International Journal of Computer Applications Technology and Research
Volume 4– Issue 3, 197 - 201, 2015, ISSN:- 2319–8656

A Comparison between FPPSO and B&B Algorithm for Solving Integer Programming Problems

Mahmoud Ismail
Department of Operations Research
Faculty of Computers and Informatics
Zagazig University
El-Zera Square, Zagazig, Sharqiyah
Egypt

Ibrahim El-Henawy
Department of Computer science
Faculty of Computers and Informatics,
Zagazig University
El-Zera Square, Zagazig, Sharqiyah
Egypt

Abstract: Branch and Bound technique (B&B) is commonly used for intelligent search in finding a set of integer solutions within a space of interest. The corresponding binary tree structure provides a natural parallelism allowing concurrent evaluation of sub-problems using parallel computing technology. Flower pollination Algorithm is a recently-developed method in the field of computational intelligence. In this paper is presented an improved version of Flower pollination Meta-heuristic Algorithm, (FPPSO), for solving integer programming problems. The proposed algorithm combines the standard flower pollination algorithm (FP) with the particle swarm optimization (PSO) algorithm to improve the searching accuracy. Numerical results show that the FPPSO is able to obtain the optimal results in comparison to traditional methods (branch and bound) and other harmony search algorithms. However, the benefits of this proposed algorithm is in its ability to obtain the optimal solution within less computation, which save time in comparison with the branch and bound algorithm.

Keywords: Branch and bound, flower pollination Algorithm; meta-heuristics; optimization; the particle swarm optimization; integer programming.

1. INTRODUCTION

We ask that authors follow some simple guidelines. This document is a template. An electronic copy can be downloaded from the journal website. For questions on paper guidelines, please contact the conference publications committee as indicated on the conference website. Information about final paper submission is available from the conference website

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 IBACH 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:

$$\text{Max } cx \quad (1)$$
$$\text{s.t. } Ax \leq b, \quad (2)$$
$$x \in Zn \quad (3)$$

where the solution $x \in Zn$ is a vector of n integer variables: $x = (x1, x2, \ldots, xn)T$ and the data are rational and are given by the $m \times n$ matrix A, the $1 \times n$ matrix c, and the $m \times 1$ 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 $x1 = 8.2$ 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[10].

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 trees problems and many scheduling problems can also be solved as linear integer optimization problems [11-14].

2. BRANCH AND BOUND

The branch and bound is the divide and conquer method. We divide a large problem into a few smaller ones. (This is the "branch" part). The conquering part is done by estimate how good a solution we can get for each smaller problems (to do this, we may have to divide the problem further, until we get a problem that we can handle), that is the "bound" part. The branch and bound algorithm is able to be parallelized by distributing computation of subproblems on multiple computing nodes. Parallel branch and bound algorithms with the master-worker algorithm, where a single master process dispatches tasks to multiple worker processes, have been proposed in many literatures [3]. In master-worker algorithm, a single master process dispatches subproblems, which correspond to leaf nodes on the search tree, to multiple worker processes and receives the computed results from the worker processes. The computed results contain the best upper bound of the objective function, and subproblems that have generated by branching and have not been pruned on a worker process. Also, the parallel algorithm with the hierarchical master-worker paradigm is proposed to improve performance on large-scale computing environment.

Exact integer programming techniques such as cutting plane techniques [15-17]. The branch and the bound both have high computational cost, in large-scale problems [18-19]. 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 [20-24].Also Evolutionary algorithms, differential evolution and tabu search were successfully applied into solving integer programming problems [25-27]. 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, an improved version of flower pollination algorithm is applied to integer programming problems and the performance was compared with other harmony search algorithms.

This paper is organized as follows: after introduction, the original branch and bound algorithm is introduced in section 2. The flower pollination algorithm is briefly introduced in section 3. Section 4 introduces the particle swarm optimization algorithm. Section 5 introduces the meaning of chaos. While the results are discussed in section 6. Finally, conclusions are presented in section 7.

3. 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 [28]:

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∈[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 [56].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 xi 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(λ) 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, Γ(λ) 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 = 0.5as an initially value. A preliminary parametric showed that p = 0.8 might work better for most applications [28].

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 = (x1, x2, ..., xd)
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 ∈ [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

4. PARTICLE SWARM OPTIMIZATION

Particle swarm optimization (PSO) was developed by Kennedy and Eberhartin 1995 based on the swarm behavior such as fish and bird schooling in nature [29]. Since then, PSO has generated much wider interests and forms an exciting, ever expanding research subject called swarm intelligence. This algorithm searches the space of an objective function by adjusting the trajectories of individual agents, called particles, as the piecewise paths formed by positional vectors in a quasi stochastic manner. The movement of a swarming particle consists of two major components: a stochastic component and a deterministic component. Each particle is attracted toward the position of the current global best g and its own best location xi* in history, while at the same time it has a tendency to move randomly. Let xi and vi be the position vector and velocity of particle i, respectively. The new velocity vector is determined by the following formula:

$$v_i^{t+1} = v_v^t + c_1 r_1 (g - x_i^t) + c_2 r_2 (x_x^* - x_i^t) \quad (4)$$

Wherer1 and r2are two random vectors and each entry takes the values between 0and 1. The parameters c1and c2are the learning parameters or acceleration constants, which can typically be taken as, say, c1≈c2≈ 2.The initial locations of all particles should be distributed relatively uniformly so that they can sample over most regions, which is especially important for multimodal problems. The initial velocity of a particle can be taken as zero, i.e. vit=0 =0. The new positions can then be updated by:

$$x_i^{t+1} = x_i^t + v_i^{t+1} \quad (5)$$

Although vi can be any value, it is usually bounded in some range [0,vmax].

5. CHAOS

Generating random sequences with longer periods and good consistency is very important for easily simulating complex phenomena, sampling, numerical analysis, decision making and especially in heuristic optimization [30]. Its quality determines the reduction of storage and computation time to achieve a desired accuracy [31]. Chaos is a deterministic, random-like process found in a nonlinear, dynamical system, which is non-period, non-converging and non-bounded. Moreover, it depends on its initial condition and parameters [32-34]. Applications of chaos has several disciplines including operations research, physics, engineering, economics, biology, philosophy and computer science [35-37].

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 [34-38]. 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.

5.1 Chaotic Maps

At random-based optimization algorithms, the methods using chaotic variables instead of random variables are called chaotic optimization algorithms (COA) [34]. 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 [43-48]. To resolve this issue, herein one-dimensional and non-invertible maps are utilized to generate chaotic sets. We will illustrate some of well-known one-dimensional maps as:

5.1.1 The Logistic map

The Logistic map is defined by:

$$Y_{n+1} = \mu Y_n (1 - Y_n)\ Y \in (0,1)\ 0 < \mu \leq 4 \quad (6)$$

5.1.2 The Sine map

The Sine map is written as the following equation:

$$Y_{n+1} = \frac{\mu}{4} \sin(\pi Y_n)\ Y \epsilon\ (0,1)\ 0 < \mu \leq 4 \quad (7)$$

5.1.3 The iterative chaotic map

The iterative chaotic map with infinite collapses is described as:

$$Y_{n+1} = \sin\left(\frac{\mu\pi}{Y_n}\right)\ \mu \in (0,1) \quad (8)$$

5.1.4 The Circle map

The Circle map is expressed as:

$$Y_{n+1} = Y_n + \alpha - \left(\frac{\beta}{2\pi}\right) \sin(2\pi Y_n)\ mod\ 1 \quad (9)$$

5.1.5 The Chebyshev map

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

$$Y_{n+1} = \cos(k cos^{-1}(Y_n))\ \ Y \in (-1,1) \quad (10)$$

5.1.6 The Sinusoidal map

Sinusoidal map can be represented by

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

5.1.7 The Gauss map

The Gauss map is represented by:

$$Y_{n+1} = \begin{cases} 0 & Y_n = 0 \\ \frac{\mu}{Y_n} \ mod\ 1 & Y_n \neq 0 \end{cases} \quad (12)$$

5.1.8 The Sinus map

Sinus map is formulated as follows:

$$Y_{n+1} = 2.3(Y_n)^{2\sin(\pi Y_n)} \quad (13)$$

5.1.9 The Dyadic map

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

$$Y_{n+1} = 2Y_n \ mod\ 1 \quad (14)$$

5.1.10 The Singer map

Singer map can be written as:

$$Y_{n+1} = \mu(7.86Y_n - 23.31Y_n^2 + 28.75Y_n^3 - 13.3Y_n^4) \quad (15)$$

μ between 0.9 and 1.08

5.1.11 The Tent map

Tent map can be defined by the following equation:

$$Y_{n+1} = \begin{cases} \mu Y_n & Y_n < 0.5 \\ \mu(1 - Y_n) & Y_n \geq 0.5 \end{cases} \quad (16)$$

6. NUMERICAL RESULTS

Several examples have been done to verify the weight of the proposed algorithm. The initial parameters setting of the algorithms is as follows: HMS=50 and itermax=1000, HMCR = 0.9; PARmax = 1; PARmin =0.1; bwmax = 1; bwmin = 0.01. The results of FPPSO algorithm are conducted from 50 independent runs for each problem and measured according to the best values in these runs.The selected chaotic map for all examples is the Sinusoidal map, whose equation is shown below: $Y_{n+1} = \mu Y_k^2 \sin(\pi Y_n)$ (16)
Where n is the iteration number.

Table 1. Optimal solution of selected problems

Exact Method		**The Best Solution**		
No.of Variables	Optimal Solution	Optimal values	BB	FPPSO
2	55	X_i=(4,3)	55	**55**
3	26	X_i=(2,1,6)	22	**26**
5	9	X_i=(1,1,0,0,0)	7	**9**
10	9	X_i=(0,2,0,2,3,1,0,0,2,3)	7	**9**
20	16	X_i=(0,0,0,0,0,0,0,0,1,4,0,4,3,0,2,4,0,3,0)	12	**16**
30	446	X_i=(0,0,0,0,0,0,0,0,0,16,20,4,4,0,3,0,0,0,24,3,0,0,0,0,0,4,0,1,0,8)	401	**446**

Table 1 shows the results of FPPSO algorithm are privileged compared with the results of Branch and bound (B&B). In comparison with exact values we find that the results of FPPSO algorithm are very close to the exact values of selected problems under the study. If a large number of variables are to be found, then it is hard to go past the classical methods. More usually, though, users will choose to use the proposed algorithm, to save their own time and to gain reliability. for example when we solved test problem number 6 by proposed algorithm it took time 7 seconds ,but when we solved it by branch and bound(exact method) it took time 396 seconds .

The reason for getting better results than the other algorithm considered is that the search power of FP algorithm. Adding to this, using PSO algorithm improves the performance of the algorithm.

7. CONCLUSIONS

This paper has introduced an improved flower pollination Algorithm by blending with practical swarm optimization algorithm for solving integer programming problems. Several examples have been used to prove the effectiveness of FPPSO. FPPSO algorithm managed to solve a large scale of problems that traditional method could not solve due to exponential growth in time and space complexities. The solution procedure will not face the same time waste in going through non-converging iterations as traditional methods do. FPPSO algorithm is superior to B&B in terms of both efficiency and success rate. This implies that FPPSO is potentially more powerful in solving NP-hard problems.
who have contributed towards development of the template.

8. 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] D. Bertsimas and R. Weismantel, Optimization over integers vol. 13: Dynamic Ideas Belmont, 2005.

[9] J. K. Karlof, *Integer programming: theory and practice*: CRC Press, 2005.

[10] M. Jünger, T. Liebling, D. Naddef, G. Nemhauser, W. Pulleyblank, G. Reinelt, *et al.*, *50 Years of Integer Programming 1958–2008*: Springer, Berlin, 2010.

[11] D.-S. Chen, R. G. Batson, and Y. Dang, *Applied integer programming: modeling and solution*: Wiley. com, 2011.

[12] K. L. Hoffman and M. Padberg, "Solving airline crew scheduling problems by branch-and-cut," *Management Science*, vol. 39, pp. 657-682, 1993.

[13] J. D. Little, K. G. Murty, D. W. Sweeney, and C. Karel, "An algorithm for the traveling salesman problem," *Operations research*, vol. 11, pp. 972-989, 1963.

[14] M. Grotschel and L. Lovász, "Combinatorial optimization," *Handbook of combinatorics*, vol. 2, pp. 1541-1597, 1995.

[15] R. E. Gomory, "Outline of an algorithm for integer solutions to linear programs," *Bulletin of the American Mathematical Society*, vol. 64, pp. 275-278, 1958.

[16] R. E. Gomory, "An algorithm for integer solutions to linear programs," *Recent advances in mathematical programming*, vol. 64, pp. 260-302, 1963.

[17] R. E. Gomory, "Early integer programming," *Operations Research*, pp. 78-81, 2002.

[18] J. Tomlin, "Branch and bound methods for integer and non-convex programming," *Integer and Nonlinear Programming, American Elsevier Publishing Company, New York*, pp. 437-450, 1970.

[19] S. Rouillon, G. Desaulniers, and F. Soumis, "An extended branch-and-bound method for locomotive assignment," *Transportation Research Part B: Methodological*, vol. 40, pp. 404-423, 2006.

[20] M. Tuba, "Swarm intelligence algorithms parameter tuning," in *Proceedings of the 6th WSEAS international conference on Computer Engineering and Applications, and Proceedings of the 2012 American conference on Applied Mathematics*, 2012, pp. 389-394.

[21] R. Jovanovic and M. Tuba, "An ant colony optimization algorithm with improved pheromone correction strategy for the minimum weight vertex cover problem," *Applied Soft Computing*, vol. 11, pp. 5360-5366, 2011.

[22] R. Jovanovic and M. Tuba, "Ant colony optimization algorithm with pheromone correction strategy for the minimum connected dominating set problem," *Computer Science and Information Systems*, vol. 10, pp. 133-149, 2013.

[23] B. Akay and D. Karaboga, "Solving integer programming problems by using artificial bee colony algorithm," in *AI* IA 2009: Emergent Perspectives in Artificial Intelligence*, ed: Springer, 2009, pp. 355-364.

[24] M. G. Omran, A. Engelbrecht, and A. Salman, "Barebones particle swarm for integer programming problems," in *Swarm Intelligence Symposium, 2007. SIS 2007. IEEE*, 2007, pp. 170-175.

[25] G. Rudolph, "An evolutionary algorithm for integer programming," in *Parallel Problem Solving from Nature—PPSN III*, ed: Springer, 1994, pp. 139-148.

[26] M. G. Omran and A. P. Engelbrecht, "Differential evolution for integer programming problems," in *Evolutionary Computation, 2007. CEC 2007. IEEE Congress on*, 2007, pp. 2237-2242.

[27] F. Glover, "Tabu search—part II," *ORSA Journal on computing*, vol. 2, pp. 4-32, 1990.

[28] X-S. Yang, Flower pollination algorithm for global optimization, Unconventional Computation, lecture Notes in Computer Science, Vol. 7445, pp. 240-249,2012.

[29] J. Kennedy and R. Eberhart, Particle swarm optimization, in Proceedings of IEEE International Conference on Neural Network, pp. 1942–1948, December 1995.

[30] O. Abdel-Raouf, , M.Abdel-Baset, and I. El-Henawy. "An Improved Chaotic Bat Algorithm for Solving Integer Programming Problems." *International Journal of Modern Education and Computer Science (IJMECS)* 6.8 (2014): 18.

[31] O. Abdel-Raouf, , M. Abdel-Baset, and I. El-henawy. "A New Hybrid Flower Pollination Algorithm for Solving Constrained Global Optimization Problems." *International Journal of Applied* 4.2 (2014): 1-13.

[32] O. Raouf, I. El-henawy, and M. Abdel-Baset. "A novel hybrid flower pollination algorithm with chaotic harmony search for solving sudoku puzzles." *International Journal of Modern Education and Computer Science* 3 (2014): 38-44.

[33] O. Abdel-Raouf, I. El-henawy and M. Abdel-Baset "chaotic Harmony Search Algorithm with Different Chaotic Maps for Solving Assignment Problems "International Journal of Computational Engineering & Management, Vol. 17, pp. 10-15 ,2014.

[34] O. Abdel-Raouf, I. El-henawy and M. Abdel-Baset. "Chaotic Firefly Algorithm for Solving Definite Integral", IJITCS, vol.6, no.6, pp.19-24, 2014.

[35] O. Abdel-Raouf, I. El-henawy and M. Abdel-Baset "Improved Harmony Search with Chaos for Solving Linear Assignment Problems", IJISA, vol.6, no.5, pp.55 61, 2014.

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

[37] I. El-henawy, , O. Abdel-Raouf, and M. Abdelbaset. "Improved harmony search algorithm with chaos for solving definite integral." *International Journal of Operational Research* 21.2 (2014): 252-261.