Hindawi Publishing Corporation
Mathematical Problems in Engineering
Volume 2014, Article ID 627416, 10 pages
http://dx.doi.org/10.1155/2014/627416

Research Article

A Plant Propagation Algorithm for Constrained Engineering Optimisation Problems

Muhammad Sulaiman,[1,2] Abdellah Salhi,[1] Birsen Irem Selamoglu,[1] and Omar Bahaaldin Kirikchi[1]

[1] *Department of Mathematical Sciences, University of Essex, Colchester CO4 3SQ, UK*
[2] *Department of Mathematics, Abdul Wali Khan University, Mardan KPK, Pakistan*

Correspondence should be addressed to Muhammad Sulaiman; sulaiman513@yahoo.co.uk

Received 17 February 2014; Accepted 6 April 2014; Published 7 May 2014

Academic Editor: Changzhi Wu

Optimisation problems arising in industry are some of the hardest, often because of the tight specifications of the products involved. They are almost invariably constrained and they involve highly nonlinear, and non-convex functions both in the objective and in the constraints. It is also often the case that the solutions required must be of high quality and obtained in realistic times. Although there are already a number of well performing optimisation algorithms for such problems, here we consider the novel Plant Propagation Algorithm (PPA) which on continuous problems seems to be very competitive. It is presented in a modified form to handle a selection of problems of interest. Comparative results obtained with PPA and state-of-the-art optimisation algorithms of the Nature-inspired type are presented and discussed. On this selection of problems, PPA is found to be as good as and in some cases superior to these algorithms.

1. Introduction

Optimisation problems in design engineering are often highly nonlinear, constrained and involving continuous as well as discrete variables [1–5]. It is also often the case that some of the constraints are active at the global optimum [6]. This means that feasible approximate solutions are that much harder to find. There is a variety of algorithms for these problems, some exact, and others approximate. In the exact category, one can name Branch-and-Bound, [7], Recursive Quadratic Programming, [8], the Cutting Plane Algorithm [9], Bender's decomposition [10]. Of the approximate variety, one can name Simulated Annealing, [11–13], the Genetic Algorithm [14–16], and the Particle Swarm Optimisation algorithm, [17, 18], to name a few. The latter category is often referred to as the metaheuristic algorithms. In general, they are characterised by two aspects of the search: exploration of the overall search space and exploitation of good areas in order to find local optima, [19–22].

A new metaheuristic, the Plant Propagation Algorithm (PPA), has recently been introduced [23]. PPA is nature inspired [19, 23, 24]; it emulates the way plants, in particular the strawberry plant, propagate. A basic PPA has been described and tested on single objective as well as multiobjective continuous optimization problems in [23]. The test problems, though standard, were of low dimension. The results showed that PPA has merits and deserves further investigation on higher dimensional problem instances as well as problems arising in practice, for these are often very challenging. PPA is attractive because, among other things, it is simple to describe and implement; it also involves only few parameters that need arbitrary setting unlike most other metaheuristics. Here, it will be tested on constrained

optimisation problems arising in engineering. The paper is organised as follows. Section 2 describes PPA. Section 3 presents a modified version to handle constrained problems. Section 4 records the results obtained with PPA and a number of other heuristics. In Section 5 a conclusion and ideas for further investigation are given. The paper includes appendices that describe the problems considered.

2. The Strawberry Algorithm as PPA

The Strawberry algorithm is an exemplar PPA which can be seen as a multipath following algorithm unlike Simulated Annealing (SA) [13, 25], for instance, which is a single path following algorithm. It can, therefore, be perceived as a generalisation of SA and other path-following algorithms [26].

Exploration and exploitation are properties that effective global optimisation algorithms have [19, 24, 26]. Exploration refers to the property of covering the search space, while exploitation refers to the property of searching nearer to good solutions for local optima.

Consider what a strawberry plant and possibly any plant which propagates through runners will do to optimize its survival. If it is in a good spot of the ground, with enough water, nutrients, and light, then it is reasonable to assume that there is no pressure on it to leave that spot to guarantee its survival. So, it will send many short runners that will give new strawberry plants and occupy the neighbourhood as best they can. If, on the other hand, the mother plant is in a spot that is poor in water, nutrients, light, or any one of these necessary for a plant to survive, then it will try to find a better spot for its offspring. Therefore, it will send few runners further afield to explore distant neighbourhoods. One can also assume that it will send only a few, since sending a long runner is a big investment for a plant which is in a poor spot. We may further assume that the quality of the spot (abundance of nutrients, water, and light) is reflected in the growth of the plant. With this in mind and the following notation, PPA can be described as follows.

A plant p_i is in spot X_i in dimension n. This means $X_i = \{x_{i,j}, \text{ for } j = 1, \ldots, n\}$. Let NP be the number of strawberry plants to be used initially, and the PPA algorithm described in pseudo-code Algorithm 1, relies on the following strategy [23].

(i) Strawberry plants which are in good spots propagate by generating many short runners.

(ii) Those in poor spots propagate by generating few long runners.

It is clear that, in the above description, exploitation is implemented by sending many short runners by plants in good spots, while exploration is implemented by sending few long runners by plants in poor spots.

The parameters used in PPA are the population size NP which is the number of strawberry plants, the maximum number of generations $g_{\max}$, and the maximum number of possible runners $n_{\max}$ per plant. $g_{\max}$ is effectively the stopping criterion in this initial version of PPA. The algorithm uses the objective function value at different plant positions X_i, $i = 1, \ldots, NP$, in a normalised form N_i, to rank them as would a fitness function in a standard genetic algorithm (note that, unlike in the GA, individuals in PPA are clones of the mother plant; they do not improve from generation to generation). The number of plant runners n_α^i, calculated according to (1) below, has length dx^i calculated using the normalised form of the objective value at X_i, each giving a $dx^i \in (-1, 1)^n$, as calculated with (2) below. After all individuals/plants in the population have sent out their allocated runners, new plants are evaluated and the whole increased population is sorted. To keep the population constant, individuals with lower growth are eliminated. The number of runners allocated to a given plant is proportional to its fitness as in

$$n_\alpha^i = \lceil n_{\max} N_i \alpha \rceil, \quad \alpha \in (0, 1). \tag{1}$$

Every solution X_i generates at least one runner and the length of each such runner is inversely proportional to its growth as in (2) below:

$$dx_j^i = 2(1 - N_i)(\alpha - 0.5), \quad \text{for } j = 1, \ldots, n, \tag{2}$$

where n is the problem dimension. Having calculated dx^i, the extent to which the runner will reach, the search equation that finds the next neighbourhood to explore is

$$y_{i,j} = x_{i,j} + (b_j - a_j)\, dx_j^i, \quad \text{for } j = 1, \ldots, n. \tag{3}$$

If the bounds of the search domain are violated, the point is adjusted to be within the domain $[a_j, b_j]$, where a_j and b_j are lower and upper bounds delimiting the search space for the jth coordinate.

3. An Effective Implementation of PPA for Constrained Optimization

In this implementation of PPA, the initial population is crucial; we run the algorithm a number of times from randomly generated populations. The best individual from each run forms a member of the initial population. The number of runs to generate the initial population is NP; therefore, the population size is $r = NP$. In the case of mixed integer problems, the integer variable values are fixed when they are showing a trend to converge to some values. This trend is monitored by calculating the number of times their values have not changed. When this number is greater than a certain threshold, the variables are fixed for the rest of the run. This strategy seems to work on the problems considered. Let pop be a general matrix containing the population of a given run. Its rows correspond to individuals. The following equation is used to generate a random population for each of the initial runs:

$$x_{i,j} = a_j + (b_j - a_j)\, \alpha, \quad j = 1, \ldots, n, \tag{4}$$

where $x_{i,j} \in [a_j, b_j]$ is the jth entry of solution X_i and a_j and b_j are the jth entries of the lower and upper bounds describing the search space of the problem and $\alpha \in (0, 1)$.

In the main body of the algorithm, before updating the population we create a temporary population Φ to hold new solutions generated from each individual in the population. Then we implement three rules with fixed modification parameter P_m, chosen here, as $P_m = 0.8$. The first two rules are implemented if the population is initialized randomly. Rule 01 uses the following equation to update the population:

$$x^*_{i,j} = x_{i,j}(1+\beta), \quad j = 1,\ldots,n, \tag{5}$$

where $\beta \in [-1,1]$ and $x^*_{i,j} \in [a_j, b_j]$.

The generated individual X^*_i is evaluated according to the objective function and is stored in Φ. In rule 02 another individual is created with the same modification parameter $P_m = 0.8$ as in the following equation:

$$x^*_{i,j} = x_{i,j} + \left(x_{l,j} - x_{k,j}\right)\beta, \quad j = 1,\ldots,n, \tag{6}$$

where $\beta \in [-1,1]$, $x^*_{i,j} \in [a_j, b_j]$. l, k are mutually exclusive indices and are different from i.

The generated individual X^*_i is evaluated according to the objective function and is stored in Φ. The first two rules are applicable for $r \leq NP$ the number of runs. For $r > NP$ the algorithm also tries to recognise entries which are settling to their final values through a counter IN. If the jth entry in current population has a low IN value, then it is modified by implementing (7); otherwise it is left as it is. The value (for IN) that is suggested by experimentation over a number of problems is 4. The following equation is used when modification is necessary:

$$x^*_{i,j} = x_{i,j} + \left(x_{i,j} - x_{k,j}\right)\beta, \quad j = 1,\ldots,n, \tag{7}$$

where $\beta \in [-1,1]$, $x^*_{i,j} \in [a_j, b_j]$, and k is different from i. To keep the size of the population constant, the extra plants at the bottom of the sorted population are eliminated.

4. Examples of Structural Engineering Optimization Problems

PPA as explained in the pseudo-code Algorithm 1 is extended to cater for constrained optimisation problems to be found in the appendices [6, 27]. This extended version of PPA is fully explained in the pseudo-code Algorithm 2. Note that the penalty function approach is used to handle the constraints [19, 22]. Equations are first transformed into inequality constraints before they are taken into consideration. Table 1 records the parameter values used in the implementation. Column 4 shows the value of P_m used throughout the experiments. This value has been found through experimentation on the problem described in Appendix B. The different runs are represented in Figure 1 where for 40 trials corresponding to the 30000 function evaluations threshold, $P_m = 0.8$ seems to be the optimal value for this parameter. Other aspects of the extended algorithm such as exploration and exploitation are investigated in Figures 2 and 3. These figures, as one expects, show that the magnitudes of the steps/perturbations of the plant positions, that is the lengths of the runners,

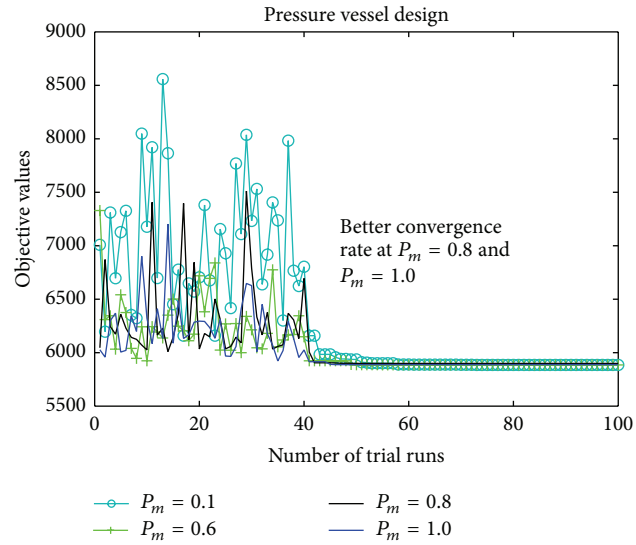

Figure 1: Impact of the particular probability P_m on the performance of PPA.

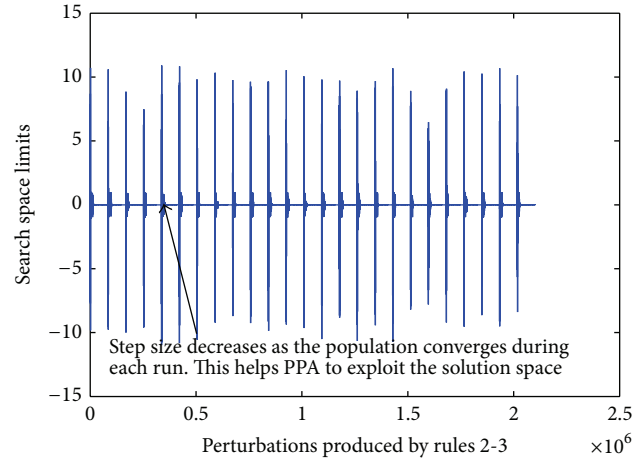

Figure 2: Exploitation characteristic of PPA using rules 2-3.

get shorter and shorter as the search progresses. Figure 4 is a representation of the convergence of the objective values of the problems described in Appendices A and C. In both cases, these values fluctuate wildly before they settle down to very good approximate values. The numerical results of the experiments on all problems described in Appendices A, through G are compiled in Tables 2, 3, 4, 5, 6, 7, and 8.

5. Conclusion

We have implemented PPA to solve seven well known difficult constrained optimization problems arising in engineering design with continuous domains. PPA found either near best known solutions or optimal ones to all of them. The results are compared to those obtained with other algorithms found in the literature, namely GA (and variants of it, here denoted EC and EP), PSO, HSA (and variants of it

Table 1: Parameters used in PPA for seven engineering problems.

Problem	Population size	Maximum iteration	P_m	Max function evaluations	Number of runners	Runs
Welded beam	40	20	0.8	30000	3	100
Pressure vessel	40	20	0.8	30000	3	100
Spring design	40	25	0.8	30000	3	100
Speed reducer	40	20	0.8	30000	3	100
Constrained Problem I	40	20	0.8	24000	3	100
Constrained Problem II	40	20	0.8	24000	3	100
Himmelblau's function	40	25	0.8	30000	3	100

Table 2: Welded beam design optimisation.

Solution vector	GA [28]	HSA [29]	HSA [30]	HHSA [31]	BDA [32]	PHS [27]	IPHS [27]	PPA
w	0.2088	0.2442	0.2057	0.2057	0.2057	0.2057	0.2057	0.2057
L	3.4205	6.2231	3.4704	3.4706	3.4704	3.4704	3.4704	3.4704
d	8.9975	8.2915	9.0366	9.0368	9.0366	9.0366	9.0366	9.0366
h	0.2100	0.2443	0.2057	0.2057	0.2057	0.2057	0.2057	0.2057
$g_1(x)$	−0.3378	*[a]	*	*	0.0	0.0	0.0	$-1.2733E-11$
$g_2(x)$	−353.9026	*	*	*	0.0	0.0	0.0	$-3.2378E-11$
$g_3(x)$	−0.0012	*	*	*	$-5.5511E-17$	$-5.5511E-17$	$-5.55E-17$	$-1.64E-13$
$g_4(x)$	−3.4118	*	*	*	−3.4329	−3.4329	−3.4329	−3.4329
$g_5(x)$	−0.0838	*	*	*	−0.0807	−0.0807	−0.0807	−0.0807
$g_6(x)$	−0.2356	*	*	*	−0.2355	−0.2355	−0.2355	−0.2355
$g_7(x)$	−363.2323	*	*	*	$-9.0949E-13$	$-9.0949E-13$	$-9.09E-13$	$-6.2755E-11$
$f(x)$	1.7483	2.38	1.7248	1.7248	1.7248	1.7248	1.7248	1.7248

[a]Not available.

Table 3: Pressure vessel design optimisation.

Solution vector	IP(M-5) [7]	GA [14]	HSA [29]	HSA [30]	PHS [27]	IPHS [27]	PPA
d_1	1.125	1.125	1.125	1.125	1.125	1.125	0.7781
d_2	0.625	0.625	0.625	0.625	0.625	0.625	0.3846
r	48.97	58.1978	58.2789	58.2901	58.2874	58.2901	40.3196
L	106.72	44.2930	43.7549	43.6926	43.7075	43.6927	200.0
$g_1(x)$	−0.1799	0.0017	−0.0002	0.0000	$-5.2058E-005$	$-3.3595E-7$	$3.627E-12$
$g_2(x)$	−0.1578	−0.0697	−0.0690	−0.0689	−0.0689	−0.0689	$1.441E-12$
$g_3(x)$	−97.760	−974.3	−3.7162	−2.0150	−0.6122	−0.0705	$1.1641E-9$
$g_4(x)$	−133.28	−195.707	−196.245	−196.307	−196.29	−196.307	−40.0
$f(x)$	7980.894	7207.494	7198.433	7197.730	7197.896	7197.730	5885.3327

Table 4: Minimization of the weight of a compression spring.

Solution vector	MP(M-5) [33]	EC [34]	GA [28]	BDAs [32]	PHS [27]	IPHS [27]	PPA
x_1	0.0500	0.0533	0.0519	0.0514	0.0500	0.0518	0.0515
x_2	0.3159	0.3991	0.3639	0.3513	0.3173	0.3608	0.3541
x_3	14.2500	9.1854	10.8905	11.6086	14.0375	11.0503	11.4387
$g_1(x)$	−0.00001	0.00001	−0.00001	−0.0033	$-4.7653E-6$	$-2.1962E-6$	$-5E-15$
$g_2(x)$	−0.0037	−0.00001	−0.00002	$-1.0970E-4$	$-1.8124E-4$	$-2.8408E-7$	$-1.3901E-9$
$g_3(x)$	−3.9383	−4.1238	−4.0613	−4.0263	−3.9672	−4.0618	−4.0487
$g_4(x)$	−0.7560	−0.6982	−0.7226	−0.7312	−0.7550	−0.7248	−0.7294
$f(x)$	0.01283	0.01273	0.01268	0.01266	0.01272	0.01266	0.01266

Table 5: Speed reducer design optimisation.

Solution vector	PSO [6]	AAI [35]	PHS [27]	IPHS [27]	PPA [$\mu = E5.488$]	PPA [$\mu = E5.93101$]
x_1	3.50	3.50	3.50	3.50	3.4922	3.4971
x_2	0.70	0.70	0.70	0.70	0.70	0.70
x_3	17.0	17.0	17.0	17.0	17.0	17.0
x_4	7.30	7.30	7.30	7.30	7.30	7.30
x_5	7.80	7.71	7.7159	7.7153	7.80	7.80
x_6	3.3502	3.35	3.3502	3.3502	3.3496	3.3500
x_7	5.2866	5.29	5.2869	5.2866	5.2846	5.2866
$g_1(x)$	−0.0739	*[a]	*	*	−0.07185	−0.07317
$g_2(x)$	−0.1979	*	*	*	−0.1962	−0.1973
$g_3(x)$	−0.4991	*	*	*	−0.4988	−0.4990
$g_4(x)$	−0.9014	*	*	*	−0.9013	−0.9014
$g_5(x)$	0.0000	*	*	*	$4.6229E-4$	$1.667E-4$
$g_6(x)$	$-5.000E-16$	*	*	*	$1.816E-3$	$6.564E-4$
$g_7(x)$	−0.7025	*	*	*	−0.70250	−0.7025
$g_8(x)$	$-1.000E-16$	*	*	*	$2.2248E-3$	$8.0440E-4$
$g_9(x)$	−0.5833	*	*	*	−0.5842	−0.58366
$g_{10}(x)$	−0.0513	*	*	*	−0.0514	−0.05136
$g_{11}(x)$	−0.0108	*	*	*	−0.01114	−0.0108
$f(x)$	2,996.3481	2994.4	2994.9	2994.4	2994.4449	2996.1137

[a]Not available.

Table 6: Constrained optimization problem I.

Solution vector	GAs [36]	EP [37]	HSA [29]	PHS [27]	IPHS [27]	PPA [$\mu = E4$]	Optimal solution
x_1	0.8080	0.8350	0.8343	0.8230	0.8229	0.8298	0.8228
x_2	0.8854	0.9125	0.9121	0.9113	0.9113	0.9098	0.9114
$g_1(x)$	$3.7E-2$	$1E-2$	$5E-3$	$1.1880E-4$	$3.8744E-5$	$7.8953E-05$	$7.05E-9$
$g_2(x)$	$5.2E-2$	$-7E-2$	$5.4E-3$	$3.5443E-4$	$1.8967E-4$	$9.1722E-05$	$1.73E-8$
$f(x)$	1.4339	2.3772	1.3770	1.3931	1.3932	1.3774	1.3935

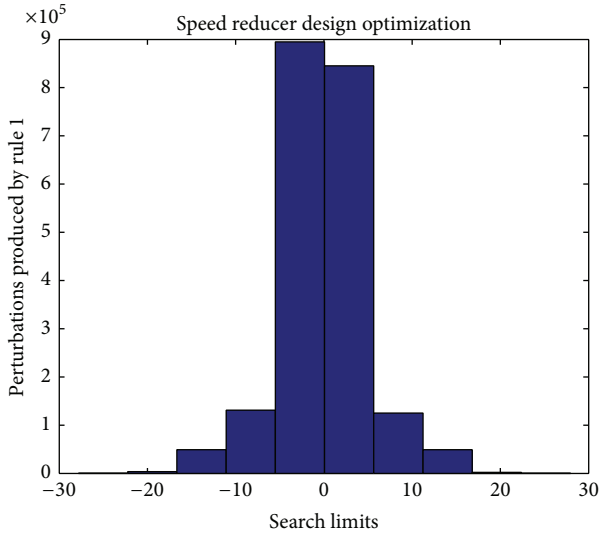

Figure 3: Exploration characteristic of PPA using rule 1.

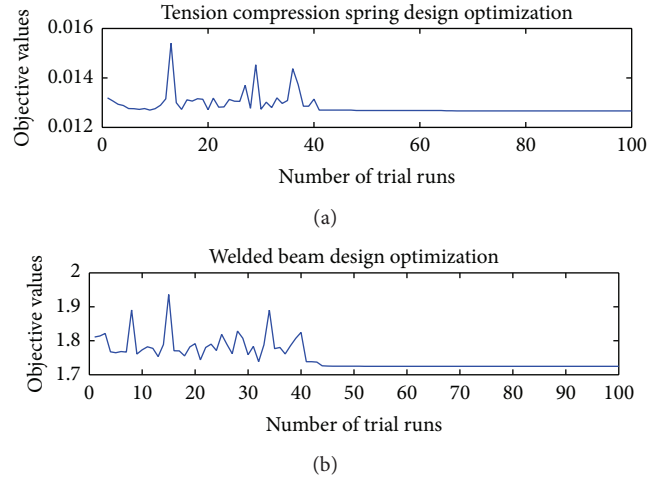

Figure 4: Convergence plot of welded beam design and tension compression spring design optimization problems over 100 trial runs.

(1) Initialization: Generate a population $P = \{X_i, i = 1, \ldots, \text{NP}\}$;
(2) $g \leftarrow 1$;
(3) **for** $g = 1: g_{\max}$ **do**
(4) Compute $N_i = f(X_i), \forall X_i \in P$;
(5) Sort P in ascending order of N (for minimization);
(6) Create new population Φ;
(7) **for** each $X_i, i = 1, \ldots, \text{NP}$ **do**
(8) $\alpha_i \leftarrow$ set of runners where both the size of the set and the distance for each runner (individually) are proportional to N_i, the normalized objective value
(9) $\Phi \leftarrow \Phi \cup \alpha_i$ {append to population};
(10) **end for**
(11) $P \leftarrow \Phi$ {new population};
(12) **end for**
(13) ***Return*** P, the population of solutions.

Algorithm 1: Pseudocode of PPA, [23].

(1) Initialization: $g_{\max} \leftarrow$ Maximum number of generations; NP $\leftarrow$ population size; $r \leftarrow$ trial run
(2) **if** $r \leq \text{NP}$ **then**
(3) Create a random population of plants $pop = \{X_i \mid i = 1, 2, \ldots, \text{NP}\}$, using (4) and gather the best solutions.
(4) **end if**
(5) **while** $r > \text{NP}$ **do**
(6) Use population pop_g formed by gathering all best solutions from previous runs.
Calculate IN_j value for each column j of pop_g (see Section 3).
(7) **end while**
(8) Evaluate the population. In case of pop_g the algorithm does not need to evaluate the population,
(9) Set number of runners, $n_r = 3$, $ngen = 1$,
(10) **while** $(ngen < g_{\max})$ **or** $(n\_eval < max\_eval)$ **do**
(11) Create Φ:
(12) **for** $i = 1$ to NP **do**
(13) **for** $k = 1$ to n_r **do**
(14) **if** $r \leq \text{NP}$ **then**
(15) **if** rand $\leq P_m$ **then**
(16) Generate a new solution X^* according to (5);
(17) Evaluate it and store it in Φ;
(18) **end if**
(19) **if** rand $\leq P_m$ **then**
(20) Generate a new solution X^* according to (6);
(21) Evaluate it and store it in Φ;
(22) **end if**
(23) **else**
(24) **for** $j = 1: n$ **do**
(25) **if** $(IN_j < 4)$ **or** (rand $\leq P_m$) **then**
(26) update the jth entry of $X_i, i = 1, 2, \ldots, \text{NP}$, according to (7);
(27) **end if**
(28) Evaluate new solution X^* and store it in Φ;
(29) **end for**
(30) **end if**
(31) **end for**
(32) **end for**
(33) Add Φ to current population;
(34) Sort the population in ascending order of the objective values;
(35) Update current best;
(36) **end while**
(37) ***Return***: Updated population.

Algorithm 2: Pseudocode of PPA for constrained optimisation.

Table 7: Constrained optimization problem II.

Algorithms	x_1	x_2	$f(x)$
Deb [38]			
GA with PS ($R = 0.01$)	*[a]	*	13.5895
GA with PS ($R = 1$)	*	*	13.5910
GA with TS-R	*	*	13.5908
HSA [30]	2.2468	2.3818	13.5908
HSA [29]	2.2468	2.3821	13.5908
PHS [27]	2.2480	2.4066	13.6117
IPHS [27]	2.2468	2.3818	13.5908
PPA	2.2468	2.3818	13.5908
Optimal solution	2.2468	2.3818	13.5908

[a]Not available.

namely PHS and IPHS), IP, MP, and BDA. Note that some of the problems have not been solved by all algorithms. For instance, the pressure vessel design problem has not been solved with BDA or MP as far as we know. Similarly the spring compression problem has not been solved with IP and HSA. These gaps in the computational results found in the literature on the considered problems are not substantial to hinder our experimental work and conclusions. Indeed, the recorded evidence in the large majority of cases points to the overwhelming superiority of PPA. Having said that, it must be added that further improvements to PPA and testing are being carried out on a more extensive collection of test problems including discrete ones.

Appendices

A. Welded Beam Design Optimisation

The welded beam design is a standard test problem for constrained design optimisation [6, 22]. There are four design variables: the width w and length L of the welded area, the depth d and thickness h of the main beam. The objective is to minimise the overall fabrication cost, under the appropriate constraints of shear stress τ, bending stress σ, buckling load P, and maximum end deflection δ. The optimization model is summarized as follows, where $x^T = (w, L, d, h)$:

$$\begin{aligned}
\text{Minimise} \quad & f(x) = 1.10471 w^2 L + 0.04811 dh\,(14.0 + L), \\
\text{subject to} \quad & g_1(x) = w - h \le 0, \\
& g_2(x) = \delta(x) - 0.25 \le 0, \\
& g_3(x) = \tau(x) - 13,600 \le 0, \\
& g_4(x) = \sigma(x) - 30,000 \le 0, \\
& g_5(x) = 1.10471 w^2 + 0.04811 dh\,(14.0 + L) \\
& \qquad - 5.0 \le 0, \\
& g_6(x) = 0.125 - w \le 0, \\
& g_7(x) = 6000 - P(x) \le 0,
\end{aligned} \tag{A.1}$$

where

$$\begin{aligned}
\sigma(x) &= \frac{504,000}{hd^2}, \qquad D = \frac{1}{2}\sqrt{L^2 + (w+d)^2}, \\
Q &= 6000\left(14 + \frac{L}{2}\right), \qquad \delta = \frac{65,856}{30,000 hd^3}, \\
J &= \sqrt{2} wL\left(\frac{L^2}{6} + \frac{(w+d)^2}{2}\right), \qquad \alpha = \frac{6000}{\sqrt{2} wL}, \\
\beta &= \frac{QD}{J}, \qquad P = 0.61423 \times 10^6 \frac{dh^3}{6}\left(1 - \frac{\sqrt[d]{30/48}}{28}\right), \\
\tau(x) &= \sqrt{\alpha^2 + \frac{\alpha\beta L}{D} + \beta^2}.
\end{aligned} \tag{A.2}$$

The simple limit or bounds are $0.1 \le L, d \le 10$ and $0.1 \le w, h \le 2.0$.

B. Pressure Vessel Design Optimisation

Pressure vessels are widely used in our daily life, such as champagne bottles and gas tanks [6, 39]. For a given volume and working pressure, the basic aim of designing a cylindrical vessel is to minimize the total cost. Typically, the design variables are the thickness d_1 of the head, the thickness d_2 of the body, the inner radius r, and the length L of the cylindrical section [6]. This is a well-known test problem for optimization, where $x^T = (d_1, d_2, r, L)$, and it can be written as

$$\begin{aligned}
\text{Minimise} \quad & f(x) = 0.6224 d_1 rL + 1.7781 d_2 r^2 \\
& \qquad + 3.1661 d_1^2 L + 19.84 d_1^2 r, \\
\text{subject to} \quad & g_1(x) = -d_1 + 0.0193 r \le 0, \\
& g_2(x) = -d_2 + 0.00954 r \le 0, \\
& g_3(x) = -\pi r^2 L - \frac{4\pi}{3} r^3 + 1296000 \le 0, \\
& g_4(x) = -L - 240 \le 0.
\end{aligned} \tag{B.1}$$

The simple limits on the design variables are

$$\begin{aligned}
0.0625 \le d_1, d_2 \le 99 \times 0.0625, \\
10.0 \le r, L \le 200.
\end{aligned} \tag{B.2}$$

C. Spring Design Optimisation

The main objective of this problem [33, 34] is to minimize the weight of a tension/compression spring, subject to constraints of minimum deflection, shear stress, surge frequency, and limits on outside diameter and on design variables. There are three design variables: the wire diameter x_1, the mean coil diameter x_2, and the number of active coils x_3 [6].

Table 8: Optimal solutions for a variation on Himmelblau's function.

Solution vector	GA [38]	HSA [29]	PHS [27]	IPHS [27]	PPA
x_1	*[a]	78.0	78.0	78.0	78.0
x_2	*	33.0	33.0	33.0	33.0
x_3	*	29.995	30.0053	29.9953	29.9952
x_4	*	45.0	45.0	45.0	45.0
x_5	*	36.776	36.7521	36.7756	36.7758
$g_1(x)$	*	*	91.9963	91.9999	92.0
$g_2(x)$	*	*	98.8362	98.8404	98.8405
$g_3(x)$	*	*	20.0	20.0	20.0
$f(x)$	−30665.500	−30665.500	−30663.845	−30665.533	−30665.540

[a]Not available.

The mathematical formulation of this problem, where $x^T = (x_1, x_2, x_3)$, is as follows:

$$\begin{aligned}
&\text{Minimize} \quad f(x) = (x_3 + 2)\, x_2 x_1^2,\\
&\text{subject to} \quad g_1(x) = 1 - \frac{x_2^3 x_3}{7,178 x_1^4} \le 0,\\
&\qquad g_2(x) = \frac{4x_2^2 - x_1 x_2}{12,566\,(x_2 x_1^3) - x_1^4} + \frac{1}{5,108 x_1^2} - 1 \le 0,\\
&\qquad g_3(x) = 1 - \frac{140.45 x_1}{x_2^2 x_3} \le 0,\\
&\qquad g_4(x) = \frac{x_2 + x_1}{1.5} - 1 \le 0.
\end{aligned} \tag{C.1}$$

The simple limits on the design variables are $0.05 \le x_1 \le 2.0$, $0.25 \le x_2 \le 1.3$, and $2.0 \le x_3 \le 15.0$.

D. Speed Reducer Design Optimization

The problem of designing a speed reducer [40] is a standard test problem. It consists of the design variables as face width x_1, module of teeth x_2, number of teeth on pinion x_3, length of the first shaft between bearings x_4, length of the second shaft between bearings x_5, diameter of the first shaft x_6, and diameter of the first shaft x_7 (all variables continuous except x_3 that is integer). The weight of the speed reducer is to be minimized subject to constraints on bending stress of the gear teeth, surface stress, transverse deflections of the shafts, and stresses in the shaft [6]. The mathematical formulation of the problem, where $x^T = (x_1, x_2, x_3, x_4, x_5, x_6, x_7)$, is as follows:

$$\begin{aligned}
&\text{Minimise} \quad f(x) = 0.7854 x_1 x_2^2 \left(3.3333 x_3^2 + 14.9334 x_3 43.0934\right) - 1.508 x_1 \left(x_6^2 + x_7^3\right) + 7.4777 \left(x_6^3 + x_7^3\right) + 0.7854 \left(x_4 x_6^2 + x_5 x_7^2\right),\\
&\text{subject to} \quad g_1(x) = \frac{27}{x_1 x_2^2 x_3} - 1 \le 0,\\
&\qquad g_2(x) = \frac{397.5}{x_1 x_2^2 x_3^2} - 1 \le 0,\\
&\qquad g_3(x) = \frac{1.93 x_4^3}{x_2 x_3 x_6^4} - 1 \le 0,\\
&\qquad g_4(x) = \frac{1.93 x_5^3}{x_2 x_3 x_7^4} - 1 \le 0,\\
&\qquad g_5(x) = \frac{1.0}{110 x_6^3} \sqrt{\left(\frac{745.0 x_4}{x_2 x_3}\right)^2 + 16.9 \times 10^6} - 1 \le 0,\\
&\qquad g_6(x) = \frac{1.0}{85 x_7^3} \sqrt{\left(\frac{745.0 x_5}{x_2 x_3}\right)^2 + 157.5 \times 10^6} - 1 \le 0,\\
&\qquad g_7(x) = \frac{x_2 x_3}{40} - 1 \le 0,\\
&\qquad g_8(x) = \frac{5 x_2}{x_1} - 1 \le 0,\\
&\qquad g_9(x) = \frac{x_1}{12 x_2} - 1 \le 0,\\
&\qquad g_{10}(x) = \frac{1.5 x_6 + 1.9}{x_4} - 1 \le 0,\\
&\qquad g_{11}(x) = \frac{1.1 x_7 + 1.9}{x_5} - 1 \le 0.
\end{aligned} \tag{D.1}$$

The simple limits on the design variables are

$$\begin{aligned}
&2.6 \le x_1 \le 3.6, \quad 0.7 \le x_2 \le 0.8,\\
&17 \le x_3 \le 28, \quad 7.3 \le x_4 \le 8.3, \quad 7.8 \le x_5 \le 8.3,\\
&2.9 \le x_6 \le 3.9, \quad 5.0 \le x_7 \le 5.5.
\end{aligned} \tag{D.2}$$

E. Constrained Optimization Problem 1

$$\begin{aligned} \text{Minimize} \quad & f(x) = (x_1 - 2)^2 + (x_2 - 1)^2, \\ \text{subject to} \quad & g_1(x) = x_1 - 2x_2 + 1 = 0, \\ & g_2(x) = \frac{-x_1}{4} - x_2^2 + 1 \geq 0, \\ & -10 \leq x_1, x_2 \leq 10. \end{aligned} \tag{E.1}$$

F. Constrained Optimization Problem 2

$$\begin{aligned} \text{Minimize} \quad & f(x) = \left(x_1^2 + x_2 - 11\right)^2 + \left(x_1 + x_2^2 - 7\right)^2, \\ \text{subject to} \quad & g_1(x) = 4.84 - (x_1 - 0.05)^2 \\ & \quad - (x_2 - 2.5)^2 \geq 0, \\ & g_2(x) = x_1^2 + (x_2 - 2.5)^2 - 4.84 \geq 0, \\ & 0 \leq x_1, x_2 \leq 6. \end{aligned} \tag{F.1}$$

G. Himmelblau's Optimization Problem

$$\begin{aligned} \text{Minimize} \quad & f(x) = 5.357847x_3^2 + 0.8356891x_1x_5 \\ & \quad + 37.293239x_1 - 40792.141, \\ \text{subject to} \quad & g_1(x) = 85.334407 + 0.0056858x_2x_5 \\ & \quad + 0.00026x_1x_4 - 0.0022053x_3x_5, \\ & g_2(x) = 80.51249 + 0.0071317x_2x_5 \\ & \quad + 0.0029955x_1x_2 + 0.0021813x_3^2, \\ & g_3(x) = 9.300961 + 0.0047026x_3x_5 \\ & \quad + 0.0012547x_1x_3 + 0.0019085x_3x_4, \\ & 0 \leq g_1(x) \leq 92, \qquad 90 \leq g_2(x) \leq 110, \\ & 20 \leq g_3(x) \leq 25, \qquad 78 \leq x_1 \leq 102, \\ & 33 \leq x_2 \leq 45, \qquad 27 \leq x_i \leq 45, \quad i = 3, 4, 5. \end{aligned} \tag{G.1}$$

Conflict of Interests

The authors have no conflict of interests.

Acknowledgments

This work is supported by Abdul Wali Khan University, Mardan, Pakistan, Grant no. F.16-5/P& D/AWKUM/238.

References

[1] X.-S. Yang, "Biology-derived algorithms in engineering optimization," in *Handbook of Bioinspired Algorithms and Applications*, S. Olarius and A. Y. Zomaya, Eds., chapter 32, pp. 589–600, Chapman & Hall/CRC Press, 2005.

[2] K. Deb, *Optimization for Engineering Design: Algorithms and Examples*, PHI Learning, 2004.

[3] X.-S. Yang, "Biology-derivedalgorithms in engineering optimization," http://arxiv.org/abs/1003.1888.

[4] H. Wang, A. Liu, X. Pan, and J. Yang, "Optimization of power allocation for multiusers in multi-spot-beam satellite communication systems," *Mathematical Problems in Engineering*, vol. 2014, Article ID 780823, 10 pages, 2014.

[5] L. Chen and H.-L. Liu, "An evolutionary algorithm based on the four-color theorem for location area planning," *Mathematical Problems in Engineering*, vol. 2013, Article ID 271935, 9 pages, 2013.

[6] L. C. Cagnina, S. C. Esquivel, and C. A. C. Coello, "Solving engineering optimization problems with the simple constrained particle swarm optimizer," *Informatica*, vol. 32, no. 3, pp. 319–326, 2008.

[7] E. Sandgren, "Nonlinear integer and discrete programming in mechnical design optimization," *Journal of Mechanical Design*, vol. 112, no. 2, pp. 223–229, 1990.

[8] J. Z. Cha and R. W. Mayne, "Optimization with discrete variables via recursive quadratic programming—part 1: concepts and definitions," *Journal of Mechanical Design*, vol. 111, no. 1, pp. 124–129, 1989.

[9] J. E. Kelley, Jr., "The cutting-plane method for solving convex programs," *Journal of the Society for Industrial & Applied Mathematics*, vol. 8, no. 4, pp. 703–712, 1960.

[10] M. L. Balinski and P. Wolfe, "On benders decomposition and a plant location problem," Tech. Rep. ARO-27, Mathematica, Princeton, NJ, USA, 1963.

[11] N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, and E. Teller, "Equation of state calculations by fast computing machines," *The Journal of Chemical Physics*, vol. 21, no. 6, pp. 1087–1092, 1953.

[12] C. Zhang and H.-P. Wang, "Mixed-discrete nonlinear optimization with simulated annealing," *Engineering Optimization*, vol. 21, no. 4, pp. 277–291, 1993.

[13] A. Salhi, L. G. Proll, D. R. Insua, and J. I. Martin, "Experiences with stochastic algorithms for a class of constrained global optimisation problems," *Recherche Operationnelle*, vol. 34, no. 2, pp. 183–197, 2000.

[14] S.-J. Wu and P.-T. Chow, "Genetic algorithms for nonlinear mixed discrete-integer optimization problems via meta-genetic parameter optimization," *Engineering Optimization*, vol. 24, no. 2, pp. 137–159, 1995.

[15] J. H. Holland and J. S. Reitman, "Cognitive systems based on adaptive algorithms," *ACM SIGART Bulletin*, no. 63, p. 49, 1977.

[16] K. Lamorski, C. Sławiński, F. Moreno, G. Barna, W. Skierucha, and J. L. Arrue, "Modelling soil water retention using support vector machines with genetic algorithm optimisation," *The Scientific World Journal*, vol. 2014, Article ID 740521, 10 pages, 2014.

[17] R. Eberhart and J. Kennedy, "A new optimizer using particle swarm theory," in *Proceedings of the 6th International Symposium on Micro Machine and Human Science (MHS '95)*, pp. 39–43, IEEE, Nagoya, Japan, October 1995.

[18] G.-N. Yuan, L.-N. Zhang, L.-Q. Liu, and K. Wang, "Passengers' evacuation in ships based on neighborhood particle swarm optimization," *Mathematical Problems in Engineering*, vol. 2014, Article ID 939723, 10 pages, 2014.

[19] X.-S. Yang, *Nature-Inspired Metaheuristic Algorithms*, Luniver Press, 2011.

[20] C. Blum and A. Roli, "Metaheuristics in combinatorial optimization: overview and conceptual comparison," *ACM Computing Surveys*, vol. 35, no. 3, pp. 268–308, 2003.

[21] V. Gazi and K. M. Passino, "Stability analysis of social foraging swarms," *IEEE Transactions on Systems, Man, and Cybernetics B: Cybernetics*, vol. 34, no. 1, pp. 539–557, 2004.

[22] X.-S. Yang and S. Deb, "Engineering optimisation by Cuckoo search," *International Journal of Mathematical Modelling and Numerical Optimisation*, vol. 1, no. 4, pp. 330–343, 2010.

[23] A. Salhi and E. S. Fraga, "Nature-inspired optimisation approaches and the new plant propagation algorithm," in *Proceedings of the The International Conference on Numerical Analysis and Optimization (ICeMATH '11)*, Yogyakarta, Indonesia, 2011.

[24] J. Brownlee, *Clever Algorithms: Nature-Inspired Programming Recipes*, 2011.

[25] E. H. L. Aarts, J. H. M. Korst, and P. J. M. van Laarhoven, "Simulated annealing," in *Local Search in Combinatorial Optimization*, pp. 91–120, 1997.

[26] G. El-Talbi, *Metaheuristics: From Design to Implementation*, vol. 74, John Wiley & Sons, 2009.

[27] M. Jaberipour and E. Khorram, "Two improved harmony search algorithms for solving engineering optimization problems," *Communications in Nonlinear Science and Numerical Simulation*, vol. 15, no. 11, pp. 3316–3331, 2010.

[28] C. A. C. Coello, "Use of a self-adaptive penalty approach for engineering optimization problems," *Computers in Industry*, vol. 41, no. 2, pp. 113–127, 2000.

[29] K. S. Lee and Z. W. Geem, "A new meta-heuristic algorithm for continuous engineering optimization: harmony search theory and practice," *Computer Methods in Applied Mechanics and Engineering*, vol. 194, no. 36–38, pp. 3902–3933, 2005.

[30] M. Mahdavi, M. Fesanghary, and E. Damangir, "An improved harmony search algorithm for solving optimization problems," *Applied Mathematics and Computation*, vol. 188, no. 2, pp. 1567–1579, 2007.

[31] M. Fesanghary, M. Mahdavi, M. Minary-Jolandan, and Y. Alizadeh, "Hybridizing harmony search algorithm with sequential quadratic programming for engineering optimization problems," *Computer Methods in Applied Mechanics and Engineering*, vol. 197, no. 33–40, pp. 3080–3091, 2008.

[32] Y. Shi and R. Eberhart, "A modified particle swarm optimizer," in *Proceedings of the IEEE World Congress on Computational Intelligence, Evolutionary Computation (ICEC '98)*, pp. 69–73, Anchorage, Alaska, USA, May 1998.

[33] A. D. Belegundu and J. S. Arora, "A study of mathematical programming methods for structural optimization—part I: theory," *International Journal for Numerical Methods in Engineering*, vol. 21, no. 9, pp. 1583–1599, 1985.

[34] J. S. Arora, *Introduction to Optimum Design*, Academic Press, 2004.

[35] P. Papalambros and H. L. Li, "A production system for use of global optimization knowledge," *Journal of Mechanical Design*, vol. 107, no. 2, pp. 277–284, 1985.

[36] A. Homaifar, C. X. Qi, and S. H. Lai, "Constrained optimization via genetic algorithms," *Simulation*, vol. 62, no. 4, pp. 242–254, 1994.

[37] D. B. Fogel, "A comparison of evolutionary programming and genetic algorithms on selected constrained optimization problems," *Simulation*, vol. 64, no. 6, pp. 397–404, 1995.

[38] K. Deb, "An efficient constraint handling method for genetic algorithms," *Computer Methods in Applied Mechanics and Engineering*, vol. 186, no. 2–4, pp. 311–338, 2000.

[39] X.-S. Yang, "Firefly algorithm, stochastic test functions and design optimisation," *International Journal of Bio-Inspired Computation*, vol. 2, no. 2, pp. 78–84, 2010.

[40] J. Golinski, "An adaptive optimization system applied to machine synthesis," *Mechanism and Machine Theory*, vol. 8, no. 4, pp. 419–436, 1973.