Hindawi
Mathematical Problems in Engineering
Volume 2018, Article ID 6357935, 8 pages
https://doi.org/10.1155/2018/6357935

Research Article

On the Theoretical Analysis of the Plant Propagation Algorithms

Muhammad Sulaiman,[1] **Abdellah Salhi**,[2] **Asfandyar Khan**,[1] **Shakoor Muhammad**,[1] **and Wali Khan**[3]

[1]*Department of Mathematics, Abdul Wali Khan University Mardan, Khyber Pakhtunkhwa, Pakistan*
[2]*Department of Mathematical Sciences, University of Essex, Wivenhoe Park, Colchester CO4 3SQ, UK*
[3]*Department of Mathematics, Kohat University of Science & Technology (KUST), Khyber Pakhtunkhwa, Pakistan*

Correspondence should be addressed to Muhammad Sulaiman; sulaiman513@yahoo.co.uk and Abdellah Salhi; as@essex.ac.uk

Received 10 September 2017; Accepted 21 January 2018; Published 21 March 2018

Academic Editor: Haranath Kar

Plant Propagation Algorithms (PPA) are powerful and flexible solvers for optimisation problems. They are nature-inspired heuristics which can be applied to any optimisation/search problem. There is a growing body of research, mainly experimental, on PPA in the literature. Little, however, has been done on the theoretical front. Given the prominence this algorithm is gaining in terms of performance on benchmark problems as well as practical ones, some theoretical insight into its convergence is needed. The current paper is aimed at fulfilling this by providing a sketch for a global convergence analysis.

1. Introduction

The theoretical analysis of stochastic algorithms for global optimisation is not new and can be found in a number of sources such as [1–5]. The majority of the algorithms considered use random search one way or another to find the optimum solution [6–13]. Here, we consider the algorithmic scheme of the Plant Propagation Algorithm for continuous optimisation or PPA-C [14] and theoretically investigate its global convergence to the optimum solution. The optimisation problems of concern are continuous and defined in finite n-dimensional domains.

The basic version of PPA [15] models the propagation of strawberry plants. The scheme uses short runners for exploitation or local search refinement while long runners are used for diversification and exploration of the search space. Since the propagation of strawberries is due to seeds as well as runners, a Seed-based Plant Propagation Algorithm (SbPPA) has also been introduced in [16]. Both PPA-C and SbPPA have been shown to be efficient on continuous unconstrained and constrained optimisation problems; statistical convergence analyses of PPA-C and SbPPA can be found in [9–11, 14–16].

PPA-C [14, 15] consists of two steps:

(1) Initialization: a population of parent plants is generated randomly.

(2) Propagation: a new population is created from persistent parents (strawberry plants) and their children (new strawberry plants at the end of runners, i.e., a distance away from parent plants).

Let S denote the search space such that $S \subset R^n$, where n is its dimension. By an iteration of PPA-C we mean a new generation of child plants produced by parent plants. These child plants are the result of either short or long runners [14, 16]. This is the basic setup that we consider to sketch a proof of convergence to the global optimum of a given continuous optimisation problem.

The paper is organised as follows. Section 2 presents the terminology used in the analysis of PPA-C. Section 3 analyses a population of plants. Section 3.1 describes the convergence analysis of PPA-C. Section 4 is the conclusion.

2. Terminology and Notation

We consider single objective minimization problems [17]. $X^{\text{optimal}} \in S$ such that $f(X^{\text{optimal}}) \leq f(X)$ for all $X \in S$,

where the objective function is defined as $f : S \subset R^n \rightarrow R$, denotes the best spot for a plant in the search space. X is an n-dimensional position vector.

The population at the gth iteration is denoted by $\text{pop}_g = \{X_{1,g}, X_{2,g}, \ldots, X_{NP,g}\}$, where NP is the population size. The coordinates of runners, or more precisely their endpoints, are denoted by $X_i = (x_i^1, x_i^2, \ldots, x_i^n)^T$, where n is the space dimension of the given problem.

2.1. Search Equations and Evaluation of New Plants. Variants of PPA can be found in [14–16]. In this paper we analyse PPA-C as Algorithm 1 of [14].

In order to send a short or long runner, X_i' is generated [14, 19–21], as in (1a), (1b), and (1c)

$$x_{i,j}' = x_{i,j} + \beta_j x_{i,j} \quad \text{if } \text{rand}_j \leq P_m,\ r \leq NP, \tag{1a}$$

$$x_{i,j}' = x_{i,j} + \left(x_{l,j} - x_{k,j}\right)\beta_j \quad \text{if } \text{rand}_j \leq P_m,\ r \leq NP \tag{1b}$$

$$x_{i,j}' = x_{i,j} + \left(x_{i,j} - x_{k,j}\right)\beta_j \quad \text{if } \text{rand}_j \leq P_m \text{ or } IN_j < 4,\ r > NP, \tag{1c}$$

where NP is the population size, r is the Monte Carlo trial run counter, P_m is the modification probability, and $\text{rand}_j \in (0\,1)$ is a randomly generated number for each jth entry, $j = 1, 2, \ldots, n$. The indices $i, l, k = 1, 2, \ldots, NP$ are mutually exclusive; that is, $i \neq k \neq l$. Another version of PPA called SbPPA [16] which is inspired by propagation via seeds implements the following search equation instead:

$$x_{i,j}^* = \begin{cases} x_{i,j} + L_i\left(x_{i,j} - \theta_j\right) & \text{if } PR \leq 0.8,\ \theta_j \in \left[a_j b_j\right]\ i = 1, 2, \ldots, NP;\ j = 1, 2, \ldots, n \\ x_{i,j} & \text{Otherwise,} \end{cases} \tag{2}$$

where L_i is a step drawn from the Lévy distribution [22] and θ_j is a random coordinate within the search space. Equations (1a), (1b), (1c), and (2) perturb the current solution, the results of which can be seen in Figures 1(a) and 1(b), respectively.

2.2. A Case Study. Let c be the class of runners sent by the ith parent plant and stored in F. Each runner in class c is decomposed into two vectors τ_c and ω_c, where τ_c denotes the vector of indices which are perturbed with respect to the current position of the plant, while ω_c represents the vector of corresponding indices of the unperturbed coordinates with respect to the current position of plants. This can be represented as

$$\tau_c \cup \omega_c = \{1, 2, \ldots, n\}. \tag{3}$$

To clarify this idea, let us take an example [17] of a newly generated runner by the ith plant as

$$X_i' = \left[x_1', x_2', x_3', x_4', x_5'\right]^T, \tag{4}$$

such that

$$\begin{aligned} \tau_c &= \{1, 2, 4\}, \\ \omega_c &= \{3, 5\}; \end{aligned} \tag{5}$$

then we can write

$$X_i' = \left(X_{\tau_c}', X_{\omega_c}'\right), \tag{6}$$

where

$$\begin{aligned} X_{\tau_c}' &= \left[x_1', x_2', 0, x_4', 0\right], \\ X_{\omega_c}' &= \left[0, 0, x_3', 0, x_5'\right]. \end{aligned} \tag{7}$$

The dot product of these vectors is zero, which shows that they are mutually orthogonal. Mathematically, this can be written as

$$X_{\tau_c}' \bullet X_{\omega_c}' = 0. \tag{8}$$

Let V_{τ_c} and V_{ω_c} denote two vector spaces such that

$$\begin{aligned} V_{\tau_c} &= \text{containing vectors having dimensions as in } \tau_c, \\ V_{\omega_c} &= \text{containing vectors having dimensions as in } \omega_c. \end{aligned} \tag{9}$$

V_{τ_c} and V_{ω_c} are subspaces of R^n. This implies that $X_{\tau_c}' \in V_{\tau_c}$ and $X_{\omega_c}' \in V_{\omega_c}$.

A scalar objective function f defined over X_i' can be represented as

$$f\left(X_i'\right) = f\left(x_1', x_2', x_3', x_4', x_5'\right), \tag{10}$$

where $x_1' = X_{\tau_1}$, $x_2' = X_{\tau_2}$, $x_4' = X_{\tau_3}$ and $x_3' = X_{\omega_1}$, $x_5' = X_{\omega_2}$,

$$\Longrightarrow f\left(X_{\tau_1}, X_{\tau_2}, X_{\omega_1}, X_{\tau_3}, X_{\omega_2}\right) = f\left(X_{\tau_c}, X_{\omega_c}\right), \tag{11}$$

where (11) represents an objective value corresponding to a new runner in position X_i'. Similarly, different runners are produced to correspond to different classes c and evaluated by the same procedure. This procedure can be generalized for n-dimensional problems [1–3].

3. Graphical and Theoretical Analysis of a Population of Plants

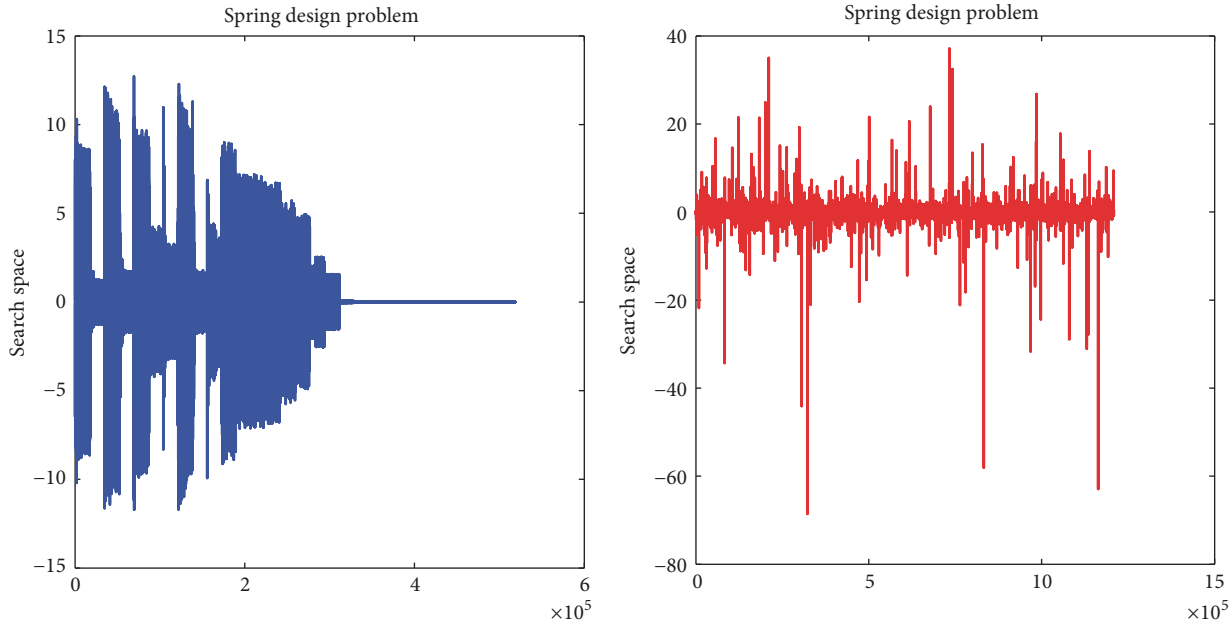
Algorithm 1 states that the jth coordinate of an ith parent plant is perturbed with probability P_m and it remains

```
(1) r ← Counter for trial runs; NP ← Population size
(2)
(3) F ← Population of runners
(4)
(5) for r = 1 : 100 do
(6)
(7)     if r ≤ NP then
(8)
(9)         Create a random population of plants pop = {X_i | i = 1, 2, ..., NP}, and
            gather the best solution from each run.
(10)
(11)    end if
(12)
(13)    while r > NP do
(14)
(15)        Use population pop_g formed by gathering all the best solutions of previous runs.
            Calculate IN_j value for each column j of pop_g.
(16)
(17)    end while
(18)
(19)    Evaluate the population pop.
(20)
(21)    Assume number of runners to be n_r = 3,
(22)
(23)    while (the stopping criteria is not satisfied) do
(24)
(25)        for i = 1 to NP do
(26)
(27)            for h = 1 to n_r do
(28)
(29)                if r ≤ NP then
(30)
(31)                    if rand ≤ P_m then
(32)
(33)                        Generate a new solution X'^1 according to Equation (1a);
(34)
(35)                        Evaluate it and store it in F;
(36)
(37)                    end if
(38)
(39)                    if rand ≤ P_m then
(40)
(41)                        Generate a new solution X'^2 according to Equation (1b);
(42)
(43)                        Evaluate it and store it in F;
(44)
(45)                    end if
(46)
(47)                else
(48)
(49)                    for j = 1 : n do
(50)
(51)                        if (IN_j < 4) or (rand ≤ P_m) then
(52)
(53)                            Update the jth entry of X_i, i = 1, 2, ..., NP, by using Equation (1c);
(54)
(55)                        end if
(56)
(57)                        Evaluate new solution X'^3 and store it in F;
(58)
(59)                    end for
(60)
```

Algorithm 1: Continued.

```
(61)                end if
(62)
(63)            end for
(64)
(65)        end for
(66)
(67)        Append F to current population;
(68)
(69)        Sort the population in ascending order of objective values;
(70)
(71)        Update current best;
(72)
(73)    end while
(74)
(75)    Return: Updated population and global best solution.
(76)
(77) end for
(78)
```

Algorithm 1: PPA for constrained optimisation [14].



(a) Perturbations by (1a), (1b), and (1c)

(b) Perturbations by (2)

Figure 1: Overall performance of (1a), (1b), (1c), and (2) on a design optimisation problem given in Appendix [16].

unchanged with probability $1 - P_m$. Thus there are 2^n possible runners to be generated for each ith parent plant using (1a), (1b), and (1c), where n is the space dimension of the given problem.

Let, at any generation g, the random population be represented as pop $= \{X_{1,g}, X_{2,g}, \ldots, X_{NP,g}\}$, where NP denotes the population size. To create a runner by using (1a), (1b), and (1c), for next generation $g+1$, PPA uses a population of parent plants at generation g for this purpose. It is not required to know about any other runner in generation $g+1$. This shows that all runners created at generation $g+1$ are statistically mutually independent. Furthermore, the initial population is random and all parent plants do not depend on each other. Thus, by induction, the runners at any further generations are mutually independent.

From (1a), (1b), and (1c), a runner X_i' may be formed by itself or by choosing three different coordinates from current population. In case of using (1a) [19], there are NP possibilities to send (long or short) runners as in Figure 1. On the other hand, by using (1b) or (1c), $NP - 1$ vectors

are used to send a new runner. Thus, in later cases there are ${}^{NP-1}P_3$ possibilities to send new runners. In (1a)–(1c), different possibilities are of the form

$$\begin{aligned} X_g^{\prime 1} &= X_{i,g} + \beta \cdot X_{i,g}, \\ X_g^{\prime 2} &= X_{i,g} + \beta \cdot X_{j,g}, \\ X_g^{\prime 3} &= X_{i,g} + \beta \cdot X_{k,g}, \end{aligned} \tag{12}$$

where $X_{i,g}$, $X_{j,g}$, and $X_{k,g}$ are calculated according to (1a), (1b), and (1c), in which g denotes the current generation and β is an n-dimensional random vector within interval $[-1\ 1]$. The probability density functions (PDFs) [17, 23] of these new vectors $X_{i,g}$, $\beta \cdot X_{i,g}$, $\beta \cdot X_{j,g}$, and $\beta \cdot X_{k,g}$ can be written as $p_{X_{i,g}}(x_{i,g})$, $p_{X_{i,g}}(x_{i,g}/\beta)/\beta$, $p_{X_{j,g}}(x_{j,g}/\beta)/\beta$, $p_{X_{k,g}}(x_{k,g}/\beta)/\beta$, respectively.

The PDFs of new runners created with (1a)–(1c) are given in (15).

Definition 1 (convolution operation $\star$ [24]). Let $f(x)$ and $g(x)$ be Laplace transformable piecewise continuous functions defined on $[0\ \ \infty]$. The convolution product of these two functions is again a function of x defined as

$$(f \star g)(x) = \int_0^x f(\xi)\, g(x-\xi)\, d\xi, \tag{13}$$

$$\begin{aligned} p_{X_g^{\prime 1}}\left(x_g^{\prime 1}\right) &= \frac{1}{\beta}\left(p_{X_{i,g}}\left(x_g^{\prime 1}\right) \star p_{X_{i,g}}\left(\frac{x_g^{\prime 1}}{\beta}\right)\right), \\ p_{X_g^{\prime 2}}\left(x_g^{\prime 2}\right) &= \frac{1}{\beta}\left(p_{X_{i,g}}\left(x_g^{\prime 2}\right) \star p_{X_{j,g}}\left(\frac{x_g^{\prime 2}}{\beta}\right)\right), \\ p_{X_g^{\prime 3}}\left(x_g^{\prime 3}\right) &= \frac{1}{\beta}\left(p_{X_{i,g}}\left(x_g^{\prime 3}\right) \star p_{X_{k,g}}\left(\frac{x_g^{\prime 3}}{\beta}\right)\right), \end{aligned} \tag{14}$$

$$\Downarrow$$

$$\begin{aligned} &p_{X_{i,g}^{\prime 1}}\left(x_{i,g}^{\prime 1}\right) \\ &\quad = \frac{1}{\beta_{i,1} NP}\left(\sum\sum p_{X_{i,g}}\left(x_{i,g}^{\prime 1}\right) \star p_{X_{i,g}}\left(\frac{x_{i,g}^{\prime 1}}{\beta_{i,1}}\right)\right), \\ &p_{X_{i,g}^{\prime 2}}\left(x_{i,g}^{\prime 2}\right) \\ &\quad = \frac{1}{\beta_{i,2}{}^{NP-1}P_3}\left(\sum\sum_{i\neq j} p_{X_{i,g}}\left(x_{i,g}^{\prime 2}\right) \star p_{X_{j,g}}\left(\frac{x_{i,g}^{\prime 2}}{\beta_{i,2}}\right)\right), \\ &p_{X_{i,g}^{\prime 3}}\left(x_{i,g}^{\prime 3}\right) \\ &\quad = \frac{1}{\beta_{i,3}{}^{NP-1}P_3}\left(\sum\sum_{i\neq k} p_{X_{i,g}}\left(x_{i,g}^{\prime 3}\right) \star p_{X_{k,g}}\left(\frac{x_{i,g}^{\prime 3}}{\beta_{i,3}}\right)\right). \end{aligned} \tag{15}$$

Let X_{g+1} be any plant in generation $g+1$, X_g a plant position in current generation, and $X_g^{\prime h}$ the runner produced by the ith plant $X_{i,g}$ at the generation g. Then the joint PDF of the parent plant in the next generation X_{g+1} based on the parent X_g and runner $X_g^{\prime h}$, where $h = 1, 2, 3$, is given by

$$\begin{aligned} &p_{X_{g+1},X_g,X_g^{\prime h}}\left(x_{g+1}, x_g, x_g^{\prime h}\right) \\ &\quad = p_{X_{g+1}|X_g,X_g^{\prime h}}\left(x_{g+1} \mid x_g, x_g^{\prime h}\right) \cdot p_{X_g^{\prime h}|X_g}\left(x_g^{\prime h} \mid x_g\right) \\ &\qquad \cdot p_{X_g}\left(x_g\right) \cdot p_{X_g^{\prime h}}\left(x_g^{\prime h}\right). \end{aligned} \tag{16}$$

Note that a runner is selected for the next population only if its rank is less than or equal to NP, the population size. Its objective value is less than the maximum objective value (in case of minimization problem) in the current population. Note also that instead of greedy selection we sort the population and eliminate those plants whose rank is higher than NP. The model for this selection mechanism can be represented as follows:

$$\begin{aligned} p_{X_{g+1}|X_g,X_g^{\prime h}}\left(x_{g+1} \mid x_g, x_g^{\prime h}\right) &= \delta\left(x_{g+1} - x_g^{\prime h}\right), \\ \forall X_g^{\prime h} \in F;\ \forall X_g \in \text{pop};\ f\left(X_g\right) &\geq f\left(X_g^{\prime h}\right). \end{aligned} \tag{17}$$

3.1. Convergence Analysis of PPA-C. For illustration purposes, we have implemented a combined version of (1a), (1b), (1c), and (2). This version of PPA-C called H-PPA-SbPPA is a hybridisation of PPA and SbPPA [10]. We have plotted the position of plants in populations through solving the Branin and Matyas test functions (see Figures 2, 3, 4, and 5). It is obvious from Figures 2 and 4 that (1a), (1b), and (1c) have generated short runners which exploit the search space locally. On the other hand, in Figures 3 and 5, (2) has generated a diverse range of solutions which are spread over the whole search space. This equation helps the algorithm escape from local minima and to explore the solution space better and hence the global search qualities of this algorithm. Mathematically this can be shown as follows.

Let S denote the search space containing the solution of a given optimisation problem defined as

$$\min \quad \{f(X) \mid X \in S\}, \tag{18}$$

where $f(X)$ is the objective function. Then the optimal solution set [25] can be represented as

$$\begin{aligned} &S^* \\ &\quad = \left\{X^{\text{optimal}} \mid f\left(X^{\text{optimal}}\right) = \min\left\{f(X)\right\},\ \forall X \in S\right\}, \end{aligned} \tag{19}$$

where X^{optimal} is the optimum solution. The region of attraction [25] of the solution set S^* is defined as

$$S_\xi^* = \left\{X \mid f\left(X^{\text{best}}\right) - f(X) < \xi\right\}, \tag{20}$$

where ξ is a small positive real number and X^{best} is the current best solution.

In PPA-C, each parent plant produces $X_g^{\prime h}$ runners (solutions), where $h = 1, 2, 3$. The probability that at generation

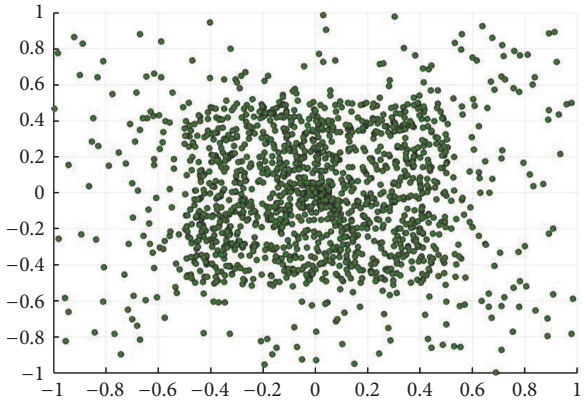


Figure 2: The exploitation capability of PPA while solving Branin test function.

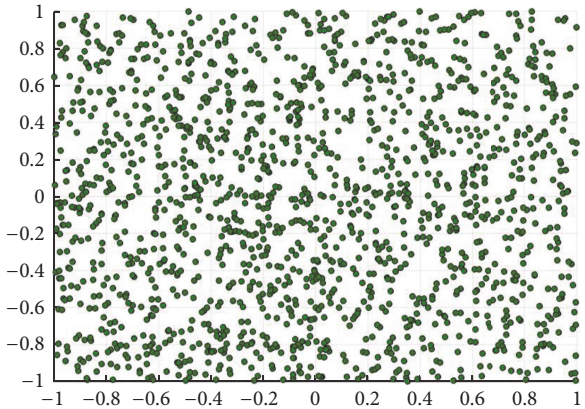


Figure 3: The exploration capability of SbPPA while solving Branin test function.

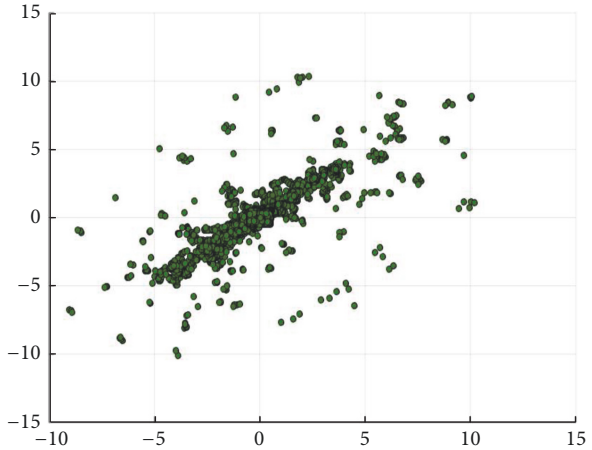


Figure 4: A scatter plot of plants produced by PPA when optimising Matyas function [15].

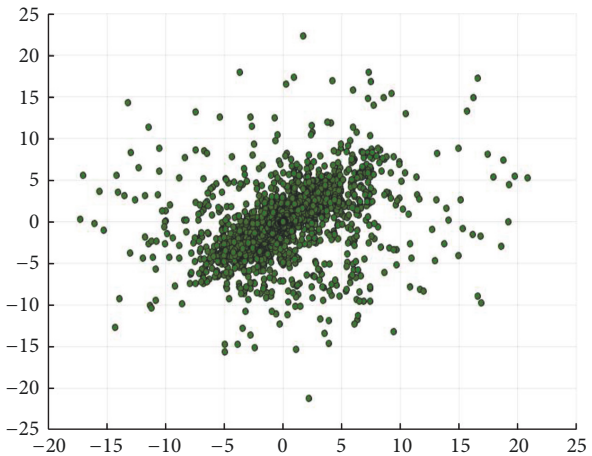


Figure 5: A scatter plot of plants produced by SbPPA when optimising Matyas function [16, 18].

g a subset of the temporary population $F' \subset F$, containing solutions which are not good enough to be retained in the next generation by the selection model as in (17), is given as

$$p\left\{F'_g \cap S^*_\xi = \varnothing\right\} \leq 1 - \xi_g, \tag{21}$$

where ξ_g is a small positive real number. Obviously, in all previous generations $g-1$, some of the solutions died and some succeeded to survive into the next generation. This shows that in previous generations we have some solutions X'^h_g which do not belong to the region of attraction S^*_ξ.

$$\prod_{i=1}^{g-1} p\left\{F'_i \cap S^*_\xi = \varnothing\right\} \leq \prod_{j=1}^{g-1}\left(1 - \xi_j\right). \tag{22}$$

At the end of each generation, the temporary population F is appended to the main population pop_g. Then all individuals are sorted with respect to their objective values. The individuals with higher rank than size of population are omitted. Thus the probability that a generation g does not contain an optimum is given below

$$\begin{aligned} p\left\{\text{pop}_g \cap S^*_\xi = \varnothing\right\} &\approx \prod_{i=1}^{g-1} p\left\{F'_i \cap S^*_\xi = \varnothing\right\} \\ &\leq \prod_{j=1}^{g-1}\left(1 - \xi_j\right). \end{aligned} \tag{23}$$

After sorting the final population at the end of each generation, the probability that the optimum may exist in a subpopulation F' (population of weak or dead runners/solutions) is less than that of the population pop_g. This can be represented as

$$\begin{aligned} \lim_{g\to\infty} p\left\{\text{pop}_g \cap S^*_\xi \neq \varnothing\right\} &\geq \lim_{g\to\infty} p\left\{F'_g \cap S^*_\xi \neq \varnothing\right\} \\ &= 1 - \lim_{g\to\infty} p\left\{F'_g \cap S^*_\xi = \varnothing\right\} = 1 - \prod_{g=1}^{+\infty}\left(1 - \xi_g\right). \end{aligned} \tag{24}$$

Following [25–27], the right hand side term of inequality (24) is zero if the series $\sum_{g=1}^{+\infty}(\xi_g)$ diverges. The convergence of PPA follows and can be summarised in the theorem below.

Theorem 2. *PPA converges to the global optimum with probability 1 if it is left to run for a reasonable amount of time [1–3]; in other words,*

$$\lim_{g\to\infty} p\left\{\text{pop}_g \cap S_\xi^* \neq \varnothing\right\} = 1. \tag{25}$$

Remark 3. Every population pop_g includes some solutions which are in set S_ξ^*.

Remark 4. The above remark is due to the exploitation characteristic of PPA-C.

Remark 5. Remark 3 implies that X^{best} is always improving or changing its position until the optimum is reached.

Remark 6. X^{best} converges approximately to X^{optimal}, as generation g grows.

4. Conclusion

The Plant Propagation Algorithm (PPA) and its variants for continuous optimisation problems are getting notoriety as flexible and powerful solvers. PPA is a heuristic inspired by the way plants and in particular the strawberry plant propagate. It is also referred to as the Strawberry Algorithm. While there is a growing body of experimental and computational works that show its good behaviour and performance against well-established algorithms and heuristics, there is very little if any in terms of theoretical investigation. This gap, therefore, needs to be filled. The aim of course, in analysing the convergence of any algorithm (in this case PPA-C), is to give confidence to the potential users that the solutions that it returns are of good quality. The convergence analysis put forward in this paper relies on the exploitation and exploration characteristics of the algorithm. Since it does not get stuck in local optima and explores thoroughly the search space it is only a matter of time before the global optimum is discovered. The approach is probabilistic in nature and ascertains that the global optimum will be found with probability 1 provided the algorithm is run reasonably long enough. The argument for this to hold is that at each iteration new and better solutions are generated which means that, in the limit, the global optimum is reached. Questions still remain concerning what is considered a reasonable amount of time. Bounds on the time it will take to converge are being developed and results will be presented in a follow-up paper.

Appendix

Spring Design Optimisation

The main objective of this problem [28, 29] is to minimize the weight of a tension/compression string, 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 [30]. The mathematical formulation of this problem, where $x^T = (x_1, x_2, x_3)$, is as follows:

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

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 $2.0 \le x_3 \le 15.0$.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

This work has been partially sponsored by ESRC Grant ES/L011859/1.

References

[1] A. Zhigljavsky and A. Žilinskas, *Stochastic Global Optimization*, vol. 9, Springer Science & Business Media, 2007.

[2] C. A. Floudas and P. M. Pardalos, *Encyclopedia of Optimization*, Springer Science & Business Media, 2008.

[3] Y. D. Sergeyev, R. G. Strongin, and D. Lera, *Introduction to Global Optimization Exploiting Space-Filling Curves*, Springer Science & Business Media, 2013.

[4] N. Brahimi, A. Salhi, and M. Ourbih-Tari, "Convergence of the plant propagation algorithm for continuous global optimisation," *RAIRO Operations Research*, 2018.

[5] J. He and L. Kang, "On the convergence rates of genetic algorithms," *Theoretical Computer Science*, vol. 229, no. 1-2, pp. 23–39, 1999.

[6] W. K. Mashwani, A. Salhi, M. A. Jan, R. A. Khanum, and M. Sulaiman, "Enhanced version of multi-algorithm genetically adaptive for multiobjective optimization," *Advanced Computer Science and Applications (IJACSA)*, vol. 12, no. 6, 2015.

[7] W. K. Mashwani, A. Salhi, O. Yeniay, M. A. Jan, and R. A. Khanum, "Hybrid adaptive evolutionary algorithm based on decomposition," *Applied Soft Computing*, vol. 57, pp. 363–378, 2017.

[8] W. K. Mashwani, A. Salhi, M. A. Jan, M. Sulaiman, R. A. Khanum, and A. Algarni, "Evolutionary algorithms based on decomposition and indicator functions: state-of-the-art survey," *Advanced Computer Science and Applications (IJACSA)*, vol. 7, no. 2, 2016.

[9] M. Sulaiman, *A Nature-inspired Metaheuristic: The Plant Propagation Algorithm [Ph.D. thesis]*, University of Essex, 2015.

[10] M. Sulaiman and A. Salhi, "A hybridisation of runner-based and seed-based plant propagation algorithms," *Studies in Computational Intelligence*, vol. 637, pp. 195–215, 2016.

[11] M. Sulaiman, A. Salhi, E. S. Fraga, W. K. Mashwani, and M. M. Rashidi, "A novel plant propagation algorithm: modifications and implementations," *Science International*, vol. 28, no. 1, 2015.

[12] B. I. Selamoglu, A. Salhi, and M. Sulaiman, "Strip algorithms as an efficient way to initialise population-based metaheuristics," in *Recent Developments in Metaheuristics*, pp. 319–331, Springer, 2018.

[13] M. Sulaiman, A. Ahmad, A. Khan, and S. Muhammad, "Hybridized Symbiotic Organism Search Algorithm for the Optimal Operation of Directional Overcurrent Relays," *Complexity*, vol. 2018, Article ID 4605769, 11 pages, 2018.

[14] M. Sulaiman, A. Salhi, B. I. Selamoglu, and O. B. Kirikchi, "A plant propagation algorithm for constrained engineering optimisation problems," *Mathematical Problems in Engineering*, vol. 2014, Article ID 627416, 10 pages, 2014.

[15] A. Salhi and E. Fraga, "Nature-inspired optimisation approaches and the new plant propagation algorithm," in *Proceedings of the International Conference on Numerical Analysis and Optimization (ICeMATH '11)*, pp. K2–1–K2–8, Yogyakarta, Indonesia, 2011.

[16] M. Sulaiman and A. Salhi, "A seed-based plant propagation algorithm: the feeding station model," *The Scientific World Journal*, vol. 2015, Article ID 904364, 16 pages, 2015.

[17] S. Ghosh, S. Das, A. V. Vasilakos, and K. Suresh, "On convergence of differential evolution over a class of continuous functions with unique global optimum," *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, vol. 42, no. 1, pp. 107–124, 2012.

[18] M. Sulaiman and A. Salhi, "The 5th International Conference on Metaheuristics and Nature Inspired Computing," in *Proceedings of the 5th International Conference on Metaheuristics and Nature Inspired Computing*, Morocco, UAE, October 2014, http://meta2014.sciencesconf.org/40158.

[19] R. Hooke and R. Jeeves, "Direct search solution of numerical and statistical problems," *Journal of the ACM*, vol. 8, pp. 212–229, 1961.

[20] R. S. Rahnamayan, H. R. Tizhoosh, and M. M. A. Salama, "Opposition-based differential evolution," *IEEE Transactions on Evolutionary Computation*, vol. 12, no. 1, pp. 64–79, 2008.

[21] B. Akay and D. Karaboga, "A modified Artificial Bee Colony algorithm for real-parameter optimization," *Information Sciences*, vol. 192, pp. 120–142, 2012.

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

[23] M. Taboga, *Lectures on Probability Theory and Mathematical Statistics*, CreateSpace Independent Pub., 2012.

[24] R. N. Bracewell and R. Bracewell, *The Fourier Transform and Its Applications*, vol. 31999, McGraw-Hill, New York, NY, USA, 1986.

[25] Z. Hu, S. Xiong, Q. Su, and X. Zhang, "Sufficient conditions for global convergence of differential evolution algorithm," *Journal of Applied Mathematics*, vol. 2013, Article ID 193196, 2013.

[26] C. Chen, F. Jin, X. Zhu, and G. Ouyan, *Mathematics Analysis*, Press: Higher Education Press, 2000.

[27] K. Knopp, *Theory and Application of Infinite Series*, Courier Dover Publications, 2013.

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

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

[30] L. C. Cagnina, S. C. Esquivel, and C. A. C. Coello, "Solving engineering optimization problems with the simple constrained particle swarm optimizer," *Informatica (Slovenia)*, vol. 32, pp. 319–326, 2008.