Employing Three Swarm Intelligent Algorithms for Solving Integer Fractional Programming Problems

Ibrahim M. Hezam, Osama Abdel Raouf

Abstract---This paper seeks for the integer optimal solution of the Fractional Programming Problem (IFPP) using three different Swarm Intelligence (SI) algorithms. The three algorithms are: Particle Swarm Optimization (PSO), Firefly Algorithm (FA), and Cuckoo Search (CS). The proposed approaches perform this by embedding the search space truncating the real values to the nearest integers inside feasible region. This method idea based on SI, rounds the real solutions after every step and after the final step. SI approach can get near optimal solutions in a reasonable time and effort. SI is an effective techniques for non-smooth functions. The numerical result and statistical analysis show that the proposed methods perform significantly better than previously used classical methods. A comparative study among the three SI algorithms on the selected benchmark examples was carried out. The study revealed an almost similarity in performance with a privilege in computation time and optimization results for cuckoo search.

Index Terms--Swarm Intelligence, Particle Swarm Optimization, firefly algorithm, cuckoo search, Fractional Programming, Integer Programming.

---

1. INTRODUCTION

Integer fractional programming plays an important part in the optimization modeling and real-world applications, require the variables to be optimized to be integers, such as fixed-charge problems, job-shop scheduling problems, including resource allocation, production scheduling, marketing, capital budgeting, assignment, transportation, and reliability networks. In order to solve this kind of problem, optimization techniques developed for real search spaces can be applied to integer programming problems and determine the optimal solution by rounding of the real optimum values to the nearest integers inside feasible region.

There are several studies in recent years used nontraditional methods to solve integer programming as: Laskari E.C., et al. in [1] (2002) proposed the Particle swarm Optimization for integer programming. Kitayama S., and Yasuda K., [2] (2006) proposed Particle Swarm Optimization for mixed integer programming. The proposed method is that discrete variables are treated by a penalty function, so that all variables can be handled as the continuous ones. And they proposed a new method of setting and updating the penalty parameter when discrete variables are treated in terms of the penalty function. Li Y., et al. in [3] (2007) used chaotic ant swarm to solve the problem of integer programming by embedding the search space Z into R and truncating the real values to the nearest integers. They introduced two novel methods based on the chaotic ant swarm, rounding the real solutions after every step and after the final step.

- *Ibrahim M. Hezam, Department of Mathematics& computer, Faculty of Education, Ibb University, Yemen. E-mail: Ibrahizam.math@gmail.com*
- *Osama Abdel Raouf Department of Operations Research and Decision Support, Faculty of Computers & Information, Minufiya University. E-mail: osamaabd@hotmail.com*

Matsui T., et al. in [4] (2008) develop a new particle swarm optimization method which was applied to discrete optimization problems by incorporating a new method for generating initial search points, the rounding of values obtained by the move scheme and the revision of move methods. Omran M. G. H, et al. in [5] (2007) evaluated the performance of two versions of the barebones PSO in solving Integer Programming problems. Matsui T., et al. in [4] (2008) develop a new particle swarm optimization method which was applied to discrete optimization problems by incorporating a new method for generating initial search points, the rounding of values obtained by the move scheme and the revision of move methods. Chen H., et al. in [6] (2009) developed particle swarm optimization based on genetic operators for nonlinear integer programming. The integer restriction of problems was transformed by coding solutions. According to optimal solutions of population and individual, the new particle was updated by crossover, mutation and selection operators. Wu P., et al. [7] (2010) proposed an effective global harmony search algorithm for integer programming problems. The effective global harmony search algorithm was proposed to solve integer programming problems. The effective global harmony search algorithm designs a novel location updating equation, which enables the improvised solution to move to the global best solution rapidly in each iteration. Random selection with a low probability was carried out for the improvised solution after updating location, for it can prevent the effect global harmony search algorithm from being trapped into the local optimum. Yang H., and Zhao S., [8] (2010), presented a novel hybrid approach for solving the Container Loading problem based on the combination of immune particle swarm optimization and Integer linear programming model.

Liu Y., and Ma L., [9] (2011) presented Bee Colony Foraging Algorithm for Integer Programming, The principle of the optimization algorithm was discussed and the implementation of the method was presented. Pal A., et al.[10] (2011) presented use of a particle swarm optimization (PSO) algorithm for solving integer and mixed integer optimization problems. Datta D., and Figueira J R.,[11] (2011) develop some integer coded versions of PSO for dealing with integer and discrete variables of a problem. Bacanin, N., et al. in [12] (2013) applied firefly algorithm to integer programming problems by rounded the parameter values to the closest integer after producing new solutions. Ibrahim M. H. and Osama A. [13] (2013) used particle swarm optimization approach for solving complex variable fractional programming problems.

However, all the above researches did not address cuckoo search for integer programming generally and did not apply it directly to handle the integer fractional programming problems.

The purpose of this research work is to investigate the solution the integer fractional programming problem using swarm intelligence. Section 2 introduces the formulation of the integer fractional programming problems. Particle swarm optimization algorithm is reviewed in section 3. Section4 review firefly algorithm. Cuckoo search algorithm introduced in section 5. Illustrative examples and discussing the results are presented in Section 6. Finally, Conclusions are presented in Section7.

2. INTEGER FRACTIONAL PROGRAMMING

The paper, consider the general fractional programming problem (FPP) as in the following form:

$$\min/\max \quad z(x_1,...,x_n)=\sum_{i=1}^{p}\frac{f_i(x)}{g_i(x)}$$

$$subject\ to \quad x\in S, x\geq 0$$

$$S=\left\{x\in R^n \middle| \begin{array}{l} h_k(x)\geq 0, \quad k=1,...,K; \\ m_j(x)=0, \quad j=1,...,J; \\ x_i^l \leq x_i \leq x_i^u,\ i=1,..,n. \end{array}\right\}$$

(1)

$$g_i(x)\neq 0,\ i=1,..,n.$$

$$x_1,...,x_n\in Z.$$

where $f_i(x)$, $g_i(x)$, are continuous functions, Z is the set of integers. S is compact.

Fractional programming of the form (1) arises in a natural way whenever rates such as the ratios (profit/revenue), (profit/time), (-waste of raw material/quantity of used raw material), are to be maximized often these problems are linear or at least concave-convex fractional programming [14].

There are many different direct algorithms to solve the fractional programming problem.

Classical algorithms contain two stages: stage 1: solve the FPP via Charnes–Cooper's transformation, Dinkelbach algorithms both types, Isbell Marlow method, Wolf parametric approach, Martos' Algorithm, Cambini–Martein's Algorithm, etc. For converting the fractional programming problems to non-fractional programming problems. Stage 2: use the branch and bound technique, cutting planes, the implicit enumerative method and other methods to solve the integer programming problems IPP.

However, when the dimension of IFPP is small, the classical algorithm such as the above methods is able to handle the problem and give the required solution. But, in large-scale problems, and nonlinear cases, this is not always efficient. It will cost much computing convergence time, which cannot satisfy the requirement of engineering.

3. PARTICLE SWARM OPTIMIZATION (PSO)[15–17]

Particle swarm optimization is a population based stochastic optimization technique developed by Eberhart and Kennedy in 1995, inspired by social behavior of bird flocking or fish schooling.

The characteristics of PSO can be represented as follows:

x_i^k The current position of the particle *i* at iteration *k;*

v_i^k The current velocity of the particle *i* at iteration *k;*

y_i^k The personal best position of the particle *i* at iteration *k;*

$\tilde{y}_i^k$ The neighborhood best position of the particle.

The velocity update step is specified for each dimension $j \in 1,\ldots,N_d$, hence, $v_{i,j}$ represents the *jth* element of the velocity vector of the *ith* particle. Thus, the velocity of particle *i* is updated using the following equation:

$$v_i^k(t+1)=round\left(wv_i^k(t)\right)+round\left(c_1r_1(t)\left(y_i(t)-x_i(t)\right)\right)$$

$$+round\left(c_2r_2(t)\left(\tilde{y}_i(t)-x_i(t)\right)\right)$$

(2)

where w is weighting function, $c_{1,2}$ are weighting coefficients, $r_{i,2}(t)$ are random numbers between 0 and 1.

The current position (searching point in the solution space) can be modified by the following equation:

$$x_i^{k+1}=x_i^k+v_i^{k+1} \qquad (3)$$

The detailed operation of particle swarm optimization is given below:

Step 1: Initialize parameters and population.

Step 2: Initialization. Randomly set the position and velocity of all particles, within pre-defined ranges. And on D dimensions in the feasible space (i.e.it satisfies all the constraints)

Step 3: Velocity Updating. At each iteration, velocities of all particles are updated according to equation (2)

After updating, v_i^k should be checked and maintained within a pre-specified range to avoid aggressive random walking.

Step 4: Position Updating. Assuming a unit time interval between successive iterations, the positions of all particles are updated according to equation (3).

After updating, x_i^k should be checked and limited within the allowed range.

Step 5: Memory updating. Update y_i^k and $\tilde{y}_i^k$ when condition is met.

$$y_i^k(t+1)=\begin{cases} y_i^k(t) & \text{if } f\left(x_i^k(t+1)\right)\geq f\left(y_i^k(t)\right) \\ x_i^k(t+1) & \text{if } f\left(x_i^k(t+1)\right)<f\left(y_i^k(t)\right) \end{cases}$$

where *f(x)* is the objective function subject to maximization.

Step 6: Termination Checking. Repeat Steps 2 to 4 until definite termination conditions are met, such as a pre-defined number of iterations or a failure to make progress for a fixed number of iterations.

4. FIREFLY ALGORITHM (FA) [16], [18–20]

The Firefly Algorithm was developed by Yang (2008) and it was based on the following idealized behavior of the flashing characteristics of fireflies:

• All fireflies are unisex so that one firefly is attracted to other fireflies regardless of their sex;

• Attractiveness is proportional to their brightness, thus for any two flashing fireflies, the less bright one will move towards the brighter one. The attractiveness is proportional to the brightness and they both decrease as their distance increases. If no one is brighter than a particular firefly, it moves randomly;

• The brightness or the light intensity of a firefly is affected or determined by the landscape of the objective function to be optimized.

The operation of the Firefly Algorithm is as follows:

Step 1: Define Objective function $f(x), x=(x_1,x_2,...,x_d)$ and Generate initial population of fireflies placed at random positions within the n-dimensional search space, x_i. Define the light absorption coefficient γ.

Step 2: Define the Light Intensity of each firefly, Li, as the value of the cost function for x_i.

Step 3: For each firefly, x_i, the light Intensity, L_i, is compared for every firefly x_j $j\in\{1,2,...,d\}$

Step 4: If L_i is less than L_j, then move firefly x_i towards x_j in n-dimensions. The value of attractiveness between flies varies relatively the distance *r* between them:

$$x_i^{t+1}=round\left(x_i^t+\beta_\circ e^{-\gamma r_{ij}^2}\left(x_j^t-x_i^t\right)+\alpha\varepsilon_i^t\right) \quad (4)$$

where $\beta_\circ$ is the attractiveness at *r=0* the second term is due to the attraction, while the third term is randomization with the vector of random variables εi being drawn from a Gaussian distribution. $\alpha\in[0,1]$. The distance between any two fireflies *i* and *j* at x_i *and* x_j can be regarded as the Cartesian distance $r_{ij}=\left\|x_i-x_j\right\|_2$ or the l_2-norm.

Step 5: Calculate the new values of the cost function for each fly, x_i, and update the Light Intensity, L_i.

Step 6: Rank the fireflies and determine the current 'best'.

Step 7: Repeat Steps 2 to 6 until definite termination conditions are met, such as a pre-defined number of iterations or a failure to make progress for a fixed number of iterations.

5. CUCKOO SEARCH ALGORITHM (CS) [21–26]

The Cuckoo search algorithm is a Meta heuristic search algorithm which has been proposed recently by Yang and Deb (2009) and it was based on the following idealized rules:

• Each cuckoo lays one egg at a time, and dumps it in a randomly chosen nest.

• The best nests with high quality of eggs (solutions) will carry over to the next generations.

• The number of available host nests is fixed, and a host can discover an alien egg with a probability $p_a\in[0,1]$. In this case, the host bird can either throw the egg away or abandon the nest so as to build a completely new nest in a new location.

Cuckoo search algorithm

Begin

Objective function $f(x), x=(x_1,x_2,...,x_d)^T$;

Initial a population of n host nests $x_i\ (i=1,2,...,d)$

while (t < MaxGeneration) or (stop criterion);

Get a cuckoo (say i) randomly

and generate a new solution by Lévy flights;

Evaluate its quality/fitness; F_i

Choose a nest among n (say *j*) randomly;

if ($F_i > F_j$),

Replace *j* by the new solution;

end

Abandon a fraction (Pa) of worse nests

[and build new ones at new locations via Lévy flights];

Keep the best solutions (or nests with quality solutions);

Rank the solutions and find the current best;

end while

Post process results and visualization;

End

when generating new solutions $x_i(t+1)$ for the *ith* cuckoo, the following Lévy flight is performed

$$x_i^{(t+1)}=round\left(x_i^{(t)}+\alpha\oplus Levy(\lambda)\right) \quad (5)$$

where $\alpha>0$ is the step size, which should be related to the scale of the problem of interest. The product $\oplus$ means entry-wise multiplications [26]. In this research work, we consider a Lévy flight in which the step-lengths are

distributed according to the following probability distribution

$$Levyu = t^{-\lambda}, 1 < \lambda \le 3$$

which has an infinite variance. Here the consecutive jumps/steps of a cuckoo essentially form a random walk process which obeys a power-law step length distribution with a heavy tail.

For integer programming, the SI can be used by embedding the search space Z into R and truncating the real values to the nearest integers after every step.
The only difference between a real-coded PSO, FA, and CS and an integer coded is the data-type after evolution according to (2), (3), (4), and (5) respectively.

6. ILLUSTRATIVE EXAMPLES WITH DISCUSSION

Five examples were collected from literature to demonstrate the efficiency and robustness of the proposed algorithms in solving fractional programming problems. The numerical results which are compared among the present algorithms are illustrated in Table 1. The algorithms have been implemented by MATLAB R2011. Where the functions are as the following:

$$f_1:\ \min\ z = 0.5 + \frac{\sin^2\left(x^2+y^2\right) - 0.5}{\left(1+0.001\left(x^2+y^2\right)\right)^2}$$

$$subject\ to\quad -100 \le x, y \le 100, where\ x, y\ are\ integer$$

$$f_2:\ \min\ z = 0.5 + \frac{\cos\left(\sin\left|x^2+y^2\right|\right) - 0.5}{\left(1+0.001\left(x^2+y^2\right)\right)^2}$$

$$subject\ to\quad -100 \le x, y \le 100, where\ x, y\ are\ integer$$

$$f_3:\ \min\ z = \frac{6x+6y}{11x+y}$$

$$subject\ to\quad 0 \le x \le 4;\quad 0 \le 2y \le 7,\ where\ x, y\ are\ integer$$

$$f_4:\ \min\ z = \frac{1}{x^2+y^2+1} - 1.1e^{-x^2-y^2}$$

$$subject\ to\quad -3 \le x, y \le 3,\ where\ x, y\ are\ integer$$

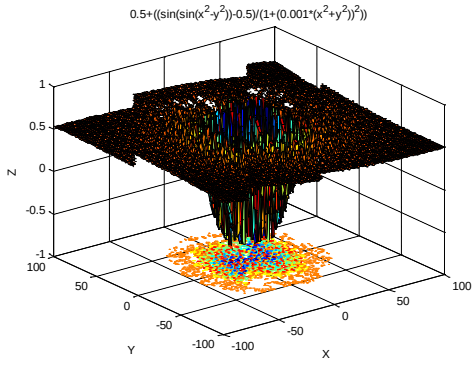
$$f_5:\ \min\ z = \frac{x^2+e^y}{e^x+y^2}$$

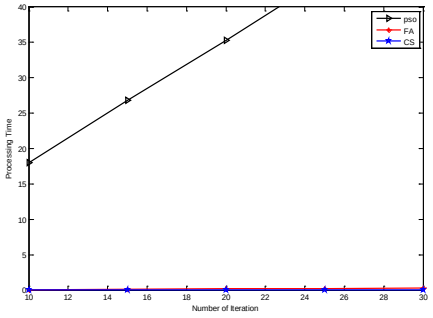
$$subject\ to\quad 1 \le x \le 4;\quad 0 \le y \le 2, where\ x, y\ are\ integer$$

The figures (1-5.a,b) shows the graphical representation of the solution space of the objective function, side by side, with a 2d plot comparing PSO, FA and CS with respect to Computational time. The value of the decision variables obtained were almost identical for the three different SI algorithms. However a slight improvement in the objective function and its decision variables values can be noticed in CS solution results. Then CS have the advantage of reaching a better objective function value as well as the fast convergence.

Table (1): Comparison results of the PSO, FA and CS.

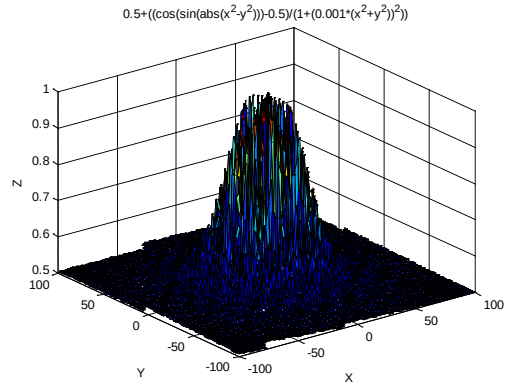
Fun. / Tec.	Num. of Iteration	PSO		FA		CS	
		Optimal value	Time (Sec.)	Optimal value	Time (Sec.)	Optimal value	Time (Sec.)
f_1	10	(-6,5) z=-0.8365	18.00	(-2,-50) z=-0.596	0.081	(-3,-100) z=-0.70691	0.024
	15	(-66,5) z=-0.8365	26.82	(-66,32) z=-0.3745	0.164	(-1,-52) z=-0.5873	0.035
	20	(4,7) z=-0.8358	35.37	(0,-66) z=-0.6436	0.177	(-4,-100) z=-0.6821	0.043
	25	(0,-11) z=- 0.8215	44.01	(0,-8) z=-0.50467	0.231	(4,-100) z=-0.69531	0.044
	30	(-1,3) z=-0.8355	52.55	(-3,-18) z=-.5117	0.265	(3,-100) z=-0.7094	0.065
f_2	10	(82,64) z=0.50035	13.721	(-100,-46) z=0.50035	0.0944	(94,100) z=0.50011	0.0312
	15	(-84,42) z=0.50052	19.632	(-89,72) z=0.500247	0.142	(-95,-100) z=0.50013	0.0337
	20	(-29,86) z=0.5006	26.657	(-100,-7) z=0.50039	0.196	(100,98) z=0.50010	0.0399
	25	(-87,-58) z= 0.5003	33.486	(-98,91) z=0.50016	0.244	(-98,100) z=0.50011	0.0518
	30	(22,96) z=0.50043	40.007	(-87,98) z=0.50014	0.266	(-98,100) z=0.50011	0.0669
f_3	10	(2,0) z=0.5454	0.753	(2,0) z=0.5454	0.0734	(2,0) z=0.5454	0.0275
	15	(2,0) z=0.5454	1.433	(2,0) z=0.5454	0.1074	(3,0) z=0.5454	0.0308
	20	(2,0) z=0.5454	1.463	(2,0) z=0.5454	0.1321	(3,0) z=0.5454	0.0387
	25	(2,0) z=0.5454	1.737	(1,0) z=0.5454	0.1752	(1,0) z=0.5454	0.0531
	30	(1,0) z=0.5454	2.323	(3,0) z=0.5454	0.2322	(3,0) z=0.5454	0.0566
f_4	10	(0,0) z=-0.1	1.0775	(0,0) z=-0.1	0.07951	(0,0) z=-0.1	0.04279
	15	(0,0) z=-0.1	1.5725	(0,0) z=-0.1	0.11831	(0,0) z=-0.1	0.04904
	20	(0,0) z=-0.1	2.2214	(0,0) z=-0.1	0.15030	(0,0) z=-0.1	0.05308
	25	(0,0) z=-0.1	2.5044	(0,0) z=-0.1	0.20335	(0,0) z=-0.1	0.06190
	30	(0,0) z=-0.1	3.3674	(0,0) z=-0.1	0.22500	(0,0) z=-0.1	0.08736
f_5	10	(4,0) z=0.31136586	0.7349	(4,0) z=0.31136586	0.07689	(4,0) z=0.31136586	0.02968
	15	(4,0) z=0.31136586	1.0383	(4,0) z=0.31136586	0.13825	(4,0) z=0.31136586	0.03175
	20	(4,0) z=0.31136586	1.4458	(4,0) z=0.31136586	0.21487	(4,0) z=0.31136586	0.06566
	25	(4,0) z=0.31136586	1.7264	(4,0) z=0.31136586	0.21881	(4,0) z=0.31136586	0.06974
	30	(4,0) z=0.31136586	1.9723	(4,0) z=0.31136586	0.24019	(4,0) z=0.31136586	0.08310

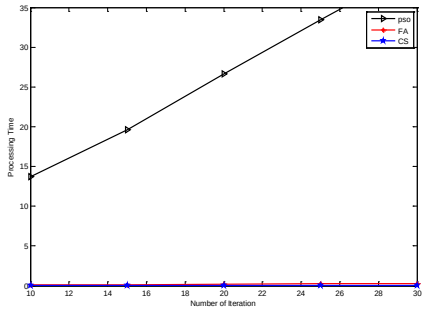




(a) (b)

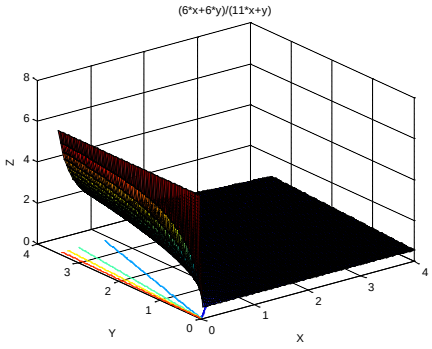
Fig.1. (a) Objective function f_1 iterative space using SI algorithms. (b) 2d plot for the convergence time of PSO, FA, and CS.

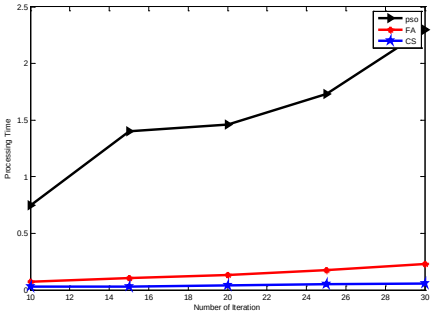




(a) (b)

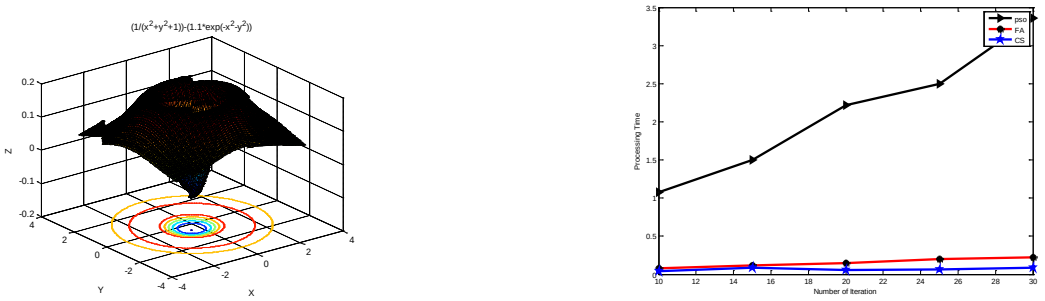
Fig.2. (a) Objective function f_2 iterative space using SI algorithms. (b) 2d plot for the convergence time of PSO, FA, and CS.





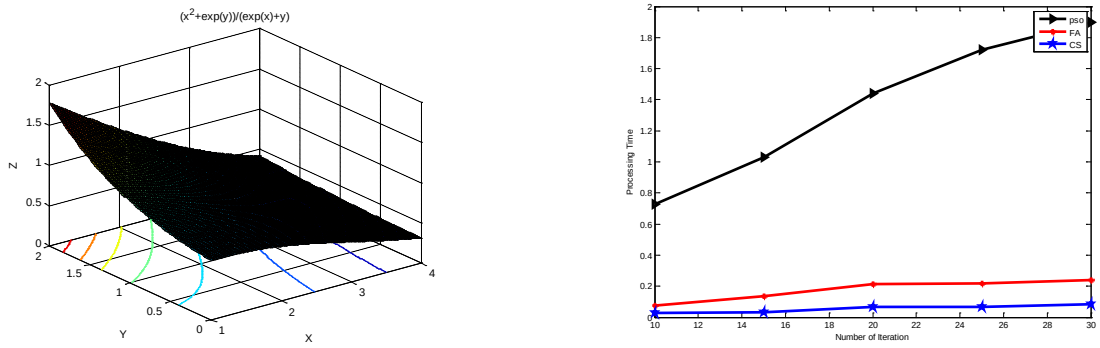
(a) (b)

Fig.3. (a) Objective function f_3 iterative space using SI algorithms. (b) 2d plot for the convergence time of PSO, FA, and CS.



(a) (b)

Fig.4. (a) Objective function f_4 iterative space using SI algorithms. (b) 2d plot for the convergence time of PSO, FA, and CS.



(a) (b)

Fig.5. (a) Objective function f_5 iterative space using SI algorithms. (b) 2d plot for the convergence time of PSO, FA, and CS.

Fig. 1-2, (b). Evidently shows that CS and FA algorithms are quite better than the PSO in the terms of computational time for reaching optimum or near optimum optimization. PSO computation time reached a value of 40-50 second to obtain optimality while FA and CS algorithms took less than a second to reach the same optimal goal. This could be indicated to the multi-packs in the functions f_1,f_2 only. While in simple function such as f_3,f_4 and f_5 shown in figures (3, 4, 5-b) respectively, CS algorithm is still in advance with less dominance. Then comes the FA algorithm, while PSO algorithm comes last in terms of computation time.

Overall the three algorithms reached almost exactly the same objective function and decision variables value in all the simple functions f_3,f_4 and f_5

7. Conclusions

The current research work managed to solve integer fractional programming problem (IFPP) using three different swarm intelligence (SI) algorithms. The numerical result and statistical analysis indicated that the proposed methods perform significantly better than the previously used the classical methods. The three used algorithms named particle swarm optimization (PSO), firefly algorithm (FA), and cuckoo search (CS) were tested on benchmark examples and managed to solve it all. The comparative study of the results gave a clear indication for the superiority of CS in reducing the computational time. These observations could easily be noticed in multi-packs functions. The CS algorithm is firstly ranked again in terms of the value of the obtained near optimal solution. According to the IFPP solution results, the three algorithms could be ordered as follows CS, FA, and finally PSO with respect to convergence time and obtained optimization value.

References

[1] E. C. Laskari, K. E. Parsopoulos, and M. N. Vrahatis, "Particle swarm optimization for integer programming," in *Evolutionary Computation, 2002. CEC'02. Proceedings of the 2002 Congress on*, 2002, vol. 2, pp. 1582–1587.

[2] S. Kitayama and K. Yasuda, "A method for mixed integer programming problems by particle swarm optimization," *Electrical Engineering in Japan*, vol. 157, pp. 40–49, 2006.

[3] Y. Li, L. Li, Q. Wen, and Y. Yang, "Integer programming via chaotic ant swarm," in *Natural Computation, 2007. ICNC 2007. Third International Conference on*, 2007, vol. 4, pp. 489–493.

[4] A. W. Mohemmed, N. C. Sahoo, and T. K. Geok, "Solving shortest path problem using particle swarm optimization," *Applied Soft Computing*, vol. 8, pp. 1643–1653, 2008.

[5] 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.

[6] H. Chen, S. Wang, and H. Wang, "Particle Swarm Optimization Based on Genetic Operators for Nonlinear Integer Programming," in *Intelligent Human-Machine Systems and Cybernetics, 2009. IHMSC'09. International Conference on*, 2009, vol. 1, pp. 431–433.

[7] P. Wu, L. Gao, Y. Ge, and D. Zou, "An effective global harmony search algorithm for integer programming problems," in *Computer Design and Applications (ICCDA), 2010 International Conference on*, 2010, vol. 1, pp. 1–40.

[8] H. Yang and S. Zhao, "A Hybrid Approach Based on Immune Particle Swarm Optimization and Integer Liner Programming for the Container Loading Problem," in *Biomedical Engineering and Computer Science (ICBECS), 2010 International Conference on*, 2010, pp. 1–4.

[9] Y. Liu and L. Ma, "Bee colony foraging algorithm for integer programming," in *Business Management and Electronic Information (BMEI), 2011 International Conference on*, 2011, vol. 5, pp. 199–201.

[10] K. D. S.B. Singh Pal A., "Use of Particle Swarm Optimization Algorithm for Solving Integer and Mixed Integer Optimization Problems," *International Journal of Computing Science and Communication Technologies*, vol. VOL. 4, NO. 1,, pp. 663–667.

[11] D. Datta and J. R. Figueira, "A real-integer-discrete-coded particle swarm optimization for design problems," *Applied Soft Computing*, vol. 11, pp. 3625–3633, 2011.

[12] N. BACANIN, I. BRAJEVIC, and T. Milan, "Firefly Algorithm Applied to Integer Programming Problems."

[13] I. M. Hezam and O. AbdelRaouf, "Particle Swarm Optimization Approach For Solving Complex Variable Fractional Programming Problems," *International Journal of Engineering Research & Technology (IJERT)*, vol. Vol. 2 Issue 4,, pp. 2672–2677, 2013.

[14] T. B. Farag, "A Parametric Analysis on Multicriteria Integer Fractional Decision-Making Problems," Faculty of Science, Helwan University.

[15] D. Bratton and J. Kennedy, "Defining a standard for particle swarm optimization," in *Swarm Intelligence Symposium, 2007. SIS 2007. IEEE*, 2007, pp. 120–127.

[16] K. O. Jones and G. Boizanté, "Comparison of Firefly algorithm optimisation, particle swarm optimisation and differential evolution," in *Proceedings of the 12th International Conference on Computer Systems and Technologies*, 2011, pp. 191–197.

[17] K. Y. Lee and M. A. El-Sharkawi, *Modern heuristic optimization techniques: theory and applications to power systems*, vol. 39. Wiley-IEEE press, 2008.

[18] X.-S. Yang, *Nature-inspired metaheuristic algorithms*. Luniver Press, 2011.

[19] X.-S. Yang, "Firefly algorithms for multimodal optimization," in *Stochastic algorithms: foundations and applications*, Springer, 2009, pp. 169–178.

[20] X.-S. Yang, S. S. Sadat Hosseini, and A. H. Gandomi, "Firefly algorithm for solving non-convex economic dispatch problems with valve loading effect," *Applied Soft Computing*, vol. 12, pp. 1180–1186, 2012.

[21] A. H. Gandomi, X.-S. Yang, and A. H. Alavi, "Cuckoo search algorithm: a metaheuristic approach to solve structural optimization problems," *Engineering with Computers*, vol. 29, pp. 17–35, 2013.

[22] R. Rajabioun, "Cuckoo optimization algorithm," *Applied Soft Computing*, vol. 11, pp. 5508–5518, 2011.

[23] E. Valian, S. Mohanna, and S. Tavakoli, "Improved cuckoo search algorithm for feed forward neural network training," *International Journal of Artificial Intelligence & Applications*, vol. 2, pp. 36–43, 2011.

[24] S. Walton, O. Hassan, K. Morgan, and M. Brown, "Modified cuckoo search: A new gradient free optimisation algorithm," *Chaos, Solitons & Fractals*, vol. 44, pp. 710–718, 2011.

[25] X.-S. Yang and S. Deb, "Engineering optimisation by cuckoo search," *International Journal of Mathematical Modelling and Numerical Optimisation*, vol. 1, pp. 330–343, 2010.

[26] X.-S. Yang and S. Deb, "Cuckoo search via Lévy flights," in *Nature & Biologically Inspired Computing, 2009. NaBIC 2009. World Congress on*, 2009, pp. 210–214.