RAIRO-Oper. Res.
DOI: 10.1051/ro/2017037

RAIRO Operations Research
www.rairo-ro.org

CONVERGENCE ANALYSIS OF THE PLANT PROPAGATION ALGORITHM FOR CONTINUOUS GLOBAL OPTIMIZATION

Nassim Brahimi[1], Abdellah Salhi[2] and Megdouda Ourbih–Tari[3]

Abstract. The Plant Propagation Algorithm (PPA) is a Nature-Inspired stochastic algorithm, which emulates the way plants, in particular the strawberry plant, propagate using runners. It has been experimentally tested both on unconstrained and constrained continuous global optimization problems and was found to be competitive against well established algorithms. This paper is concerned with its convergence analysis. It first puts forward a general convergence theorem for a large class of random algorithms, before the PPA convergence theorem is derived and proved. It then illustrates the results on simple problems.

Mathematics Subject Classification. 90C09, 05C15.

Received August 28, 2016. Accepted May 6, 2017.

1. Introduction

In recent years, a number of random search algorithms, the so-called heuristic and meta-heuristic algorithms, have been suggested for optimization. They are popular because they are easy to understand and implement, and they work well in practice although they do not guarantee optimality. Theoretical conditions under which convergence to the global optimum is achieved, can be derived.

In the class of biologically inspired optimization algorithms, there are several types (Brownlee [6]; Yang [28]). The Flower Pollination Algorithm (FPA) is inspired by the pollination of flowers through different agents such as insects (Yang [29]); the Swarm Data Clustering (SDC) algorithm is inspired by the pollination of flowers by bees (Kazemian *et al.* [17]); Particle Swarm Optimization (PSO) is inspired by the foraging behaviour of groups of animals and insects (Eberhart and Kennedy [9]; Clerc [7]) the Artificial Bee Colony (ABC) algorithm simulates the foraging behaviour of honeybees (Karaboga [15]; Karaboga and Basturk [16]); the Firefly algorithm is inspired by flashing fireflies to attract a mate (Yang [27]; Gandomi *et al.* [10]); the Social Spider Optimization (SSO) algorithm is inspired by the cooperative behaviour of social spiders (Cuevas and Cienfuegos [8]). The list can easily be extended.

Keywords. Strawberry algorithm, randomised algorithms, convergence analysis, global optimisation.

[1] Laboratoire de Mathématiques appliquées, Faculté des Sciences Exactes, Université de Bejaia, 06000, Bejaia, Algeria.
nassim.brahimi@univ-bejaia.dz

[2] Department of Mathematical Sciences, Faculty of Science and Health, University of Essex, CO4 3SQ, United Kingdom.
as@essex.ac.uk

[3] Institut des Sciences et de la Technologie, Centre Universitaire Morsli Abdellah de Tipaza, 42000, Tipaza, Algeria.
ourbihmeg@gmail.com

Article published by EDP Sciences

The Plant Propagation Algorithm (PPA) also known as the Strawberry Algorithm, introduced by Salhi and Fraga, in 2011, has been found to be competitive with many of the algorithms here mentioned; its performance characteristics have been established numerically by (Sulaiman *et al.* [25], [23]). However, its theoretical analysis is yet to be carried out. This is what we intend to do in this paper. More precisely, we investigate the convergence of PPA based on the propagation by runners as in the strawberry plant (Salhi and Fraga [20]). Note that there is a version of PPA which is based on the propagation of the strawberry plant using seeds, (Sulaiman *et al.* [24]). Runner-based PPA follows the principle that plants in good spots with plenty of nutrients send many short runners. Those in nutrient poor spots send few long runners. Long runners allow PPA to explore the search space (exploration) while short runners enable it to carry out refined local searches (exploitation).

Analyses of the convergence behaviour of some of Nature-Inspired algorithms have already been carried out. For instance, (Yang [26]) analyzed the convergence of Simulated Annealing, (Gutjahr [11], [12]; Brahimi *et al.* [5]) analyzed the convergence of the Ant Colony algorithm, (Bergh and Engelbrecht [3]) analyzed Particle Swarm Optimization, (Agapie *et al.* [2]; Bienvenue and Francois [4]) and others analyzed that of Evolutionary Algorithms.

Recall that convergence analysis provides an estimate of the range of the parameter values of the algorithm that allows it to find the optimum solution. It can also improve these values leading potentially to a better performance of the algorithm.

Our approach to the analysis here is first to propose a general convergence theorem that provides the range of parameter values that ensure convergence to the global optimum for a large class of algorithms. Second, we deduce a specific convergence theorem for PPA. This theorem is general enough to be applicable in a wider range of continuous optimization contexts that can be formulated as

$$\min_{x\in\Omega} f(x), \tag{1.1}$$

where Ω is a subset of $\mathbb{R}^n$ and f is a real-valued function defined on Ω and assumed to have at least one global minimum over Ω. Note that Ω is the feasible region defined by the constraint set of problem (1). Illustrations will be provided. Here, we investigate the asymptotic convergence. The general convergence theorem is especially important for algorithms with a bounded distribution. For those which have a non null distribution throughout the search space, the asymptotic convergence is trivial, as we shall see later.

The paper is organized as follows. In Section 2, we present the original PPA and a variant of it which is more amenable to analysis. In Section 3 we state and prove the general convergence theorem and we deduce from it the conditions under which the global convergence of PPA is achieved. Section 4 discusses special cases and provides examples. Section 5 is the conclusion.

2. The Plant Propagation Algorithm

PPA as the Strawberry algorithm, (Salhi and Fraga [20]), is a neighborhood search-type algorithm. However, while Variable Neighborhood Search (VNS) (Hansen and Mladenovie [13]) is trajectory-based, like Simulated Annealing (Aarts *et al.* [1]; Salhi *et al.* [21]), PPA is population-based.

Let us for a moment contemplate what a strawberry plant, and possibly any plant which propagates through runners, does to optimize its survival. If it is in a good spot of the ground, with enough water, nutrients and light, then one can reasonably assume that there is no pressure on it to expand further afield and away from that spot. So, it sends many short runners that give new strawberry plants and occupy the neighborhood of the mother plant as best they can. On the other hand, if 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 off-spring. Therefore, it will send few runners away from its current position to explore distant neighborhoods. It is also reasonable to 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. It is also reasonable to assume that the quality of the spot (abundance of nutrients, water and light) is reflected in the growth of the plant. With these assumptions in hand, consider the following notation that will help describe the general PPA/Strawberry Algorithm.

A plant p_i is in spot X_i in dimension n. This means that $X_i = (x_{i,1}, \ldots, x_{i,n})$. A general PPA paradigm would therefore be as in Algorithm 1. Note that in this algorithm, 'Good/Bad' refers to the value that function F achieves at X_i; for a maximization problem, large values are good. It is clear that, in this paradigm, exploitation is implemented by sending many short runners. Exploration is implemented by sending few long runners by plants in poor spots; the long runners allow to explore distant neighborhoods.

Algorithm 1. Paradigm of PPA

```
Generate a population of NP plants, P = {p_i, i = 1,...,NP} in spots X_i, i = 1,...,NP, with plant p_i
achieving growth F(X_i);
For i from 1 to NP Do
    If growth F(X_i) of plant p_i is 'good' Then
        Send n_r > 1 short runners to generate n_r new plants
        {X_i + dx_1, X_i + dx_2, ..., X_i + dx_{n_r}};
    Else
        Send one long runner X_i + dx;
    Endif
Endfor
Keep the best NP plants as the new population;
Repeat from Step 3 until a termination criterion is satisfied;
Return the best plant as candidate solution.
```

The parameters used in PPA are: the population size, the number of generations, the number of runners per plant and their lengths which are the distances from current solutions to new ones. Length/distance dx, is not an arbitrary parameter since it is decided according to the objective value of the solution X_i, in a normalized form, giving a $dx \in [-1, 1]$, as calculated with equation 2.2 below. Equally, the number of runners per plant can be decided according to the quality of the spot where the plant happens to be, *i.e.* the value of the objective function at that point. It, therefore, does not have to be set arbitrarily. This means PPA requires only two arbitrary parameters: the population size NP, and the maximum number of generations $g_{\max}$, which is one of many possible stopping criteria. This issue of arbitrary parameters is important in heuristic and meta-heuristic design. Algorithms such as PPA with few arbitrary parameters are more desirable than those with many such parameters. This is because finding good default parameters is difficult and relying on too much experimentation restricts the use of the algorithm.

To highlight the attractiveness/desirability of PPA based on this "loose" metric that is the number of arbitrary parameters, we consider two of the well established Nature-inspired algorithms namely the Genetic Algorithm (GA) (Holland [14]) and Simulated Annealing (SA) (Metropolis *et al.* [19], Kirkpatrick *et al.* [18]). Recall the list of arbitrarily set parameters which are necessary for the implementation of GA. They are:

(1) The population size;
(2) The maximum number of generations;
(3) The number of generations without improvement to stop;
(4) The rate of crossover;
(5) The rate of mutation;
(6) The length of the chromosome;
(7) The number of points of crossover.

Recall also the list of parameters required for the implementation of SA.

(1) The maximum temperature;
(2) The minimum temperature;

(3) α the percentage improvement in the objective value expected in each move;
(4) The maximum number of moves without achieving $\alpha\%$ improvement in objective value;
(5) The number of iterations at each temperature.

To this, one can add the individual temperatures to be considered between the minimum and maximum temperatures.

Clearly, compared to the two parameters that PPA requires, it is fair to say that on the above metric, PPA is easier to implement. This, coupled with its competitive performance, (Sulaiman *et al.* [23, 25]), makes it, therefore, more desirable.

The individuals in the random population of PPA represent strawberry plants. After all individuals/plants in the population have sent out their allocated runners, new plants are created and evaluated, and the resulting increased population is sorted. To keep the population constant, individuals with lower fitness and ranking below the NP^{th} position, are eliminated. Note that the number of runners allocated to a given plant are proportional to its fitness as in,

$$n_r = \lceil n_{\max} N_i r \rceil, \tag{2.1}$$

where n_r is the number of runners generated by the i^{th} plant in the population. $n_{\max}$ is the maximum number of runners to generate. N_i is the fitness value, *i.e.* the normalized objective function value of the i^{th} solution. $r \in [0, 1]$, is randomly selected for each solution in each generation. Every solution X_i generates at least one runner and the length of each runner is inversely proportional to its fitness as in equation below

$$dx_j^r = 2(1 - N_i)(r - 0.5), \text{ for } j = 1, \ldots, n, \tag{2.2}$$

where n is the number of dimensions. Having calculated dx^r, the extent to which the runner will reach, the search equation that finds exactly where to find the next neighborhood to explore is

$$y_{i,j} = x_{i,j} + (b_j - a_j) dx_j^r, \text{ for } j = 1, \ldots, n. \tag{2.3}$$

The new solution is almost guaranteed to be within $[a, b]$, where a and b are lower and upper bounds delimiting the search space. When it is not, the offending coordinates are forced to take the upper/lower bound values as may be necessary. Note that new solutions are generated in the hyper-cube

$$[a_1, b_1] \times \ldots \times [a_n, b_n].$$

3. A Variant of PPA

Here we introduce a variant of PPA for the continuous global optimization problem which is more concise than the above one and more amenable to mathematical analysis as will be seen later. The variant of PPA for solving Problem (1.1) suggested here, can be built from the original PPA as follows. As we previously noticed, the original PPA generates its new populations in a hyper-cube of side at most $[a_j, b_j]$, in dimension $j = 1, \ldots, n$. Generating new individuals from each plant in position X_i in a ball of center X_i and radius at most $R_{\max} \in]0, +\infty[$ (which is the maximum length of runners) is more streamlined. It is important to point out that parameter $R_{\max}$ will be used later, in defining the convergence condition of PPA. Note that if a plant is generated outside of Ω, it dies out and so it is eliminated (or it is projected onto Ω). As we study the convergence of the algorithm, there is no need to define a stopping criterion (such as the maximum number of generations). Convergence is shown for t tending to $+\infty$. Thus, we substitute in the algorithm formulas 2.2 and 2.3 by

$$Y_i = X_i + r(1 - N_i) R_{\max} S^2, \tag{3.1}$$

where $Y_i = (y_{i,1}, \ldots, y_{i,n})$ are the new plants, $r \rightsquigarrow U(0, 1)$ and S^2 is the random vector uniformly distributed in the Euclidean unit sphere. The fitness N_i of $p_i, i = 1, \ldots, NP$, takes its values in $[0, 1[$ and can be defined as

$$N_i = (1 - \alpha) \frac{\max_i f(X_i) - f(X_i)}{\max_i f(X_i) - \min_i f(X_i)}, i = 1, \ldots, NP \text{ and } 0 < \alpha < 1. \tag{3.2}$$

Algorithm 2. Variant of PPA

```
Generate a population P = {p_i, i = 1, ..., NP} in spots
X_i, i = 1, ..., NP, with plant p_i achieving growth f(X_i);
Initialise the generation counter: g ← 1;
For g from 1 to ∞ Do
    Compute f(X_i), ∀p_i ∈ P;
    Sort P in ascending order according to the fitness of its
    elements;
    Create new population Φ;
    For X_i, i = 1, ..., NP Do
        r_i ← set of runners where both size of set and length of
        each runner are proportional to the fitness/objective
        value f_i as in Equation 3.1;
        Φ ← Φ ∪ r_i {append to population; death occurs by omission};
    Endfor
    P ← Φ {new population};
Endfor
Return P, the population of solutions.
```

Thus, $N_i \in [0, 1-\alpha]$. Note that the choice of the fitness function influences the performance of the algorithm.

The variant of PPA as Algorithm 2, first considers NP plants uniformly randomly generated in Ω. Each plant then generates n_r new plants by equation 3.1. Finally, the algorithm selects the best NP individuals among new and old, according to f, the objective function of problem (1). A new generation/iteration starts with a new population.

For ease of analysis, we represent Algorithm 2 by its current solutions $P_t = \{X_t^1, \ldots, X_t^N\}$. PPA will refer, from now on, to the variant of PPA, or Algorithm 2.

4. Convergence analysis

We proceed by presenting a general convergence theorem for a large class of stochastic algorithms when applied to continuous global optimization problems. From it, we, later deduce the PPA convergence theorem. The concern here is with asymptotic convergence, *i.e.* when t tends to $+\infty$. Let us first define the implements used in the theorem.

Definition 4.1. Let $\Omega \subseteq \mathbb{R}^n$. We define $Open(\Omega)$ as the set of open sets of Ω.

Definition 4.2. Let $\Omega \subseteq \mathbb{R}^n$ and O the function defined by

$$\begin{aligned} O : \Omega &\longmapsto Open(\Omega) \\ x &\longmapsto O_x. \end{aligned}$$

We say that function O is continuous within the meaning of Lebesgue if and only if for all $x \in \Omega$

$$\lim_{y \longrightarrow x} \lambda \left(O_y \Delta O_x \right) = 0$$

where $y \in \Omega$ is in the vicinity of x and λ is the measure of Lebesgue.

Note that, this definition means that O_x deforms continuously to $x \in \Omega$ because

$$\lambda(A) = 0 \Longleftrightarrow A = \emptyset$$

for $A \in Open(\Omega)$.

Definition 4.3. Let f be a continuous function defined from $\Omega \subseteq \mathbb{R}^n$ to $\mathbb{R}$. Then, $M_{\text{loc}}(f)$ is the set of strict local minima of f and $M_{\text{glo}}(f)$ its set of global minima.

Lemma 4.4. *Let C be an open set of $\Omega \subseteq \mathbb{R}^n$ and L a function defined from Ω to $Open(\Omega)$ which is continuous in the sense of Lebesgue. Let $x_0 \in \Omega$. If $L_{x_0} \cap C$ is not empty, then, there exists a vicinity of x_0 noted V_{x_0} such that*

$$\forall x \in V_{x_0},\ L_x \cap C \neq \emptyset.$$

Proof. Suppose that $A = L_{x_0} \cap C$ is not empty; this set is open. From the continuity of L we have, $\forall \epsilon > 0, \exists \eta > 0$ such that

$$\|x - x_0\| < \eta \Longrightarrow \lambda\left(L_x \Delta L_{x_0}\right) < \epsilon.$$

So, by letting $B_x = A \cap \left(L_x \Delta L_{x_0}\right)$ and $\epsilon = \lambda\left(A\right)/2$ then, $\exists \eta > 0$ such that

$$\forall x \in B(x_0, \eta), \lambda\left(B_x\right) < \lambda\left(L_x \Delta L_{x_0}\right) < \frac{\lambda\left(A\right)}{2} \Longrightarrow B_x \subsetneq A,$$

and so

$$\forall x \in B(x_0, \eta), L_x \cap C = A \backslash B_x \neq \emptyset.$$

Since the ball is a vicinity of x_0, the result follows. □

The following theorem gives the convergence of any elitist random algorithm (statement (1) below) that is represented by a stochastic process $(X_t)_{t\geq 0}$ which has the property of continuity defined in statement (2) below. Such algorithms generate their new solutions at the $(t+1)^{th}$ generation/iteration with the probability density l_{X_t}, which does not depend on time t. Afterwards, they choose the best solutions among new and old. Here, we consider without loss of generality, the convergence analysis of such algorithms on continuous functions with at least one global minimum. The general idea is to show that the algorithm can get out of a local minimum, *i.e.* it doesn't get stuck, after a finite number of iterations; this means, it is sufficient to show that the algorithm progresses, or there is improvement in the search, with positive probability for all $x \in \Omega$. Recall that $\overset{\circ}{A}$ is the Interior of A, and $\overline{A}$ its closure.

Theorem 4.5. *Let $\Omega \subseteq \overline{\overset{\circ}{\Omega}} \subseteq \mathbb{R}^n$. Let f be a continuous function defined from Ω to $\mathbb{R}$ which admits at least one global minimum. If an algorithm generating the current solution $(X_t)_{t\geq 0}$, satisfies the following conditions*

(1) *$f\left(X_{t+1}\right) \leq f\left(X_t\right)$ for all $t \in \mathbb{N}$ and $X_0 \in \Omega$.*
(2) *The function L defined by*

$$L_x = \left\{s \in \Omega : l_x(s) > 0\right\},$$

is such that $\overset{\circ}{L}$ is continuous in the sense of Lebesgue,

where l_x is the probability density function of the distribution with which PPA generates its solutions, i.e. the uniform distribution over the ball $B(x, R_{\max})$, then, it converges to the global minimum with probability 1 if and only if

$$\forall x \in \Omega \backslash M_{\text{glo}} : \overset{\circ}{L}_x \cap C_x \neq \emptyset,$$

with $C_x = \{s \in \overset{\circ}{\Omega} : f(s) < f(x)\}$.

Proof. From the convergence condition given in the theorem, we get

$$\lambda\left(\overset{\circ}{L}_x \cap C_x\right) \neq 0 \ for\ x \in \Omega \backslash M_{\text{glo}}.$$

So, we have

$$\Pr\left(X_{t+1} \in C_{X_t} | X_t \in \Omega \backslash M_{\text{glo}}\right) = \int_{C_{X_t}} l_{X_t}(s) ds \tag{4.1}$$

$$= \int_{L_{X_t} \cap C_{X_t}} l_{X_t}(s) ds > 0.$$

Thus, we conclude with the condition (1) of the theorem that if $M_{\text{loc}} = \emptyset$, then our algorithm converges to a global minimum. And if $M_{\text{loc}} \neq \emptyset$ then, due to the continuity of function f, we get that $\forall x_{\text{loc}} \in M_{\text{loc}}$,$C_{x_{\text{loc}}}$ is a non empty open. Add this to the condition (2) of the theorem, and the fact that

$$\forall x_{\text{loc}} \in M_{\text{loc}}, \overset{\circ}{L}_{x_{\text{loc}}} \cap C_{x_{\text{loc}}} \neq \phi,$$

we can apply the Lemma 4.4. Thus, for each $x_{\text{loc}} \in M_{\text{loc}}$, there exists a vicinity of x_{loc} noted $V_{x_{\text{loc}}}$ such that for $A = V_{x_{\text{loc}}} \cap \Omega$, we have

$$\forall x \in A, \exists B_x = \overset{\circ}{L}_x \cap C_{x_{\text{loc}}} \neq \emptyset,$$

with $\lambda(A) \neq 0$. We note that

$$\forall x \in A,\ f(B_x) < f(x_{\text{loc}}) \leq f(A).$$

Thus, we get

$$\Pr\left(X_{t+1} \in B_{X_t} | X_t \in A\right) > 0.$$

More generally, since for all $x \in A$, $B_x \subset C_{x_{\text{loc}}}$, we get

$$\Pr\left(X_{t+1} \in C_{x_{\text{loc}}} | X_t \in A\right) > 0.$$

This means that the algorithm can get out from a vicinity of a local optimum. Add this to fact 4.1 above, we get the first implication.

For the converse, it is clear that if for all $x \in \Omega \backslash M_{\text{glo}}$ the process $(X_t)_{t \geq 0}$ with $X_0 = x$, converges to the global minimum then, for all $x \in \Omega \backslash M_{\text{glo}}$, the probability of progression is not null. So, the Lebesgue measure of $\overset{\circ}{L}_x \cap C_x$ is also not null, and therefore the set is not empty, as expected. □

Note that, Ω does not need to be compact or convex. Moreover, it can be replaced by C_{X_0}, after X_0 is chosen.

To establish that Algorithm 2 will converge to the global minimum for any starting point(s), we state the following result which gives a range of parameter values of $R_{\max}$. The idea of this theorem is as follows; the ball $B(x, R_{\max})$ must meet the progression set C_x, what is clear for non-optimal points; we, therefore, need to give the conditions for local optimality.

Theorem 4.6. *Let $\Omega \subseteq \overline{\overset{\circ}{\Omega}} \subseteq \mathbb{R}^n$. Let f be a continuous function defined on Ω taking its values in $\mathbb{R}$, and admitting at least one global minimum. Then, PPA converges to a global minimum for any starting point $X_0 \in \Omega$ with probability 1, if and only if*

$$R_{\max} > \begin{cases} \max_{x_{\text{loc}} \in M_{\text{loc}}} \left\{d\left(x_{\text{loc}}, C_{x_{\text{loc}}}\right)\right\} & if \quad M_{\text{loc}} \neq \emptyset \\ 0 & otherwise \end{cases}$$

with $\max_{x_{\text{loc}} \in M_{\text{loc}}} \left\{d\left(x_{\text{loc}}, C_{x_{\text{loc}}}\right)\right\} < +\infty$ and d being the Euclidian distance.

Proof. Describe the solutions of the t^{th} generation as follows

$$P_t = \left(X_t^1, \ldots, X_t^{NP}\right) \ with \ f\left(X_t^i\right) \leq f(X_t^j), 1 \leq i < j \leq NP.$$

It suffices to establish the convergence of each i^{th} current solution. Note that $X_t = X_t^i$ for i fixed. First, by the fact that PPA is elitist, we have, for all $X_0 \in \Omega$, $f\left(X_{t+1}\right) \leq f\left(X_t\right)$. Secondly, define for all $x \in \Omega$

$$L_x = B(x, R_{\max}) \cap \Omega \quad and \quad C_x = \left\{ s \in \overset{\circ}{\Omega} : f(s) < f(x) \right\}.$$

It is clear that the function $\overset{\circ}{L}$ is continuous on Ω and

$$\forall x \in \Omega \backslash (M_{\text{loc}} \cup M_{\text{glo}}) : \overset{\circ}{L}_x \cap C_x \neq \phi.$$

Since, in addition, $\exists \epsilon > 0$ such that

$$R_{\max} = \max_{x_{\text{loc}} \in M_{\text{loc}}} \left\{ d\left(x_{\text{loc}}, C_{x_{\text{loc}}}\right) \right\} + \epsilon \Longrightarrow \forall x_{\text{loc}} \in M_{\text{loc}}, R_{\max} \geq d\left(x_{\text{loc}}, C_{x_{\text{loc}}}\right) + \epsilon,$$

then, it is clear that

$$\forall x \in \Omega \backslash M_{\text{glo}} : \overset{\circ}{L}_x \cap C_x \neq \phi.$$

Now, if we have

$$R_{\max} \leq \max_{x_{\text{loc}} \in M_{\text{loc}}} \left\{ d\left(x_{\text{loc}}, C_{x_{\text{loc}}}\right) \right\} \Longrightarrow \exists x_{\text{loc}} \in M_{\text{loc}}, \exists \epsilon \geq 0 : R_{\max} = d\left(x_{\text{loc}}, C_{x_{\text{loc}}}\right) - \epsilon,$$

then, it is clear that

$$\overset{\circ}{L}_{x_{\text{loc}}} \cap C_{x_{\text{loc}}} = \phi.$$

We conclude that the condition on $R_{\max}$ is equivalent to the convergence condition of the Theorem 4.5. This ends the proof. □

Note that, PPA may converge to any one of the global minima.

5. Illustration

To illustrate the above results, we consider two examples, one with a single global minimum and no strict local minima and the other with many local minima.

Example 5.1. Let f be a positive definite quadratic function. We know that f admits a unique (global) minimum. From Theorem 4.6, it is clear that, for any $R_{\max} > 0$, PPA converges to the unique global minimum of f.

Example 5.2. This example shows how the algorithm can get stuck in a local minimum and how it gets out of it. Consider the continuous function

$$x \longmapsto |x| \left(\frac{3}{2} - \cos(x) \right).$$

As can be seen in the graph below, $R_{\max} > 4.4$ is the convergence condition of Algorithm 2, the variant of PPA, to the minimum.

Note that for values of $R_{\max} < 4.3$ only a local minimum is guarantedd to be found. However, global optimality cannot be guaranteed.

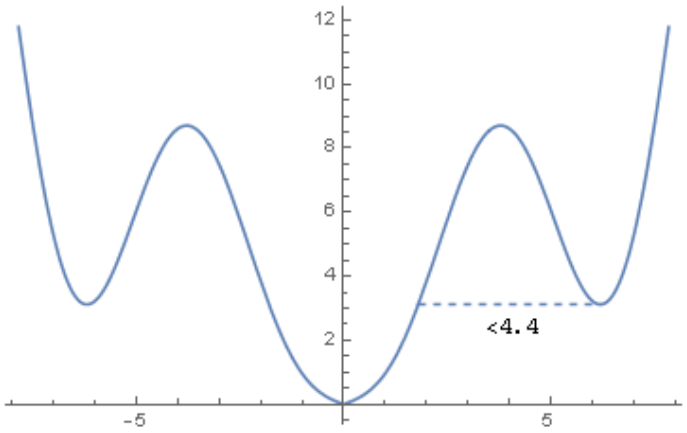

FIGURE 1. Illustration of the global convergence condition

6. Conclusion

In this paper, we have given a pseudo-code of the general PPA paradigm as Algorithm 1 and details of its implementation. Algorithm 1 searches in the hypercube centered at a given solution X_i. For ease of analysis, we have described a variant of it which searches in the hypersphere centred at X_i. The pseudo-code of this more concise version is given as Algorithm 2. To analyse its convergence, we have first shown in Theorem 1 under what conditions a general class of stochastic algorithms converges to a global minimum with probability 1. We have then extended this result to cater for the convergence of PPA as Algorithm 2, in Theorem 2. This required showing that parameter $R_{\max}$ determines its global convergence. We have then given the estimate of the range of values of $R_{\max}$ which guaranty convergence to the global minimum with a probability 1. The results also show under what conditions PPA and algorithms in the same class, get stuck in local minima.

Follow up work will try to extend the theorem to other classes of stochastic algorithms for both continuous and discrete problems.

References

[1] E.H.L. Aarts, J. Korst and P.J.M. Van Laarhoven, Simulated Annealing. In *Local Search in Combinatorial Optimization.* edited by E.H.L. Aarts and J.K. Lenstra. Wiley (1997) 91–120.

[2] A. Agapie, M. Agapie, G. Rudolph, G. Zbaganu, Convergence of evolutionary algorithms on the n-dimensional continuous space. Cybernetics. *IEEE Trans.* **43** (2013) 1462–1472.

[3] F. Bergh and A. Engelbrecht, A convergence proof for the particle swarm optimiser. *Fundamenta Informaticae* **105** (2010) 341–374.

[4] A. Bienvenue and O. Francois, Global convergence for evolution strategies in spherical problems: some simple proofs and difficulties. *Theoretical Comput. Sci.* **306** (2003) 269–289.

[5] N. Brahimi, A. Salhi and M. Ourbih−Tari, Drift Analysis of Ant Colony Optimization of Stochastic Linear Pseudo-Boolean Functions To appear in *Operat. Res. Lett.* (2017).

[6] J. Brownlee, Clever Algorithms: Nature-Inspired Programming Recipes (2011).

[7] M. Clerc, *Particle Swarm Optimization*, John Wiley and Sons **93** (2010).

[8] E. Cuevas and M. Cienfuegos, A new algorithm inspired in the behavior of the social-spider for constrained optimization. *Expert Syst. Appl.* **41** (2014) 412–425.

[9] R. Eberhart, J. Kennedy, A new optimizer using particle swarm theory. In *Proceedings of the 6th Inter. Sympos. Micro Machine and Human Scie.* (MHS'95) (1995) 39–43, IEEE, Nagoya, Japan.

[10] A.H. Gandomi, X.-S. Yang and A.H. Alavi, Mixed variable structural optimization using Firefly Algorithm. *Comput. Structures* **89** (2011) 2325–2336.

[11] W. Gutjahr, A graph-based ant system and its convergence. *Future Generation Comput. Syst.* **16** (2000) 873–888.

[12] W. Gutjahr, ACO algorithms with guaranteed convergence to the optimal solutions. *Information Processing Lett.* **82** (2002) 145-153.

[13] P. Hansen and N. Mladenovic, Variable neighborhood search: Principles and applications. *Eur. J. Operat. Res.* **130** (2001) 449–467.
[14] J.H. Holland, Adaptation in Natural and Artificial Systems, University of Michigan Press. *Ann. Arbor, MI* (1974).
[15] D. Karaboga, An idea based on honey bee swarm for numerical optimization. *Tech. Rep. (TR06)*, Erciyes University Press, Kayseri, Turkey (2005).
[16] D. Karaboga and B. Basturk, On the performance of artificial bee colony (ABC) algorithm. *Appl. Soft Comput. J.* **8** (2008) 687–697.
[17] M. Kazemian, Y. Ramezani, C. Lucas and B. Moshiri, Swarm clustering based on flowers pollination by artificial bees. *Studies in Comput. Intell.* **34** (2006) 191–202.
[18] S. Kirkpatrick, C.D. Gelatt and M.P. Vecchi, Optimization by Simulated Annealing. *Sci.* **220** (1983) 671–680.
[19] N. Metropolis, A.W. Rosenbluth, M. Rosenbluth, A.H. Teller and E. Teller, Equation of State Calculations by Fast Computing Machines. *J. Chem. Phys.* **21** (1953) 1087–1092.
[20] 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), (2011) K2-1–K2-8, Yogyakarta, Indonesia.
[21] A. Salhi, L.G. Proll, D. Rios Insua and J.I. Martin, Experiences with stochastic algorithms for a class of constrained global optimisation problems. *Recherche Opér.* **34** (2000) 183-197.
[22] A. Salhi, J.A. Vázquez−Rodríguez and Q. Zhang, An estimation of distribution algorithm with guided mutation for a complex flow shop scheduling problem. In *Proc. 9th Ann. Genetic and Evolutionary Comput. Conf. (GECCO'07)*, edited by D. Thierens. *ACM* (2007) 570–576.
[23] M. Sulaiman, A. Salhi, E.S. Fraga and W.K. Mashwani, M.M. Rashidi, A Novel Plant Propagation Algorithm: Modifications And Implementation. *Science International-Lahore* **28** (2015) 201–209.
[24] M. Sulaiman and A. Salhi, A Seed-based Plant Propagation Algorithm: The feeding Station Model. *Scientific World J.* **2015** (2015) 1–16.
[25] M. Sulaiman, A. Salhi, B. Selamoglu and O. Kirikchi, A plant propagation algorithm for constrained engineering optimisation problems. *Math. Problems Eng.* **2014** (2014) 1–10.
[26] R. Yang, Convergence of the simulated annealing algorithm for continuous global optimization. *J. Optimiz. Theory Appl.* 104 (2000) 691–716.
[27] X.-S. Yang, Firefly algorithm, stochastic test functions and design optimisation. *Inter. J. Bio-Inspired Comput.* **2** (2010) 78–84.
[28] X.-S. Yang, Nature-Inspired Metaheuristic Algorithms. Luniver Press (2011).
[29] X.-S. Yang, Flower pollination algorithm for global optimization. in *Unconventional Comput. Natural Comput.*, Springer (2012) 240–249.