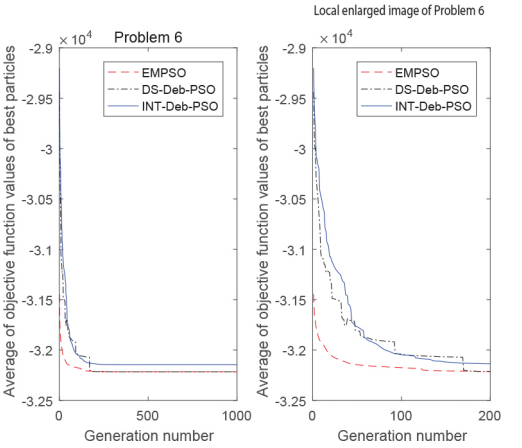


Figure 8 | The entire (left) and locally enlarged (right) optimization curves for test problem 6.

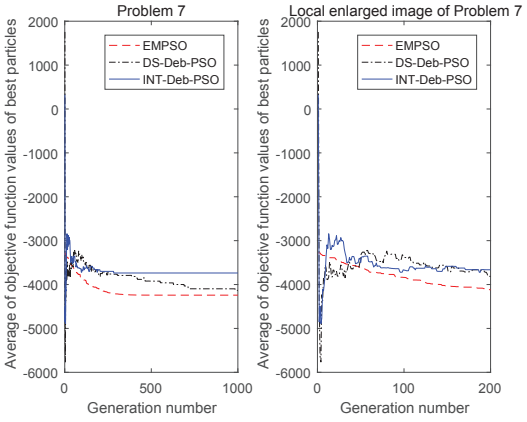


Figure 9 | The entire (left) and locally enlarged (right) optimization curves for test problem 7.

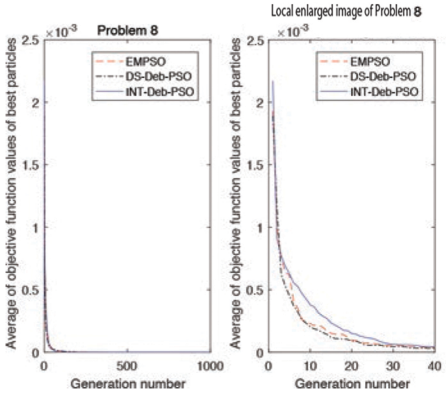


Figure 10 | The entire (left) and locally enlarged (right) optimization curves for test problem 8.

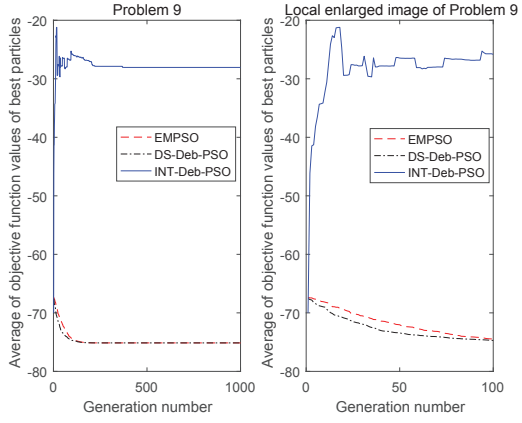


Figure 11 | The entire (left) and locally enlarged (right) optimization curves for test problem 9.

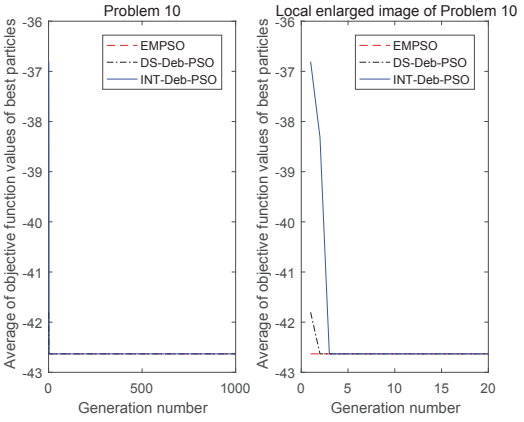


Figure 12 | The entire (left) and locally enlarged (right) optimization curves for test problem 10.

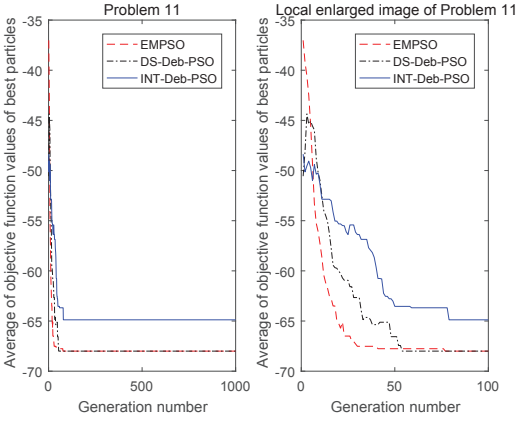


Figure 13 | The entire (left) and locally enlarged (right) optimization curves for test problem 11.

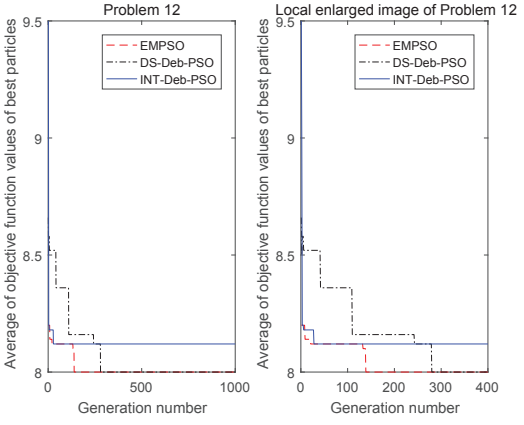


Figure 14 | The entire (left) and locally enlarged (right) optimization curves for test problem 12.

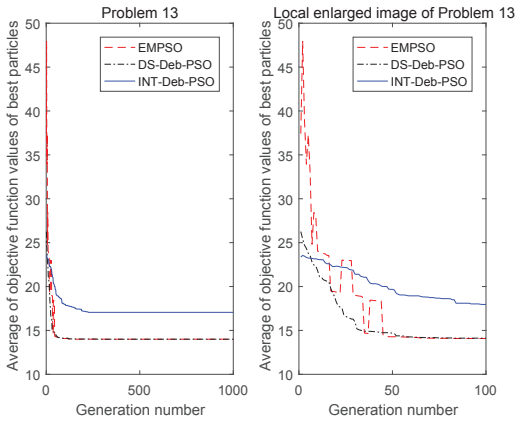


Figure 15 The entire (left) and locally enlarged (right) optimization curves for test problem 13.

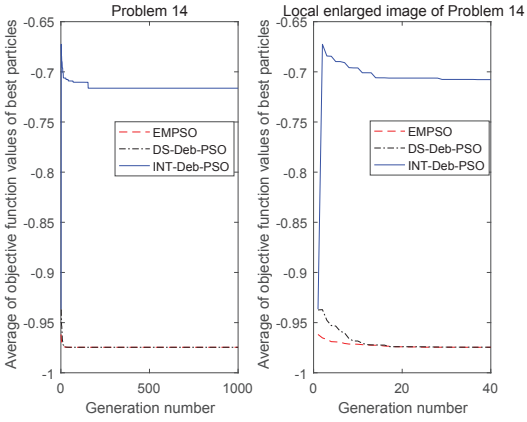


Figure 16 The entire (left) and locally enlarged (right) optimization curves for test problem 14.

3.3. Discussion of the Results

The results in Table 1 and Figure 2 show that EMPSO had the highest success rate in solving the 14 problems and provided a 100% success rate in 13 problems (i.e., all except for problem 5). Likewise, DS-Deb-PSO and INT-Deb-PSO provided a 100% success rate in 9 problems (1, 2, and 8–14) and 3 problems (2, 8, and 10), respectively. INT-Deb-PSO only had good performance in solving the problem with two continuous discrete variables, but for problem P_9, which had nonuniformly spaced integers, its success rate dropped to 44%. Therefore, a combination of the DS strategy and the IDeb strategy is much better than the combination of the INT strategy and the Deb strategy. Since the DS strategy is more complex than the INT strategy, the computation time of the INT-Deb-PSO is the shortest, except for problem P_4.

In terms of the average number of objective function evaluations, INT-Deb-PSO had the best value for five cases (6–9 and 12), whereas DS-Deb-PSO had the best value in only one case (5). In the other eight cases EMPSO obtained the best values. It can be seen that INT-Deb-PSO has obvious advantages in terms of computation time because of its simple strategy, but the advantage of the best average number of objective function evaluations is not as good as the algorithm based on the DS strategy.

EMPSO obtained the minimum standard deviation in all problems, and DS-Deb-PSO had a better standard deviation than INT-Deb-PSO for 13 problems (all except problem 8). The INT-Deb-PSO algorithm is not stable and fluctuates violently because of its lower success rate, and its standard deviation reached more than 10 for five problems (6, 7, 9, 11, and 13), it was as high as 10^6 for problem 7.

The known optimal solution could be obtained by EMPSO and DS-Deb-PSO, in our experiment, for all problems. For problems 7 and 11, INT-Deb-PSO seemed to have smaller experimental solutions than the known optimal solutions, but it could be easily verified that the two solutions were infeasible solutions. The EMPSO found the global optimal solutions within an acceptable error range (0.1%) for 13 problems and had an error of 0.02 for problem 5. The other algorithms were not 100% successful in solving all of the problems, as their error surpassed the allowed range. The algorithm with the DS strategy and the IDeb strategy, was not 100% successful in solving problems (such as problem 5), but its experimental solution was still near the known optimal solution and remained feasible. However, when other strategies are adopted, the deviation will increase, potentially generating infeasible experimental solutions.

In Figures 3–16, it appears that the EMPSO algorithm had a better convergence effect than the other algorithms, with its function curve dropping faster and smoother for all problems, except for problem 13. INT-Deb-PSO was not ideal in terms of the functional convergent curve of the 14 problems, having a slow convergence speed and poor stability. Furthermore, it could not converge to the optimal values for problems 6, 11, and 14. The convergence of DS-Deb-PSO was in between the other two algorithms. For problems 7, 11, and 13, the convergence speed of DS-Deb-PSO was faster than EMPSO at first, but it lagged behind the EMPSO algorithm before converging to the optimal value. The Deb strategy retained a feasible solution, so that the algorithm converged faster at an early stage. However, the Deb strategy ignored the guiding role of infeasible solutions, and so its ability to jump out of local optima was inferior to the IDeb strategy and the convergence effect was worse than the IDeb strategy at later stages.

Results from the above analysis show that each of the three algorithms had their own advantages in terms of time, success rate, average function evaluations, and average of the optimal solution. For INT-Deb-PSO, the computation time was obviously less than other algorithms, but the success rate was much lower than the other two algorithms. For test problem 5, the success rate of EMPSO was better than DS-Deb-PSO, but the computation time and average function evaluations of the EMPSO were inferior to DS-Deb-PSO.

3.4. Comparisons between the Three Algorithms with a New Performance Index

In order to get a better insight into the relative performance of computation time, success rate, and average function evaluations, the value of a performance index (PI) [18] is used. However, the PI cannot predict settlement in the stability of an algorithm and the distance between the calculated optimal value and the known optimal value, which are important in measuring the quality of an algorithm. In this paper, a new performance index (NPI) is proposed,

based on an old PI, which is a comprehensive index of computation time, success rate, average function evaluations, stability of the algorithm, and the distance between the calculated optimal value and the known optimal value. For the computational algorithms we compared, the value of the new performance index NPI_i for the algorithms is computed as

$$NPI_i = \frac{1}{N}\sum_{j=1}^{N}\left[k_1 SR_i^j + k_2\left(\frac{MT^j}{AT_i^j}\right) + k_3\left(\frac{MF^j}{AF_i^j}\right) + k_4\left(\frac{MD^j}{AD_i^j}\right) + k_5\left(\frac{MV^j}{AV_i^j}\right)\right],$$

where N is the total number of test problems, SR_i^j denotes the success rate obtained by the ith algorithm in solving the jth problem, AT_i^j is the average computation time used by the ith algorithm in successful runs for the jth problem, MT^j is the minimal average computation time used by the three algorithms in solving the jth problem, AF_i^j represents the average number of function evaluations used by the ith algorithm when solving the jth problem (in the case of successful runs), and MF^j is the minimum of the average function evaluations required by the different algorithms in solving the jth problem. Similarly, AD_i^j denotes the absolute value of the distance between the mean value and the known optimal value, obtained by the ith algorithm in successful runs for the jth problem, MD^j is the minimum of the absolute value of the distance between the mean value and the known optimal value obtained by the three algorithms in solving the jth problem, AV_i^j is the standard deviation (a measure of the stability of the algorithm) obtained by ith algorithm when solving the jth problem (in the case of successful runs), and MV^j is the minimum of the standard deviation obtained by the various algorithms in solving the jth problem. Further, k_1, k_2, k_3, k_4, and k_5 are nonnegative constants, such that $k_1 + \cdots + k_5 = 1$. These parameters reflect the importance of the each subindex. The larger the value of NPI, the better the performance of the algorithm. Usually when the value of NPI is calculated, the parameters are set according to the following five cases:

1. $k_1 = w, k_2 = k_3 = k_4 = k_5 = \dfrac{(1-w)}{4}$;
2. $k_2 = w, k_1 = k_3 = k_4 = k_5 = \dfrac{(1-w)}{4}$;
3. $k_3 = w, k_1 = k_2 = k_4 = k_5 = \dfrac{(1-w)}{4}$;
4. $k_4 = w, k_1 = k_2 = k_3 = k_5 = \dfrac{(1-w)}{4}$;
5. $k_5 = w, k_1 = k_2 = k_3 = k_4 = \dfrac{(1-w)}{4}$.

The graphs of NPI, corresponding to each of the five cases, are shown in Figures 17–21, respectively. It can be seen that the EMPSO algorithm outperformed the other two algorithms, except in Figure 18, in which the time was the main emphasis on the weights. In this case the evolutionary strategies for the discrete variables of the EMPSO algorithm, DS, need more time to calculate and are more complex than the approximate values (rounded-data) of INT-Deb-PSO. So, when $k_2 > 0.4$, the NPI of EMPSO was surpassed by INT-Deb-PSO. The DS-Deb-PSO did not perform well under the comprehensive index NPI, the main reason being that the algorithm uses the DS strategy, which led to calculation times longer than INT-Deb-PSO, and the Deb strategy leads to the a lower convergence of efficiency than the EMPSO. In addition to the evolutionary strategies for the discrete variables and the update strategy, the other parameters of the three algorithms are the same, so we can see that the EMPSO, with the evolutionary strategies of the discrete variables (DS) and an update strategy based on the constraints (IDeb), was much more effective than the other hybrid-PSO algorithms; those with approximate values (INT) or an update strategy based on feasible solution priority (Deb).

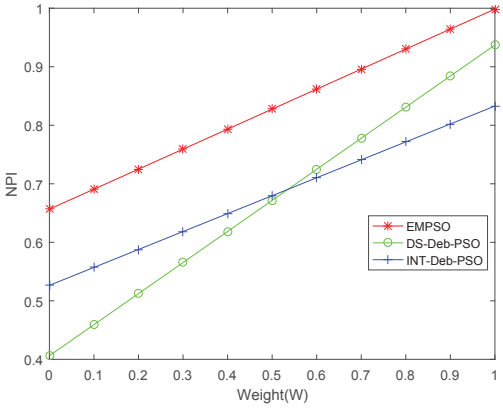


Figure 17 New performance indices (NPIs) for the efficient modified particle swarm optimization (EMPSO), DS-Deb-PSO, and INT-Deb-PSO algorithms when $k_1 = w, k_2 = k_3 = k_4 = k_5 = \dfrac{(1-w)}{4}$.

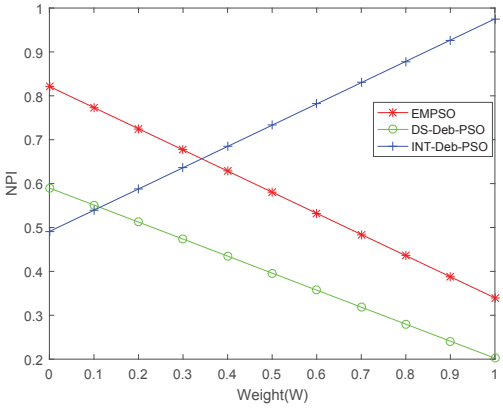


Figure 18 New performance indices (NPI) for the efficient modified particle swarm optimization (EMPSO), DS-Deb-PSO, and INT-Deb-PSO algorithms when $k_2 = w, k_1 = k_3 = k_4 = k_5 = \dfrac{(1-w)}{4}$.

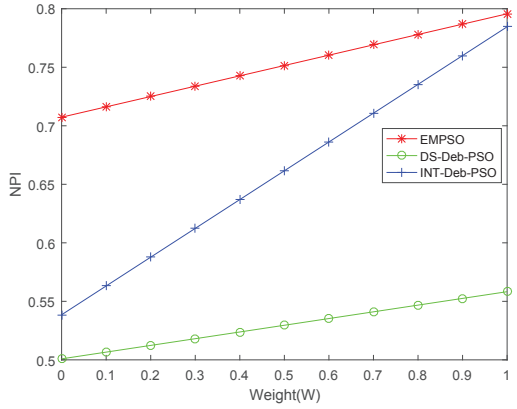


Figure 19 | New performance indices (NPI) for the efficient modified particle swarm optimization (EMPSO), DS-Deb-PSO, and INT-Deb-PSO algorithms when $k_3 = w$, $k_1 = k_2 = k_4 = k_5 = \frac{(1-w)}{4}$.

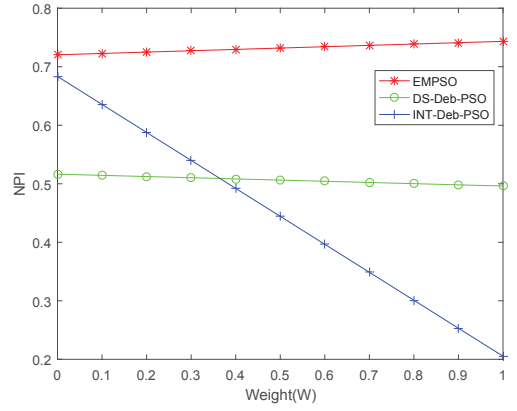


Figure 20 | New performance indices (NPI) for the efficient modified particle swarm optimization (EMPSO), DS-Deb-PSO, and INT-Deb-PSO algorithms when $k_4 = w$, $k_1 = k_2 = k_3 = k_5 = \frac{(1-w)}{4}$.

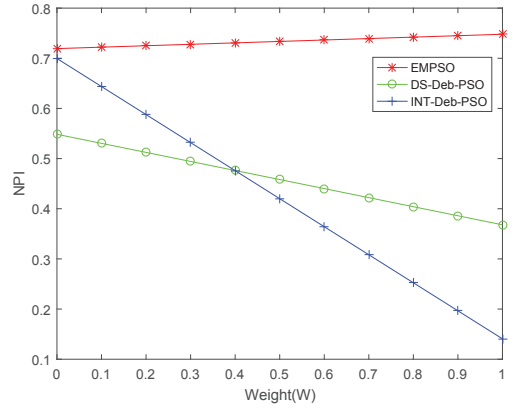


Figure 21 | New performance indices (NPI) for the efficient modified particle swarm optimization (EMPSO), DS-Deb-PSO, and INT-Deb-PSO algorithms when $k_5 = w$, $k_1 = k_2 = k_3 = k_4 = \frac{(1-w)}{4}$.

3.5. Comparison of the Computational Complexity of the Three Algorithms

In this section, in order to further compare the advantages and disadvantages of the three algorithms, the computational complexity of the proposed three algorithms is discussed. Since the main program of the three algorithms is the same, both the time and space complexities of the different improvement strategies in one generation are analyzed below.

3.5.1. Time complexity

1. *INT Strategy:*
   1. Updating the discrete variables by the continuous update strategy of the PSO (the position and velocity are updated) requires $O(N * m)$ time, where N indicates the population size and m indicates the dimension of the discrete variable.
   2. The rounding operation requires $O(N * m)$ time.
   Hence, the total time complexity of the INT strategy is $O(N * m)$.
2. *DS Strategy*:
   1. Calculating the standard spacing requires $O(m)$ time.
   2. Calculating the adjusted spacing requires $O(N * m)$ time.
   3. Determining the update interval requires $O(N * m)$ time.
   4. Updating the position vector requires $O(N * m)$ time.
   Hence, the total time complexity of the DS strategy is $O(N * m)$.
3. *Deb Strategy:*
   1. Calculating the constraint violation degree requires $O(N * m)$ time, where L indicates the number of constraint functions.
   2. Calculating the fitness values of the current position and the personal best position requires $O(N)$ time.
   3. Updating the personal best position requires $O(N)$ time.
   Hence, the total time complexity of the Deb strategy is $O(N * m)$.
4. *IDeb Strategy:*
   In the worst case,
   1. Calculating the constraint violation degree requires $O(N * m)$ time.
   2. Calculating the fitness values of the current position and the personal best position require $O(N)$ time.
   3. Calculating the constraint violation multiple requires $O(N)$ time.
   4. Calculating the function value optimization multiple requires $O(N)$ time.
   5. Updating the personal best position requires $O(N)$ time.
   Hence, the total time complexity of the IDeb strategy is $O(N * m)$.

In summary, we can see that the time complexities of the INT and DS strategies are on the same order. However, it is not difficult to

find that the former is easy to operate, and so its calculation frequency is smaller than the latter. Additionally, the time complexity of the Deb strategy is smaller than that of the IDeb strategy, but the gap is very small. Therefore, the computation time of INT-Deb-PSO is the smallest, compared with the other two algorithms, which can also be seen in Figure 18. This is the advantage of the INT-Deb-PSO algorithm, but its performance in other aspects is not good. It can be seen that the calculation effect of the EMPSO algorithm is better; while the computation time is longer than the others, they basically remain on the same order.

3.5.2. Space complexity

1. *INT Strategy:*
   As the update of the discrete variables needs to be overwritten in the calculation process, and no additional storage is needed. Hence, the space complexity is $O(N * m)$.
2. *DS Strategy:*
   1. The amount of space taken up by storing the spacing vector is $O\left(N * |\Omega_{max}| * m\right)$, where $|\Omega_{max}| = max\{|\Omega_d|, d = 1, 2, \cdots, m\}$.
   2. The update of the discrete variables needs to be overwritten in the calculation process, so the amount of space is $O(N * m)$.
      Hence, the total space complexity of the DS strategy is $O\left(N * |\Omega_{max}| * m\right)$.
3. *Deb Strategy:*
   1. The amounts of space taken up by the constraint violation degree of the current position and the personal best position are $O(N)$.
   2. The amounts of space taken up by fitness values of the current position and the personal best position are $O(N)$.
   3. The amounts of space taken up by the current position and the personal best position are $O(N * d)$, where d indicates the dimension of the particle.
      Hence, the total space complexity of the Deb strategy is $O(N * d)$.
4. *IDeb Strategy:*
   In the worst case,
   1. The amounts of space taken up by the constraint violation degree of the current position and the personal best position are $O(N)$.
   2. The amounts of space taken up by fitness values of the current position and the personal best position are $O(N)$.
   3. The amount of space taken up by the constraint violation multiple is $O(N)$.
   4. The amount of space taken up by the function value optimization multiple is $O(N)$.
   5. The amounts of space taken up by the current position and the personal best position are $O(N * d)$.
      Hence, the total space complexity of the IDeb strategy is $O(N * d)$.

We can see that the two improved strategies (DS and IDeb) take up a little more space than the two classical strategies (INT and Deb). However, for the current level of computer hardware, the impact of these additional storage requirements on the operation of the algorithm is basically negligible.

To summarize, the EMPSO algorithm not only improves the computational accuracy, but also achieves acceptable computational time and space occupancy.

4. CONCLUSIONS

This paper has presented an EMPSO for solving mixed integer nonlinear programming problems. For comparison, two hybrid PSO algorithms, the INT-Deb-PSO algorithm and the DS-Deb-PSO algorithm, were also implemented. All three algorithms have two evolutionary strategies: the evolutionary strategies of the discrete variables (DS or INT) and an update strategy of the optimal position (IDeb or Deb). For 14 integer and MIP problems, the numeric results show that EMPSO has a better robustness and faster convergence speed than the other two hybrid PSO algorithms. In order to fairly compare EMPSO with the other hybrid PSO algorithms, a comprehensive index, NPI, is proposed in this paper, which can weigh the importance of computation time, success rate, average number of function evaluations, stability of the algorithm, and the distance between the calculated optimal value and the known optimal value. The value of the NPI obtained by EMPSO is superior to the other algorithms in most cases, but the computational time of EMPSO needs to be further reduced. In this article, we only deal with the MIP problems with one objective function. In the near future, we also plan to deal with the MIP problems with multi-objective. Furthermore, most parameters in this paper (such as c_3, c_4, etc.) have certain values. It would be interesting to study whether these control parameters could adaptively change as the iteration time increases.

ACKNOWLEDGMENTS

This research was funded by the National Natural Science Foundation of P.R. China (61561001), First-Class Disciplines Foundation of NingXia (Grant No. NXYLXK2017B09) and Major Project of North Minzu University (2019MS003).

REFERENCES

[1] M.H. Alavidoost, M. Tarimoradi, M.H.F. Zarandi, Bi-objective mixed-integer nonlinear programming for multi-commodity tri-echelon supply chain networks, J. Intell. Manuf. 29(4) (2018), 809–826.
[2] M.J. Willis, M.V. Stosch, Inference of chemical reaction networks using mixed integer linear programming, Comput. Chem. Eng. 90 (2016), 31–43.
[3] H.I. Cobuloglu, İ. Esra Büyüktahtakın, A two-stage stochastic mixed-integer programming approach to the competition of biofuel and food production, Comput. Ind. Eng. 107 (2017), 251–263.

[4] F. Merchan, H. Lee, C.T. Maravelias, Discrete-time mixed-integer programming models and solution methods for production scheduling in multistage facilities, Comput. Chem. Eng. 94 (2016), 387–410.
[5] W.Y. Ku, J.C. Beck, Mixed integer programming models for job shop scheduling: a computational analysis, Comput. Oper. Res. 73 (2016), 165–173.
[6] S. Benati, S. García, J. Puerto, Mixed integer linear programming and heuristic methods for feature selection in clustering, J. Oper. Res. Soc. 69(9) (2018), 1379–1395.
[7] T.W. Liao, Two hybrid differential evolution algorithms for engineering design optimization, Appl. Soft Comput. 10(4) (2010), 1188–1199.
[8] J. Alemany, L. Kasprzyk, F. Magnago, Effects of binary variables in mixed integer linear programming based unit commitment in large-scale electricity markets, Electr. Power Syst. Res. 160 (2018), 429–438.
[9] T.A. Albahri, C.S. Khor, M. Elsholkami, A. Elkamel, Optimal design of petroleum refinery configuration using a model-based mixed-integer programming approach with practical approximation, Ind. Eng. Chem. Res. 57(22) (2018), 7555–7565.
[10] S. Vadlamani, D. Schweitzer, H. Medal, A. Nandi, B. Eksioglu, A mixed-integer programming approach for locating jamming devices in a flow-jamming attack, Comput. Oper. Res. 95(4) (2018), 83–96.
[11] F. Matteo, L. Andrea, Heuristics in mixed integer programming, in: J.J. Cochran, L.A. Cox Jr., P. Keskinocak, J.P. Kharoufeh, J.C. Smith (Eds.), Wiley Encyclopedia of Operations Research and Management Science, John Wiley and Sons, Inc., Hoboken, NJ, 2011.
[12] L.D.S. Coelho, An efficient particle swarm approach for mixed-integer programming in reliability-redundancy optimization applications, Reliab. Eng. Syst. Safety. 94(4) (2009), 830–837.
[13] V.P. Eronen, M.M. Mäkelä, T. Westerlund, Extended cutting plane method for a class of nonsmooth nonconvex MINLP problems, Optimization. 64(3) (2015), 641–661.
[14] J. Barnett, J.P. Watson, D.L. Woodruff, BBPH: using progressive hedging within branch and bound to solve multi-stage stochastic mixed integer programs, Oper. Res. Lett. 45(1) (2017), 34–39.
[15] P. Li, B. Chen, B. Zhang, L. Jing, Monte Carlo simulation-based dynamic mixed integer nonlinear programming for supporting oil recovery and devices allocation during offshore oil spill responses, Ocean Coast. Manag. 89(2) (2014), 58–70.
[16] R.L. Salcedo, Solving nonconvex nonlinear programming and mixed-integer nonlinear programming problems with adaptive random search, Ind. Eng. Chem. Res. 31(1) (1992), 262–273.
[17] Ž.N. Popovic, V.D. Kerleta, D.S. Popovic, Hybrid simulated annealing and mixed integer linear programming algorithm for optimal planning of radial distribution networks with distributed generation, Electr. Power Syst. Res. 108(108) (2014), 211–222.
[18] K. Deep, K.P. Singh, M.L. Kansal, C. Mohan, A real coded genetic algorithm for solving integer and mixed integer optimization problems, Appl. Math. Comput. 212(2) (2009), 505–518.
[19] Y.X. Li, M. Gen, Nonlinear mixed integer programming problems using genetic algorithm and penalty function, in Proceeding of 1996 IEEE International Conference on Systems, Man, and Cybernetics, Beijing, 1996, pp. 2677–2682.
[20] M.F.P. Costa, A.M.A.C. Rocha, R.B. Francisco, M.G.P. Fernandes, Firefly penalty-based algorithm for bound constrained mixed-integer nonlinear programming, Optimization. 65(5) (2016), 1085–1104.
[21] L. Sahoo, A. Banerjee, A.K. Bhunia, S. Chattopadhyay, An efficient GA-PSO approach for solving mixed-integer nonlinear programming problem in reliability optimization, Swarm Evol. Comput. 19 (2014), 43–51.
[22] Y.L. Gao, Y. Sun, J. Wu, Difference-genetic co-evolutionary algorithm for nonlinear mixed integer programming problems, J. Nonlin. Sci. Appl. 9 (2016), 1261–1284.
[23] H. Gandomi, A.R. Kashani, Evolutionary bound constraint handling for particle swarm optimization, in Proceeding 4rd International Conference Computational and Business Intelligence, Olten, 2016, pp. 148–152.
[24] C.R. Carvalho, H.S. Bernardino, P.H. Hallak, A.C.C. Lemonge, An adaptive penalty scheme to solve constrained structural optimization problems by a craziness based particle swarm optimization, Optim. Eng. 18(3) (2017), 693–722.
[25] L.A. Vardhan, A. Vasan, Evaluation of penalty function methods for constrained optimization using particle swarm optimization, in Proceeding 2rd International Conference Image Information Processing, Shimla, 2013, pp. 487–492.
[26] K. Deb, An efficient constraint handling method for genetic algorithm, Comput. Methods Appl. Mech. Eng. 186(3) (2000), 311–338.
[27] Q. He, L. Wang, A hybrid particle swarm optimization with a feasibility-based rule for constrained optimization, Appl. Math. Comput. 186 (2007), 1407–1422.
[28] J. Kennedy, R. Eberhart, Particle swarm optimization, in Proceeding of 1995 IEEE International Conference Neural Networks, Perth, 1995, pp. 1942–1948.

APPENDIX A

1. 0–1 mixed integer nonlinear programming problems

$$P_1:\quad \begin{aligned} \min\ & F = 2x + y \\ \text{s.t.}\ & 1.25 - x^2 - y \le 0 \\ & x + y \le 1.6 \\ & x \in [0, 1.6] \\ & y \in \{0, 1\} \end{aligned}$$

The known optimal solution: $F^* = 2,\ x = 0.5,\ y = 1.$

$$P_2:\quad \begin{aligned} \min\ & F = -y + 2x - ln\,(x/2) \\ \text{s.t.}\ & -x - ln\,(x/2) + y \le 0 \\ & x \in [0.5, 1.4] \\ & y \in \{0, 1\} \end{aligned}$$

The known optimal solution: $F^* = 2.1247,\ x = 1.375,\ y = 1.$

$$P_3:\quad \begin{aligned} \min\ & F = -0.7y + 5\,(x_1 - 0.5)^2 + 0.8 \\ \text{s.t.}\ & -exp\,(x_1 - 0.2) - x_2 \le 0 \\ & x_2 + 1.1y \le -1.0 \\ & x_1 - 1.2y \le 1.2 \\ & x_1 \in [0.2, 1]\,, x_2 \in [-2.22554, -1] \\ & y \in \{0, 1\} \end{aligned}$$

The known optimal solution: $F^* = 1.076543,$
$x = [0.94194, -2.1]\,,\ y = 1.$

$$P_4:\quad \begin{aligned} \min\ & F = 2x_1 + 3x_2 + 1.5y_1 + 2y_2 - 0.5y_3 \\ \text{s.t.}\ & x_1^2 + y_1 = 1.25 \\ & x_2^{1.5} + 1.5y_2 = 3 \\ & x_1 + y_1 \le 1.6 \\ & 1.333x_2 + y_2 \le 3 \\ & x_i \in [0, 2]\,, i = 1, 2 \\ & y_j \in \{0, 1\}\,, j = 1, \cdots, 3 \end{aligned}$$

The known optimal solution: $F^* = 7.667$,
$x = [1.117, 1.310]$, $y = \{0, 1, 1\}$.

$$P_5: \begin{aligned} \min\ & F = (x_1-1)^2 + (x_2-2)^2 + (x_3-3)^2 + (y_1-1)^2 \\ & + (y_2-2)^2 + (y_3-1)^2 - ln(y_4+1) \\ \text{s.t.}\ & x_1 + x_2 + x_3 + y_1 + y_2 + y_3 \le 5 \\ & x_1^2 + x_2^2 + x_3^2 + y_3^2 \le 5.5 \\ & x_1 + y_1 \le 1.2 \\ & x_2 + y_2 \le 1.8 \\ & x_3 + y_3 \le 2.5 \\ & x_1 + y_4 \le 1.2 \\ & x_2^2 + y_2^2 \le 1.64 \\ & x_3^2 + y_3^2 \le 4.25 \\ & x_3^2 + y_2^2 \le 4.64 \\ & x_1 \in [0, 1.2], x_2 \in [0, 1.281], x_3 \in [0, 2.062] \\ & y_j \in \{0,1\}, j = 1, \cdots, 4 \end{aligned}$$

The known optimal solution: $F^* = 4.5796$,
$x = [0.2, 0.8, 1.908]$, $y = \{1, 1, 0, 1\}$.

2. Mixed integer nonlinear programming (uniformly spaced integer)

$$P_6: \begin{aligned} \min\ & F = 5.357854x_1^2 + 0.835689y_1x_3 + 37.29329y_1 \\ & -40792.141 \\ \text{s.t.}\ & 85.334407 + 0.0056858y_2x_3 + 0.0029955y_1x_2 \\ & -0.0022053x_1x_3 \le 9 \\ & 80.51249 + 0.0071317y_2x_3 + 0.0029955y_1y_2 \\ & +0.0021813x_1^2 - 90 \le 20 \\ & 9.300961 + 0.0047026x_1x_3 + 0.0012547y_1x_1 \\ & +0.0019085x_1x_2 - 20 \le 5 \\ & x_i \in [27, 45], i = 1, \cdots, 3 \\ & y_1 \in \{78, \cdots, 102\}, y_2 \in \{33, \cdots, 45\} \end{aligned}$$

The known optimal solution: $F^* = -32217.4$,
$x = [27, any, 27]$, $y = \{78, any\}$.

$$P_7: \begin{aligned} \min\ & F = (y-10)^3 + (x-20)^3 \\ \text{s.t.}\ & (y-5)^2 + (x-5)^2 - 100 \ge 0 \\ & -(y-6)^2 - (x-5)^2 - 82.81 \ge 0 \\ & x \in [0, 100] \\ & y \in \{13, \cdots, 100\} \end{aligned}$$

The known optimal solution: $F^* = -4242.00473$,
$x = 3.65464$, $y = 15$

$$P_8: \begin{aligned} \min\ & F = \sum_{i=1}^{9} \left[exp - \frac{(u_i - y_2)^x}{y_1} - 0.01i \right]^2 \\ \text{s.t.}\ & u_i = 25 + \left(-50log(0.01i)\right)^{2/3} \\ & 0.1 \le y_1 \le 100, 0 \le y_2 \le 25.6, y_1, y_2 integers \\ & x \in [0, 5] \end{aligned}$$

The known optimal solution: $F^* = 0$, $x = 1.5$, $y = \{50, 25\}$

3. Mixed integer nonlinear programming (nonuniformly spaced integer)

$$P_9: \begin{aligned} \min\ & F = -x_1x_2 \\ \text{s.t.}\ & 0.145x_2^{0.1939}x_1^{0.7071}y^{-0.2343} \le 0.3 \\ & 29.67x_2^{0.4167}x_1^{-0.8333} \le 7 \\ & x_1 \in [8.6, 13.4], x_2 \in [5, 30] \\ & y \in \{120, 140, 170, 200, 230, 270, 325, 400, 500\} \end{aligned}$$

The known optimal solution: $F^* = -75.1341$,
$x = [13.4, 5.6070]$, $y = 500$

4. Integer nonlinear programming

$$P_{10}: \begin{aligned} \min\ & F = exp(-y_1) + y_1^2 - y_1y_2 - 3y_2^2 - 6y_2 + 4y_1 \\ \text{s.t.}\ & 2y_1 + y_2 \le 8.0 \\ & -y_1 + y_2 \le 2.0 \\ & y \in \{0, \cdots, 3\}, i = 1, 2 \end{aligned}$$

The known optimal solution: $F^* = -42.632$, $y = \{1, 3\}$

$$P_{11}: \begin{aligned} \min\ & F = y_1^2 + y_1y_2 + 2y_2^2 - 6y_1 - 2y_2 - 12y_3 \\ \text{s.t.}\ & 2y_1^2 + y_2^2 \le 15.0 \\ & -y_1 + 2y_2 + y_3 \le 3.0 \\ & y_i \in \{0, \cdots, 10\}, i = 1, \cdots, 3 \end{aligned}$$

The known optimal solution: $F^* = -68$, $y = \{2, 0, 5\}$

$$P_{12}: \begin{aligned} \min\ & F = y_1^2 + y_2^2 + y_3^2 + y_4^2 + y_5^2 \\ \text{s.t.}\ & y_1 + 2y_2 + y_4 \ge 4.0 \\ & y_2 + 2y_3 \ge 3.0 \\ & y_1 + 2y_5 \ge 5.0 \\ & y_1 + 2y_2 + 2y_3 \le 6.0 \\ & 2y_1 + y_3 \le 4.0 \\ & y_1 + 4y_5 \le 12.0 \\ & y_i \in \{0, \cdots, 3\}, i = 1, \cdots, 5 \end{aligned}$$

The known optimal solution: $F^* = 8$, $y = \{1, 1, 1, 1, 2\}$

$$P_{13}: \begin{aligned} \min\ & F = y_1y_7 + 3y_2y_6 + y_3y_5 + 7y_4 \\ \text{s.t.}\ & y_1 + 2y_2 + y_4 \ge 4.0 \\ & y_1 + y_2 + y_3 \ge 6.0 \\ & y_4 + y_5 + y_6 \ge 8.0 \\ & y_1y_6 + y_2 + 3y_5 \ge 7.0 \\ & 4y_2y_7 + 3y_4y_5 \ge 25.0 \\ & 3y_1 + 2y_3 + y_5 \ge 7.0 \\ & 3y_1y_3 + 6y_4 + 4y_5 \le 20.0 \\ & 4y_1 + 2y_3 + y_6y_7 \le 15.0 \\ & y_i \in \{0, \cdots, 4\}, i = 1, \cdots, 3 \\ & y_i \in \{0, \cdots, 2\}, i = 4, \cdots, 6 \\ & y_7 \in \{0, \cdots, 6\} \end{aligned}$$

The known optimal solution: $F^* = 14$, $y = \{0, 2, 4, 0, 2, 1, 4\}$

$$P_{14}: \begin{aligned} \min\ & F = -\prod_{j=1}^{4} R_j \\ \text{s.t.}\ & \sum_{j=1}^{4} d_{1j} \cdot y_j{}^2 \le 100 \\ & \sum_{j=1}^{4} d_{2j}\left(y_j + exp\left(y_j/4\right)\right) \le 150 \\ & \sum_{j=1}^{4} d_{3j}y_j exp\left(y_j/4\right) \le 160 \\ & y_j \in \{1, \cdots, 6\}, j = 1, 2, 4\ y_3 \in \{1, \cdots, 5\} \\ where\ & R_1 = 1 - q_1\left((1-\beta_1)q_1 + \beta_1\right)^{y_1 - 1} \\ & R_2 = 1 - \left(\beta_2 q_2 + p_2 q_2^{y_2}(1-\beta_2)^{y_2}\right) / \left(p_2 + \beta_2 q_2\right) \\ & R_3 = 1 - q_3^{y_3} \\ & R_4 = 1 - q_4\left((1-\beta_4)q_4 + \beta_4\right)^{y_4 - 1} \\ & \left[p_j\right] = (0.93, 0.92, 0.94, 0.91) \\ & \left[q_j\right] = (0.07, 0.08, 0.06, 0.09) \\ & \left[\beta_j\right] = (0.2, 0.06, 0.0, 0.3) \\ & \left[d_{ij}\right] = \begin{bmatrix} 1 & 2 & 3 & 4 \\ 7 & 7 & 5 & 7 \\ 7 & 8 & 8 & 6 \end{bmatrix} \end{aligned}$$

The known optimal solution: $F^* = -0.974565$, $y = \{3, 3, 2, 3\}$