Applied Mathematics & Information Sciences Letters

An International Journal

http://dx.doi.org/10.12785/amisl/030106

An Improved Flower Pollination Algorithm for Solving Integer Programming Problems

*Khalil AL-Wagih**

Faculty of Computer science & Information System, Thamar University, Thamar, Republic of Yemen
*E-mail: khalilwagih@gmail.com

Received: 8 Mar. 2014, Revised: 16 Dec. 2015, Accepted: 19 Dec. 2015

Published online: 1 Jan. 2015

Abstract: Flower pollination algorithm is a new nature-inspired algorithm, based on the characteristics of flowering plants. In this paper, a new method is developed based on the flower pollination algorithm combined with chaos theory (IFPCH) to solve integer programming problems. IFPCH rounds the parameter values to the closest integer after producing new solutions. Numerical simulation results show that the algorithm proved to be superior in almost all tested problems.

Keywords: Flower pollination algorithm; meta-heuristics; optimization; chaos; integer programming.

1 Introduction

The real world optimization problems are often very challenging to solve, and many applications have to deal with NP-hard problems [1]. To solve such problems, optimization tools have to be used even though there is no guarantee that the optimal solution can be obtained. In fact, for NP problems, there are no efficient algorithms at all. As a result of this, many problems have to be solved by trial and errors using various optimization techniques [2]. In addition, new algorithms have been developed to see if they can cope with these challenging optimization problems. Among these new algorithms, many algorithms such as particle swarm optimization, cuckoo search and firefly algorithm, have gained popularity due to their high efficiency. In this paper, we have used IFPCH algorithm for solving integer programming problems. Integer programming is NP-hard problems [3-10]. The name "linear integer programming "is referred" to the class of combinatorial constrained optimization problems with integer variables, where the objective function is a linear function and the constraints are linear inequalities." The Linear Integer Programming (also known as LIP) optimization problem can be stated in the following general form:

$$\text{Max } cx \quad (1)$$

$$\text{s.t.} Ax \leq b, \quad (2)$$

$$x \in Z^n \quad (3)$$

where the solution $x \in Z^n$ is a vector of n integer variables: $x = (x_1, x_2, \ldots, x_n)^T$ and the data are rational and are given by the m×n matrix A, the 1×n matrix c, and the m×1 matrix b. This formulation includes also equality constraints, because each equality constraint can be represented by means of two inequality constraints like those included in eq. (2).

Integer programming addresses the problem raised by non-integer solutions in situations where integer values are required. Indeed, some applications do allow a continuous solution. For instance, if the objective is to find the amount of money to be invested or the length of cables to be used, other problems preclude it: the solution must be discrete [3]. Another example, if we are considering the production of jet aircraft and x_1 = 8.2 jet airliners, rounding off could affect the profit or the cost by millions of dollars. In this case we

need to solve the problem so that an optimal integer solution is guaranteed.

The possibility to obtain integer values is offered by integer programming: as a pure integer linear programming, in which all the variables must assume an integer value, or as a mixed-integer linear programming which allows some variables to be continuous, or a 0-1 integer model, all the decision variables have integer values of zero or one[8-9].

A wide variety of real life problems in logistics, economics, social sciences and politics can be formulated as linear integer optimization problems. The combinatorial problems, like the knapsack-capital budgeting problem, warehouse location problem, travelling salesman problem, decreasing costs and machinery selection problem, network and graph problems, such as maximum flow problems, set covering problems, matching problems, weighted matching problems, spanning tree problems and many scheduling problems can also be solved as linear integer optimization problems [8-9].

Exact integer programming techniques such as cutting plane techniques [8-9]. The branch and the bound both have high computational cost, in large-scale problems [8-9]. The branch and the bound algorithms have many advantages over the algorithms that only use cutting planes. One example of these advantages is that the algorithms can be removed early as long as at least one integral solution has been found and an attainable solution can be returned although it is not necessarily optimal. Moreover, the solutions of the LP relaxations can be used to provide a worst-case estimate of how far from optimality the returned solution is. Finally, the branch method and the bound method can be used to return multiple optimal solutions.

Since integer linear programming is NP-complete, for that reason many problems are intractable. So instead of the integer linear programming, the heuristic methods must be used. For example, Swarm intelligence metaheuristics, amongst which an ant colony optimization, artificial bee colony optimization particle swarm optimization [8-9]. Also Evolutionary algorithms, differential evolution and tabu search were successfully applied into solving integer programming problems [8-9]. Heuristics typically have polynomial computational complexity, but they do not guarantee that the optimal solution will be captured. In order to solve integer programming problems, most of the heuristics truncate or round the real valued solutions to the nearest integer values. In this paper, Bat algorithm is applied to integer programming problems and the performance was compared with other harmony search algorithms [9].

Flower pollination is an intriguing process in the natural world. Its evolutionary characteristics can be used to design new optimization algorithms. The algorithm obtained good results were dealing with lower-dimensional optimization problems, but may become problematic for higher-dimensional problems because of its tendency to converge very fast initially. This paper introduced an improved Flower pollination algorithm by integrating it with chaos to improve the reliability of the global optimality, and also enhances the quality of the results.

This paper is organized as follows: after introduction, the original Flower pollination algorithm is briefly introduced. Than in section 3 introduces the meaning of chaos. In section 4, the proposed algorithm is described, while the results are discussed in section 5. Finally, conclusions are presented in section 6.

2 The Original Flower Pollination Algorithm

Flower Pollination Algorithm (FPA) was founded by Yang in the year 2012. Inspired by the flow pollination process of flowering plants are the following rules:

Rule 1: Biotic and cross-pollination can be considered as a process of global pollination process, and pollen-carrying pollinators move in a way that obeys Le'vy flights.
Rule 2: For local pollination, a biotic and self-pollination are used.
Rule 3: Pollinators such as insects can develop flower constancy, which is equivalent to a reproduction probability that is proportional to the similarity of two flowers involved.
Rule 4: The interaction or switching of local pollination and global pollination can

be controlled by a switch probability $p \in [0,1]$, with a slight bias toward local pollination .

In order to formulate updating formulas, we have to convert the aforementioned rules into updating equations. For example, in the global pollination step, flower pollen gametes are carried by pollinators such as insects, and pollen can travel over a long distance because insects can often fly and move in a much longer range[10].Therefore, Rule 1 and flower constancy can be represented mathematically as:

$$x_i^{t+1} = x_i^t + \gamma L(\lambda)(x_i^t - B) \quad (1)$$

Where x_i^t is the pollen i or solution vector x_i at iteration t, and B is the current best solution found among all solutions at the current generation/iteration. Here γ is a scaling factor to control the step size. In addition, $L(\lambda)$ is the parameter that corresponds to the strength of the pollination, which essentially is also the step size. Since insects may move over a long distance with various distance steps, we can use a Le'vy flight to imitate this characteristic efficiently. That is, we draw $L > 0$ from a Levy distribution:

$$L \sim \frac{\lambda \Gamma(\lambda) \sin(\pi\lambda/2)}{\pi} \frac{1}{S^{1+\lambda}}, (S >> S_0 > 0) \quad (2)$$

Here, $\Gamma(\lambda)$ is the standard gamma function, and this distribution is valid for large steps $s > 0$.
Then, to model the local pollination, both Rule 2 and Rule 3 can be represented as

$$x_i^{t+1} = x_i^t + U(x_j^t - x_k^t) \quad (3)$$

Where x_j^t and x_k^t are pollen from different flowers of the same plant species. This essentially imitates the flower constancy in a limited neighborhood. Mathematically, if x_j^t and x_k^t comes from the same species or selected from the same population, this equivalently becomes a local random walk if we draw U from a uniform distribution in [0, 1].Though Flower pollination activities can occur at all scales, both local and global, adjacent flower patches or flowers in the not-so-far-away neighborhood are more likely to be pollinated by local flower pollen than those faraway. In order to imitate this, we can effectively use the switch probability like in Rule 4 or the proximity probability p to switch between common global pollination to intensive local pollination. To begin with, we can use a naive value of $p = 0.5$as an initially value. A preliminary parametric showed that $p = 0.8$ might work better for most applications [10].
The basic steps of FP can be summarized as the pseudo-code shown in Figure 1.

Flower pollination algorithm
Define Objective function $f(x)$, $x = (x_1, x_2, ..., x_d)$
Initialize a population of n flowers/pollen gametes with random solutions
*Find the best solution **B**in the initial population*
Define a switch probability $p \in [0, 1]$
Define a stopping criterion (either a fixed number of generations/iterations or accuracy)
while *(t <MaxGeneration)*
for *i= 1 : n (all n flowers in the population)*
if *rand <p,*
Draw a (d-dimensional) step vector L which obeys a L´evy distribution
Global pollination via $x_i^{t+1} = x_i^t + L(B - x_i^t)$
else
Draw U from a uniform distribution in [0,1]
Do local pollination via $x_i^{t+1} = x_i^t + U(x_j^t - x_k^t)$
end if
Evaluate new solutions
If new solutions are better, update them in the population
end for
*Find the current best solution **B***
end while
Output the best solution found

Fig. 1 Pseudo code of the Flower pollination algorithm

3 Chaos Theory

Generating random sequences with a longer period and good consistency is very important for easily simulating complex phenomena, sampling, numerical analysis, decision making and especially in heuristic optimization [11]. Its quality determines the reduction of storage and computation time to achieve a desired accuracy [12]. Chaos is a deterministic, random-like process found in nonlinear, dynamical system, which is non-period, non-converging and bounded. Moreover, it depends on its initial condition and parameters [12-19]. Applications of chaos in several disciplines including operations research, physics, engineering, economics, biology, philosophy and computer science [19-22].

Recently chaos has been extended to various optimization areas because it can more easily escape from local minima and improve global convergence in comparison with other stochastic optimization algorithms [22-25]. Using chaotic

sequences in flower pollination algorithm can be helpful to improve the reliability of the global optimality, and they also enhance the quality of the results**.**

3.1 Chaotic maps

At random-based optimization algorithms, the methods using chaotic variables instead of random variables are called chaotic optimization algorithms (COA) [12]. In these algorithms, due to the non-repetition and ergodicity of chaos, it can carry out overall searches at higher speeds than stochastic searches that depend on probabilities [25-29]. To achieve this issue, herein one-dimensional, non-invertible maps are utilized to generate chaotic sets. We will illustrate some of well-known one-dimensional maps as:

3.1.1 Logistic map

The Logistic map is defined by:

$$Y_{n+1} = \mu Y_n(1 - Y_n) Y \in (0,1)\ 0 < \mu \leq 4 \quad (4)$$

3.1.2 The Sine map

The Sine map is written as the following equation:

$$Y_{n+1} = \frac{\mu}{4}\sin(\pi Y_n)\ Y \in (0,1)\ 0 < \mu \leq 4 \quad (5)$$

3.1.3 Iterative chaotic map

The iterative chaotic map with infinite collapses is described as:

$$Y_{n+1} = \sin\left(\frac{\mu\pi}{Y_n}\right) \mu \in (0,1) \quad (6)$$

3.1.4 Circle map

The Circle map is expressed as:

$$Y_{n+1} = Y_n + \alpha - \left(\frac{\beta}{2\pi}\right)\sin(2\pi Y_n)\ mod\ 1 \quad (7)$$

3.1.5 Chebyshev map

The family of Chebyshev map is written as the following equation:

$$Y_{n+1} = \cos(k\cos^{-1}(Y_n))\ Y \in (-1,1) \quad (8)$$

3.1.6 Sinusoidal map

This map can be represented by

$$Y_{n+1} = \mu Y_k^2 \sin(\pi Y_n) \quad (9)$$

3.1.7 Gauss map

The Gauss map is represented by:

$$Y_{n+1} = \begin{cases} 0 & Y_n = 0 \\ \frac{\mu}{Y_n} mod\ 1 & Y_n \neq 0 \end{cases} \quad (10)$$

3.1.8 Sinus map

Sinus map is formulated as follows:

$$Y_{n+1} = 2.3(Y_n)^{2\sin(\pi Y_n)} \quad (11)$$

3.1.9 Dyadic map

Also known as the dyadic map, bit shift map, 2*x* mod 1 map, Bernoulli map, doubling map or saw tooth map. Dyadic map can be formulated by a mod function:

$$Y_{n+1} = 2Y_n mod\ 1 \quad (12)$$

3.1.10 Singer map

Singer map can be written as:

$$Y_{n+1} = \mu(7.86Y_n - 23.31Y_n^2 + 28.75Y_n^3 - 13.3Y_n^4) \quad (13)$$

μbetween 0.9 and 1.08

3.1.11 Tent map

This map can be defined by the following equation:

$$Y_{n+1} = \begin{cases} \mu Y_n & Y_n < 0.5 \\ \mu(1 - Y_n) & Y_n \geq 0.5 \end{cases} \quad (14)$$

4 The Proposed Algorithm (IFPCH) for Solving Definite Integral

In the proposed chaotic Flower pollination algorithm, we used chaotic maps to tune the Flower pollination algorithm parameter and improve the performance [11-12]. The steps of the proposed chaotic Flower pollination algorithm for solving integer programming are as follows:

Step 1 define the objective function and initializes a population and find the best solution *B* in the initial population.
Step 2 Calculate *p* by the selected chaotic maps.
Step 3*If* (*rand* <*p*) then global pollination via
$x_i^{t+1} = x_i^t + (f\gamma)L(\lambda)(x_i^t - B)$
else do local pollination via selected chaotic map.
Step 4 Evaluate new solutions if better, update them in the population.
Step 5 Find the current best solution *B*.
Step 6 Output the best solution found.

5 Numerical Results

In this section, we will carry out numerical simulation based on some well-known unconstrained optimization problems to investigate the performances of the proposed algorithm. The best results obtained by IFPCH for test problems (1–7) are presented in Table 1. In these problems, the initial parameters are set at *n*= 50 and the number of iterations is set to *t* = 1000.The selected

chaotic map for all problems is the logistic map, according to the following equation:

$$Y_{n+1} = \mu Y_n(1 - Y_n(15)$$

Clearly, $Y_n \in [0,1]$ under the conditions that the initial $Y_0 \in [0,1]$, where n is the iteration number and $\mu = 4$.The results of IFPCH algorithm are conducted from 30 independent runs for each problem. The comparison between the results determined by the proposed approach and the standard flower pollination algorithm are reported in Table 1. The results have demonstrated the superiority of the proposed approach to finding the optimal solution.

All the experiments were performed on a Windows 7 Ultimate 64-bit operating system; processor Intel Core i5 760 running at 2.81 GHz; 6 GB of RAM and code was implemented in C#.

5.1. Test problem 1

The problem formulation is:

$P_1(x) = \|x\|_1 = |x_1| + |x_2| + ..|x_D|$, Where D is the dimension and $x \in [-100,100]^D$. The global minimum = 0.

5.2. Test problem 2

The mathematical formulation of the optimization problem can be stated as follows:

$P_2(x) = x^T.x = [x_1 ... x_D] \begin{bmatrix} x_1 \\ . \\ . \\ . \\ x_D \end{bmatrix}$, where D is the dimension and $x \in [-100,100]^D$. The global minimum = 0.

5.3. Test problem 3

The problem can be stated as follows:

$P_3(x) = (9.x_1^2 + 2.x_2^2 - 11)^2 + (3.x_1 + 4.x_2^2 - 7)^2$.

The global minimum = 0.

5.4. Test problem 4

The problem formulation is:

$P_4(x) = (x_1 + 10x_2)^2 + 5(x_3 - x_4)^2 + 5(x_2 - 2x_3)^4 + 10(x_1 - x_4)^4$

The global minimum = 0.

5.5. Test problem 5

The problem can be expressed as:

$P_5(x) = (x_1^2 + x_2 - 11)^2 + (x1 + x_2^2 - 7)^2$.

The global minimum = 0.

5.6. Test problem 6

The problem formulation is:

$P_6(x) = -3803.84 - 138.08x_1 - 232.92x_2 + 123.08x_1^2 + 203.64x_2^2 + 182.25x_1x_2$

.

The global minimum = 0.

5.7. Test problem 7

The problem can be stated as follows:

$P_7(x) = 100(x_2 - x_1^2)^2 + (1 - x_1)^2$.

The global minimum = 0.

Table 1 the best solution of proposed algorithm and FP algorithm for solving integer programming problems

Test problem	Dimension	Standard FP Algorithm			IFPCH Algorithm		
		Success Rate	Mean of Iteration Numbers	Standard Deviation of Iteration Numbers	Success Rate	Mean of Iteration Numbers	Standard Deviation of Iteration Numbers
P1	15	17/30	97.18	3.68	29/30	51.08	10.00
	20	0/30	100.00	0.00	29/30	51.94	9.82
	25	0/30	100.00	0.00	28/30	53.2	11.83
P2	10	28/30	78.04	0.00	29/30	48.04	7.49
	15	0/30	100.00	0.00	29/30	50.00	7.19
	20	0/30	100.00	0.00	29/30	52.84	11.92
P3	2	30/30	8.74	3.25	30/30	4.34	4.45
P4	4	19/30	95.56	8.02	30/30	32.76	4.45
P5	2	30/30	11.4	4.52	30/30	8.28	4.45

P6	2	30/30	9.72	4.20	30/30	8.80	3.82
P7	2	25/30	26.42	21.11	30/30	8.68	3.88

6. Conclusions

This paper introduced an improved Flower pollination algorithm by blending with chaos for solving integer programming problems. Several problems have been used to prove the effectiveness of the proposed method. IFPCH algorithm is superior to standard FP in terms of both efficiency and success rate. This implies that IFPCH is potentially more powerful in solving NP-hard problems.

The reason for getting better results than standard flower pollination algorithm, using chaos helps the algorithms to escape from local solutions. Table 1 shows the results of IFPCH algorithm are privileged compared with the results of Standard flower pollination algorithm.

References

[1] L. A. Wolsey, "Integer programming," *IIE Transactions*, vol. 32, pp. 2-58, 2000.

[2] G. B. Dantzig, *Linear programming and extensions*: Princeton university press, 1998.

[3] G. L. Nemhauser and L. A. Wolsey, *Integer and combinatorial optimization* vol. 18: Wiley New York, 1988.

[4] E. Beale, "Integer programming," in *Computational Mathematical Programming*, ed: Springer, 1985, pp. 1-24.

[5] C. H. Papadimitriou and K. Steiglitz, *Combinatorial optimization: algorithms and complexity*: Courier Dover Publications, 1998.

[6] H. Williams, "Logic and Integer Programming, International Series in Operations Research & Management Science," ed: Springer, 2009.

[7] A. Schrijver, *Theory of linear and integer programming*: Wiley. com, 1998.

[8] O. Abdel-Raouf, M. Abdel-Baset, and I. El-henawy."An Improved Flower Pollination Algorithm with Chaos." 2014.

[9] O. Abdel-Raouf, M.Abdel-Baset, and I. El-Henawy. "An Improved Chaotic Bat Algorithm for Solving Integer Programming Problems." 2014

[10] X-S. Yang," Flower pollination algorithm for global optimization", *Unconventional Computation and Natural Computation*, Lecture Notes inComputer Science, Vol. 7445, pp. 240-249,2012.

[11] L. M. Pecora and T. L. Carroll, "Synchronization in chaotic systems," *Physical review letters*, vol. 64, pp. 821-824, 1990.

[12] D. Yang, G. Li, and G. Cheng, "On the efficiency of chaos optimization algorithms for global optimization," *Chaos, Solitons & Fractals*, vol. 34, pp. 1366-1375, 2007.

[13] A. H. Gandomi, G. J. Yun, X.-S. Yang, and S. Talatahari, "Chaos-enhanced accelerated particle swarm optimization," *Communications in Nonlinear Science and Numerical Simulation*, 2012.

[14] B. Alatas, "Chaotic harmony search algorithms," *Applied Mathematics and Computation*, vol. 216, pp. 2687-2699, 2010.

[15] W. Gong and S. Wang, "Chaos Ant Colony Optimization and Application," in *Internet Computing for Science and Engineering (ICICSE), 2009 Fourth International Conference on*, 2009, pp. 301-303.

[16] B. Alatas, "Chaotic bee colony algorithms for global numerical optimization," *Expert Systems with Applications*, vol. 37, pp. 5682-5687, 2010.

[17] A. Gandomi, X.-S. Yang, S. Talatahari, and A. Alavi, "Firefly algorithm with chaos," *Communications in Nonlinear Science and Numerical Simulation*, vol. 18, pp. 89-98, 2013.

[18] J. Mingjun and T. Huanwen, "Application of chaos in simulated annealing," *Chaos, Solitons & Fractals*, vol. 21, pp. 933-941, 2004.

[19] L. d. S. Coelho and V. C. Mariani, "Use of chaotic sequences in a biologically inspired algorithm for engineering design optimization," *Expert Systems with*

Applications, vol. 34, pp. 1905-1913, 2008.

[20] M. S. Tavazoei and M. Haeri, "Comparison of different one-dimensional maps as chaotic search pattern in chaos optimization algorithms," *Applied Mathematics and Computation*, vol. 187, pp. 1076-1085, 2007.

[21] R. Hilborn, "Chaos and nonlinear dynamics, 1994," ed: Oxford University Press, New York.

[22] D. He, C. He, L.-G. Jiang, H.-W. Zhu, and G.-R. Hu, "Chaotic characteristics of a one-dimensional iterative map with infinite collapses," *Circuits and Systems I: Fundamental Theory and Applications, IEEE Transactions on*, vol. 48, pp. 900-906, 2001.

[23] A. Erramilli, R. Singh, and P. Pruthi, *Modeling packet traffic with chaotic maps*: Citeseer, 1994.

[24] R. M. May, "Simple mathematical models with very complicated dynamics," *Nature*, vol. 261, pp. 459-467, 1976.

[25] A. Wolf, "Quantifying chaos with Lyapunov exponents," *Chaos*, pp. 273-290, 1986.

[26] R. L. Devaney, "An introduction to chaotic dynamical systems," 2003.

[27] R. Barton, "Chaos and fractals," *The Mathematics Teacher*, vol. 83, pp. 524-529, 1990.

[28] E. Ott, *Chaos in dynamical systems*: Cambridge university press, 2002.

[29] C. Letellier, *Chaos in nature* vol. 81: World Scientific Publishing Company, 2013.