J Ind Eng Int (2014) 10:56
DOI 10.1007/s40092-014-0056-8

ORIGINAL RESEARCH

Solving Fractional Programming Problems based on Swarm Intelligence

Osama Abdel Raouf · Ibrahim M. Hezam

Received: 7 July 2013 / Accepted: 18 March 2014 / Published online: 11 April 2014

Abstract This paper presents a new approach to solve Fractional Programming Problems (FPPs) based on two different Swarm Intelligence (SI) algorithms. The two algorithms are: Particle Swarm Optimization, and Firefly Algorithm. The two algorithms are tested using several FPP benchmark examples and two selected industrial applications. The test aims to prove the capability of the SI algorithms to solve any type of FPPs. The solution results employing the SI algorithms are compared with a number of exact and metaheuristic solution methods used for handling FPPs. Swarm Intelligence can be denoted as an effective technique for solving linear or nonlinear, non-differentiable fractional objective functions. Problems with an optimal solution at a finite point and an unbounded constraint set, can be solved using the proposed approach. Numerical examples are given to show the feasibility, effectiveness, and robustness of the proposed algorithm. The results obtained using the two SI algorithms revealed the superiority of the proposed technique among others in computational time. A better accuracy was remarkably observed in the solution results of the industrial application problems.

Keywords Swarm intelligence · Particle swarm optimization · Firefly algorithm · Fractional programming

O. A. Raouf · I. M. Hezam
Operations Research and DSS Department, Menofia University, Shebien El-koum, Menofia 32511, Egypt
e-mail: osamaabd@ci.menofia.edu.eg

I. M. Hezam (✉)
Department of Mathematics and computer, Faculty of Education, Ibb University, Ibb city, Yemen
e-mail: ibrahizam.math@gmail.com

Introduction

This paper, considers the following general Fractional Programming Problem (FPP) mathematical model (Jaberipour and Khorram 2010):

$$\min / \max z(x_1, x_2, \dots, x_n) = \sum_{i=1}^{p} \frac{f_i(x)}{g_i(x)} \tag{1}$$

$$\begin{aligned} &h_k(x) \geq 0, \quad k = 1, \dots, K;\\ &m_j(x) = 0, \quad j = 1, \dots, J;\\ &x_i^l \leq x_i \leq x_i^u, \quad i = 1, \dots, n;\\ &g_i(x) \neq 0, \quad i = 1, 2, \dots, p. \end{aligned} \tag{2}$$

where f, g, h and m are linear, quadratic, or more general functions. Fractional programming of the form Eq. (1) arises reality 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. Fractional programming is a nonlinear programming method that has known increasing exposure recently and its importance, in solving concrete problems, is steadily increasing. Furthermore, nonlinear optimization models describe practical problems much better than the linear optimization, with many assumptions, does. The FPPs are particularly useful in the solution of economic problems in which different activities use certain resources in different proportions. While the objective is to optimize a certain indicator, usually the most favorable return on allocation ratio subject to the constraint imposed on the availability of resources; it also has a number of important practical applications in manufacturing, administration, transportation, data mining, etc.

The methods to solve FFPs can be broadly classified into exact (traditional) and metaheuristics approaches.

The traditional method as: Wolf (1985) who introduced the parametric approach, Charnes and Cooper (1973) solved the linear FFPs by converting FPP into an equivalent linear programming problem and solved it using already existing standard algorithms for LPP, Farag (2012); Hasan and Acharjee (2011); Hosseinalifam (2009); Stancu-Minasian (1997), reviewed some of the methods that treated solving the FPP as the primal and dual simplex algorithm. The crisscross, which is based on pivoting, within an infinite number of iterations, either solves the problem or indicates that the problem is infeasible or unbounded. The interior point method, as well as Dinkelbach algorithms both reduces the solution of the LFP problem to the solution of a sequence of LP problems. Isbell Marlow method, Martos Algorithm, CambiniMarteins Algorithm, Bitran and Novaes Method, Swarups Method, Harvey M. Wagner and John S. C. Yuan, Hasan, B.M., and Acharjee, S., developed a new method for solving FLPP based on the idea of solving a sequence of auxiliary problems so that the solutions of the auxiliary problems converge to the solution of the FPP.

Moreover, there are many recent approaches employing traditional mathematical methods for solving the ratio optimization FPP as: Dür et al. (2007) who introduced an algorithm called dynamic multistart improving Hit-and-Run (DMIHR) and applied it to the class of fractional optimization problems. DMIHR combines IHR, a well-established stochastic search algorithm, with restarts. The development of this algorithm is based on a theoretical analysis of Multistart Pure Adaptive Search, which relies on the Lipschitz constant of the optimization problem. Shen et al. (2009) proposed algorithm for solving sum of quadratic ratios fractional programs via monotonic function. The proposed algorithm is based on reformulating the problem as a monotonic optimization problem. It turns out that the optimal solution, which is provided by the algorithm, is adequately guaranteed to be feasible and to be close to the actual optimal solution. Jiao et al. (2013) presented global optimization algorithm for sum of generalized polynomial ratios problem which arises in various practical problems. The global optimization algorithm was proposed for solving sum of generalized polynomial ratios problem, which arise in various engineering design problems. By utilizing exponential transformation and new three-level linear relaxation method, a sequence of linear relaxation programming of the initial nonconvex programming problems are derived, which are embedded in a branch and bound algorithm. The algorithm was shown to attain finite convergence to the global minimum through successive refinement of a linear relaxation of the feasible region and/or of the objective function and the subsequent solutions of a series of linear programming sub-problems.

A few studies in recent years used metaheuristics approaches to solve FFPs. Sameeullah et al. (2008) presented a genetic algorithm-based method to solve the linear FFPs. A set of solution point is generated using random numbers, feasibility of each solution point is verified, and the fitness value for all the feasible solution points are obtained. Among the feasible solution points, the best solution point is found out which then replaces the worst solution point. A pair-wise solution points is used for crossover and a new set of solution points is obtained. These steps are repeated for a certain number of generations and the best solution for the given problem is obtained. Calvete et al. (2009) developed a genetic algorithm for the class of bi-level problems in which both level objective functions are linear fractional and the common constraint region is a bounded polyhedron. Jaberipour and Khorram (2010) proposed algorithm for the sum-of-ratios problem-based harmony search algorithm. Bisoi et al. (2011) developed neural networks for nonlinear FPP. The research proposed a new projection neural network model. It is theoretically guaranteed to solve variational inequality problems. The multi-objective mini–max nonlinear fractional programming was defined and its optimality is derived by using its Lagrangian duality. The equilibrium points of the proposed neural network model are found to correspond to the Karush Kuhn Trcker point associated with the nonlinear FPP. Xiao (2010) presented a neural network method for solving a class of linear fractional optimization problems with linear equality constraints. The proposed neural network model have the following two properties. First, it is demonstrated that the set of optima to the problems coincides with the set of equilibria of the neural network models which means the proposed model is complete. Second, it is also shown that the model globally converges to an exact optimal solution for any starting point from the feasible region. Pal et al. (2013) used Particle Swarm Optimization algorithm for solving FFPs. Hezam and Raouf (2013a b, c), introduced solution for integer FPP and complex variable FPP-based Swarm Intelligence under uncertainty.

The purpose of this paper is to investigate the solution for the FPP using Swarm Intelligence. The remainder of this paper is organized as follows. "Methodology" will introduce Swarm Intelligence methodology. Illustrative examples and discussion on the results are presented in "Illustrative examples with discussion and results". "Industry applications" introduces industry applications. Finally, conclusions are presented 'in "Conclusions".

Methodology

Swarm Intelligence (SI) is research inspired by observing the naturally intelligent behavior of biological agent swarms within their environments. SI algorithms have provided effective solutions to many real-world type optimization problems, that are NP-Hard in nature. This study investigates the effectiveness of employing two relatively new SI metaheuristic algorithms in providing solutions to the FPPs. The algorithms investigated are Particle Swarm Optimization (PSO), and Firefly Algorithm (FA). Brief descriptions of these algorithms are given in the subsections below.

1. Particle Swarm Optimization (PSO)

PSO (Yang 2011) 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;
- $\widehat{y}_i^k$ The neighborhood best position of the particle.

The velocity update step is specified for each dimension $j \in \{1, , N_d\}$ 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

$$v_i^k(t+1) = wv_i^k(t) + c_1 r_1(t)(y_i(t) - x_i(t)) + c_2 r_2(t)(\widehat{y}_i(t) - x_i(t)). \quad (3)$$

where w is weighting function, $c_{1,2}$ are weighting coefficients, $r_{1,2}(t)$ are random numbers between 0 and 1. The current position (searching point in the solution space) can be modified by the following equation:

$$x_i^k(t+1) = x_i^k + v_i^{k+1} \quad (4)$$

Penalty functions

In the penalty functions method, the constrained optimization problem is solved using unconstrained optimization method by incorporating the constraints into the objective function thus transforming it into an unconstrained problem.

$$\text{Fitness} = f(x) + \text{Penalty functions} * \text{Error}.$$

The detailed steps of the PSO algorithm is given as 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 Eq. (3). 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 Eq. (4). After updating, x_i^k should be checked and limited within the allowed range.

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

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

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

Step 6: *Termination checking* Repeat Steps 2–4 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.

2. Firefly algorithm (FA)

FA Yang (2011), is 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 detailed steps of the PSO algorithm is given as below:

Step 1 Define objective function $f(x), x = (x_1, x_2, \ldots, x_d)$, 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, L_i, 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 $x_j, j \in 1, 2, \ldots d$.

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:

$$x_i^t + 1 = x_i^t + \beta_\circ \exp - \gamma r_t^2 (x_j^t - x_i^t) + \alpha \epsilon_i^t \quad (5)$$

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 ϵ_i 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 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 2–6 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.

Illustrative examples with discussion and results

Ten diverse examples were collected from literature to demonstrate the efficiency and robustness of solving FFPs. The obtained numerical results are compared to their relevance found in references; some examples were also solved using exact method f_1 and f_3. Table 1 shows they attained the comparison result. The algorithms have been implemented by MATLAB R2011. The simulation parameter settings results of FA are: population size, 50; α (randomness), 0.25; minimum value of β, 0.20; γ (absorption), 1.0; iterations, 500; and PSO are population size of 50, the inertia weight was set to change from 0.9 (*wmax*) to 0.4 (warming) over the iterations. Set c_1:0.12 and c_2 :1.2, , iterations:500.

The functions related to the difference examples list in the previous table are followers:

f_1: $\max z = \frac{4x+2y+10}{x+2y+5}$
subject to $x + 3y \leq 30$;
$-x + 2y \leq 5$; $x, y \geq 0$.

f_2 : $\min z = \left(\frac{x+y+1}{x+y+2}\right)^{1.5} \times \left(\frac{x+y+3}{x+y+4}\right)^{2.1} \times \left(\frac{x+y+5}{x+y+6}\right)^{1.2} \times \left(\frac{x+y+7}{x+y+8}\right)^{1.1}$.
subject to $x - y = 0$; $1 \leq x \leq 2$; $x, y \geq 0$.

f_3: $\min z = \frac{x+y+1}{2x-y+3}$
subject to $0 \leq x \leq 1$; $0 \leq x \leq 1$.

f_4: $\max z = \frac{8x+7y-2.33(9x^2+4y^2)^{0.5}}{20x+12y-2.33(3x^2+2xy+4y^2)^{0.5}}$
subject to $2x + y \leq 18$; $x + 2y \leq 16$; $x, y \geq 0$.

f_5: $\max z = \frac{2x+y}{x} + \frac{2}{y}$
subject to
$2x + y \leq 6$; $3x + y \leq 8$; $-x + y \geq -1$ $x, y \geq 1$.

f_6: $\max z = \frac{-x^2+3x-y^2+3y+3.5}{x+1} + \frac{y}{x^2-2x+y^2-8y+20}$
subject to
$2x + y \leq 6$; $3x + y \leq 8$; $-x + y \geq -1$; $1 \leq x \leq 2.25$; $1 \leq y \leq 4$.

f_7 : $\max z = \frac{-x^2y^{0.5}+2xy^{-1}-y^2+2.8x^{-1}y+7.5}{xy^{1.5}+1} + \frac{y+0.1}{-x^2y^{-1}-3x^{-1}+2xy^2+9y^{-1}+12}$
subject to $2x^{-1} + xy \leq 4$; $x + 3x^{-1}y \leq 5$; $x^2 - 3y^3 \leq 2$; $1 \leq x \leq 3$; $1 \leq y \leq 3$.

f_8: $\max z = \frac{37x+73y+13}{13x+13y+13} + \frac{63x-18y+39}{13x+26y+13}$
subject to $5x + 3y = 3$; $1.5 \leq x \leq 3$; $x, y \geq 0$.

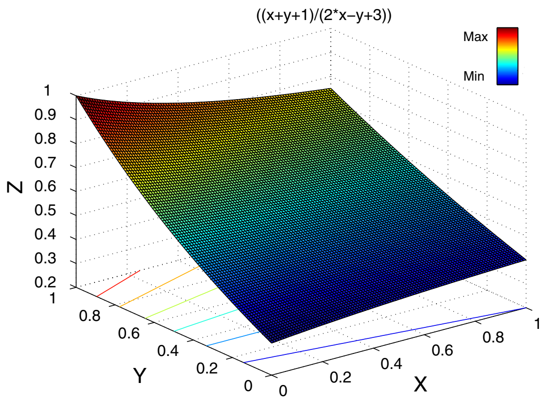
f_9: $\min z = \frac{2x+y}{x+10} + \frac{2}{y+10}$
subject to $-x^2 - y^2 + 3 \leq 0$; $-x^2 - y^2 + 8y - 3 \leq 0$; $2x + y \leq 6$; $3x + y \leq 8$; $x - y \leq 1$; $1 \leq x \leq 3$; $1 \leq y \leq 4$.

f_{10}: $\max z = \left(\frac{13x+13y+13}{37x+73y+13}\right)^{-1.4} \times \left(\frac{64x-18y+39}{13x+26y+13}\right)^{1.2} - \left(\frac{x+2y+5v+50}{x+5y+5v+50}\right)^{0.5} \times \left(\frac{x+2y+4v+50}{5y+4v+50}\right)^{1.1}$
subject to $2x + y + 5v \leq 10$; $5x - 3y = 0$; $1.5 \leq x \leq 3$; $x, y, v \geq 0$.

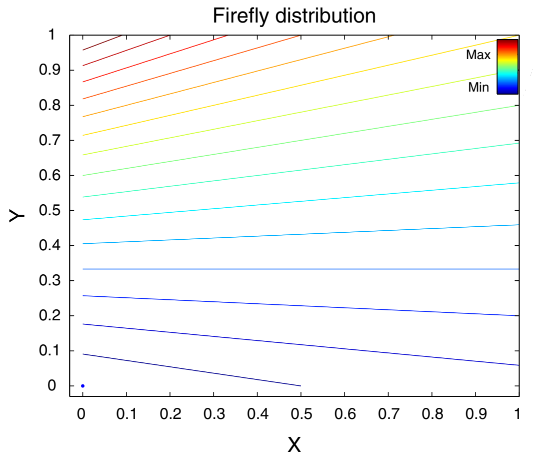
The numerical results obtained using PSO and FA techniques are compared to assorted exact methods and metaheuristic techniques as shown in Table 1. Four exact methods were selected for solving the 10 benchmark functions and carrying out the comparison. The four methods are C.C. Transformation, Dinkelbach algorithm, Goal Setting and Approximation and global optimization. Neural network and harmony search are the other two metaheuristic intelligent techniques incorporated in the compare test. The some calculations are obtained out of the numerical solutions of all the ten functions. PSO and FA proved their capability in obtaining the optimal solution for all the test functions. The results were obtained from the PSO, FA almost identical to the obtained using exact methods. PSO and FA proved also to give better results compared to other intelligent techniques, such as neural network and harmony search f_3,f_{10}. Finally, PSO and FA managed to give solutions to problems that could not be solved with exact method due to difficult mathematical calculation for complex nonlinear. Figures 1 and 2 are sample plots of two maximum and minimum function optimization results. Figure 1a shows the objective function optimized value of (0.333) for function f_3 where the blue colored dots on the objective space represent the swarm particle searching for the optimized minimum value. The same particles swarm with the same color could be observed in the decision variable space of Fig. 1b, c at values of (0,0), (0,0) trying to reach the optimized decision variables values using FA, PSO algorithms, respectively. Figure 2a shows the objective function optimized value of (4.0608) for function f_6 where the red colored dots on the objective space represent the swarm

Table 1 Comparison results of the SI with other methods

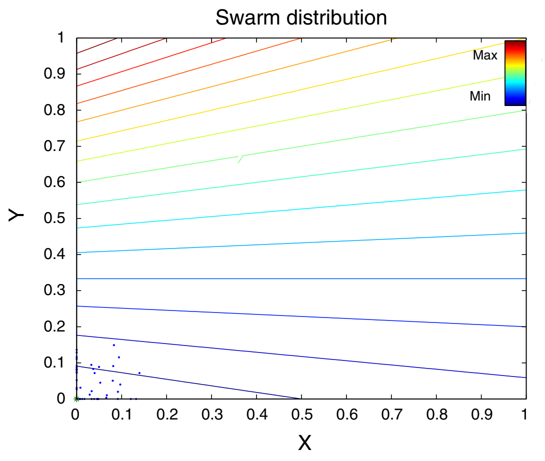
Fun.	Technique/reference	Decision variable optimal value	Objective function optimize value
f_1 (max)	C.C. transformation exact method	$(x^*, y^*) = (30, 0)$	$z^* = 3.714286$
	Dinkelbach algorithm exact method	$(x^*, y^*) = (30, 0)$	$z^* = 3.714286$
	PSO	$(x^*, y^*) = (30, 0)$	$z^* = 3.7142857$
	FA	$(x^*, y^*) = (29.949, 0)$	$z^* = 3.7138877$
f_2 (min)	C.C. Transformation exact method	$- - --$	$- - --$
	Dinkelbach algorithm exact method	$- - --$	$- - --$
	Jiao et al. (2006) Global optimization	$(x^*, y^*) = (1.00000068, 1.0000000458)$	$z^* = 0.3360$
	PSO	$(x^*, y^*) = (1, 1)$	$z^* = 0.3360$
	FA	$(x^*, y^*) = (1, 1)$	$z^* = 0.33603$
f_3 (min)	C.C. Transformation exact method	$(x^*, y^*) = (0, 0)$	$z^* = 0.333$
	Dinkelbach algorithm exact method	$(x^*, y^*) = (0, 0)$	$z^* = 0.333$
	Zhang and Lu (2012) Neural network	$(x^*, y^*) = (0.5, 3)$	$z^* = 4.5$
	PSO	$(x^*, y^*) = (0, 0)$	$z^* = 0.333$
	FA	$(x^*, y^*) = (0, 0)$	$z^* = 0.333$
f_4 (max)	C.C. transformation exact method	$- - --$	$- - --$
	Dinkelbach algorithm exact method	$- - --$	$- - --$
	Mehrjerdi (2011) Goal setting and approximation	$(x^*, y^*) = (7.229, 0)$	$z^* = 0.084$
	PSO	$(x^*, y^*) = (1.0264, 5.7391)$	$z^* = 0.3383$
	FA	$(x^*, y^*) = (1.175, 6.59)$	$z^* = 0.3382$
f_5 (max)	C.C. transformation exact method	$- - --$	$- - --$
	Dinkelbach algorithm exact method	$- - --$	$- - --$
	Shen et al. (2011) Global optimization	$(x^*, y^*) = (1, 4)$	$z^* = 6.5$
	PSO	$(x^*, y^*) = (1, 4)$	$z^* = 6.5$
	FA	$(x^*, y^*) = (1, 4)$	$z^* = 6.5$
f_6 (max)	C.C. transformation exact method	$- - --$	$- - --$
	Dinkelbach algorithm exact method	$- - --$	$- - --$
	Shen et al. (2011) Global optimization	$(x^*, y^*) = (1, 1.75)$	$z^* = 4.0608$
	PSO	$(x^*, y^*) = (1, 1.7377)$	$z^* = 4.0608$
	FA	$(x^*, y^*) = (1, 1.7438)$	$z^* = 4.06082$
f_7 (max)	C.C. transformation exact method	$- - --$	$- - --$
	Dinkelbach algorithm exact method	$- - --$	$- - --$
	Shen et al. (2011) Global optimization	$(x^*, y^*) = (1, 1)$	$z^* = 5.5167$
	PSO	$(x^*, y^*) = (1, 1)$	$z^* = 5.5167$
	FA	$(x^*, y^*) = (1, 1)$	$z^* = 5.5167$
f_8 (max)	C.C. transformation exact method	$- - --$	$- - --$
	Dinkelbach algorithm exact method	$- - --$	$- - --$
	Wang and Shen (2008) Global optimization	$(x^*, y^*) = (3, 4)$	$z^* = 5$
	PSO	$(x^*, y^*) = (3, 4)$	$z^* = 5$
	FA	$(x^*, y^*) = (3, 4)$	$z^* = 5$
f_9 (min)	C.C. transformation exact method	$- - --$	$- - --$
	Dinkelbach algorithm exact method	$- - --$	$- - --$
	Jiao et al. (2013) Global optimization	$(x^*, y^*) = (1, 1.4142)$	$z^* = 0.48558$
	PSO	$(x^*, y^*) = (1, 1.4)$	$z^* = 0.48$
	FA	$(x^*, y^*) = (1, 1.41423)$	$z^* = 0.485604$
f_{10} (max)	C.C. transformation exact method	$- - --$	$- - --$
	Dinkelbach algorithm exact method	$- - --$	$- - --$
	Jaberipour and Khorram (2010) Harmony search	$(x^*, y^*, v^*) = (1.5, 1.5, 1.1)$	$z^* = 8.1207$
	PSO	$(x^*, y^*, v^*) = (1.5, 1.5, 0)$	$z^* = 8.279866$
	FA	$(x^*, y^*, v^*) = (1.5, 1.49, 0)$	$z^* = 8.251791$



(a) Objective space of PSO



(b) Decision variables space of FA



(c) Decision variables space of PSO

Fig. 1 Swarm distributions searching for optimization value of f_3

particle searching for the optimized maximum value. The same particles swarm with the same color could be observed in the decision variable space of Fig. 2b, c at values of (1,1.7438), (1,1.7377) trying to reach the optimized decision variables values using FA, PSO algorithms, respectively.

Industry applications

A. Design of a gear train

A gear train problem is selected from Deb and Srinivasan (2006), and Shen et al. (2011); shown in Fig. 3 below. A compound gear train is to be designed to achieve a specific gear ratio between the driver and driven shafts. It is a pure integer fractional optimization problem used to validate the integer handling mechanism. The gear ratio for gear train is defined as the ratio of the angular velocity of the output shaft to that of the input shaft. It is desirable to produce a gear ratio as close as possible to $1/6.931$. For each gear, the number of teeth must be between 12 and 60. The design variables T_a, T_b, T_d, and T_f are the numbers of teeth of the gears a, b, d, and f, respectively, which must be integers.

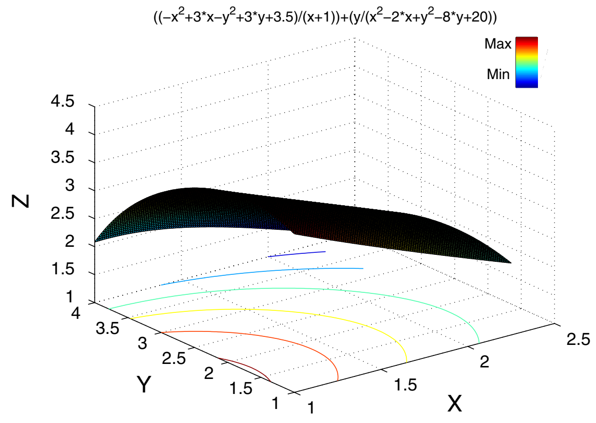
$\bar{x} = (T_d, T_b, T_a, T_f)^T$.

The optimization problem is expressed as:

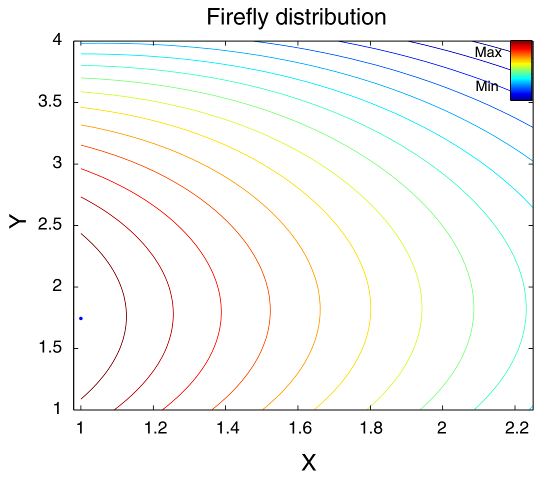
$$\min z = \left(\frac{1}{6.931} - \frac{\lfloor T_d \rfloor \lfloor T_b \rfloor}{\lfloor T_a \rfloor \lfloor T_f \rfloor}\right)^2 = \left(\frac{1}{6.931} - \frac{\lfloor x_1 \rfloor \lfloor x_2 \rfloor}{\lfloor x_3 \rfloor \lfloor x_4 \rfloor}\right)^2.$$

subject to $12 \le x_i \le 60, \quad i = 1, 2, 3, 4.$

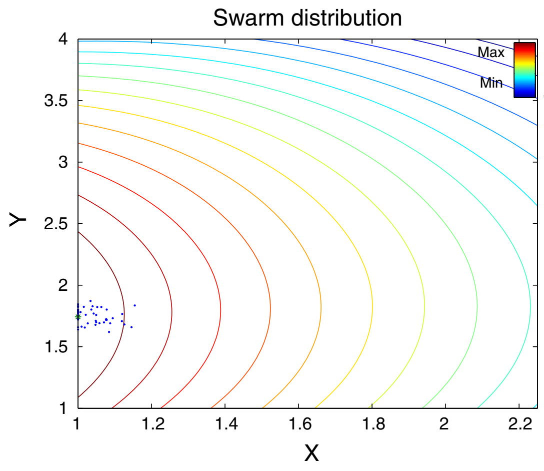
The constraint ensures that the error between obtained gear ratio and the desired gear ratio is not more than the 50 % of the desired gear ratio.



(a) Objective space of PSO



(b) Decision variables space of FA



(c) Decision variables space of PSO

Fig. 2 Swarm distributions searching for optimization value of f_6

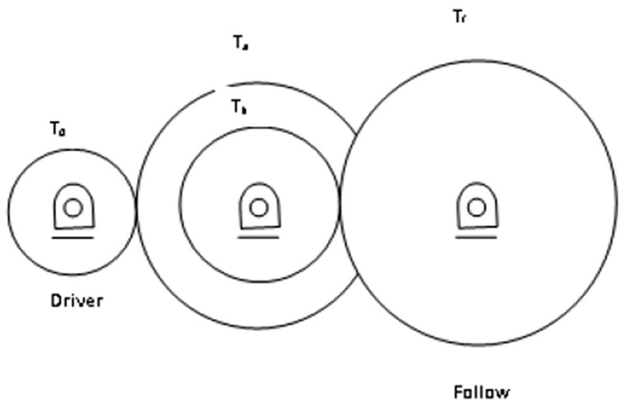


Fig. 3 A gear train

The detailed accuracy performance concerning the solution of FA and PSO is listed in Table 2. The comparison is held in terms of the best, error, mean, and standard deviation values. These values were obtained out of 20 independent runs. The table also shows the best optimization value, the convergence time and the amount addressed memory resources. First refereeing to the obtained optimization value indicates a better achievement for FA. On the other hand, the convergence time and the

Table 2 Result comparisons between FA and PSO on gear train problem

	FA	PSO
Best	2.7E−012	2.700857E−012
Error (%)	0 %	0.0003174 %
Mean	5.1314E−012	1.1371E−011
SD	7.6868E−012	1.01307E−011
Time (s)	65	45
Memory utilization	483–484	492–494

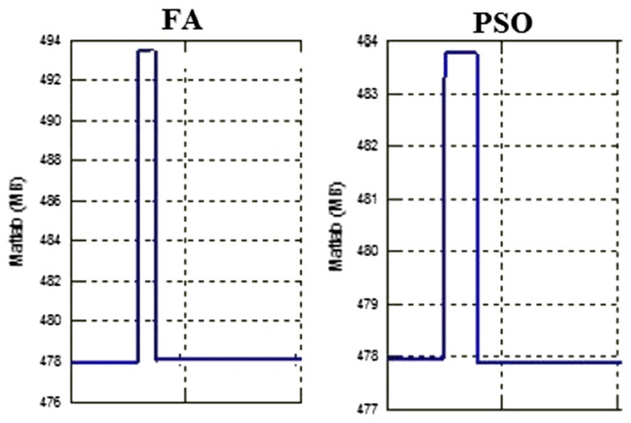


Fig. 4 Memory usage indicator

amount of addressed memory resources indicate a better achievement for PSO. The PSO algorithm utilizes memory amount of 483–484 as shown in Fig. 4, while FA algorithm utilizes memory amount of 492–494 as shown in the same figure.

B. Proportional integral derivative (PID) controller

Proportional integral derivative (PID) controllers are widely used to build automation equipment in industries; shown in Fig. 5 below. They are easy to design, implement, and are applied well in most industrial control systems process control, motor drives, magnetic, etc.

Correct implementation of the PID depends on the specification of three parameters: proportional gain (K_p), integral time (T_i) and derivative time (T_d). These three parameters are often tuned manually by trial and error, which has a major problem in the time needed to accomplish the task. and the fractional-order PID controller parameters vector is (K_p, T_i, T_d, λ, δ). The PID controller is a special case of the fractional-order PID controller, we simply set $\lambda = \delta = 1$.

Assume that the system is modeled by an nth-order process with time delay L:

$$G_p(s) = \frac{b_m s^m + b_{m-1}s^{m-1} + \ldots + b_1 s + b_0}{s^n + a_{n-1}s^{n-1} + \ldots + a_1 s + a_0} \times e^{-Ls} \quad (6)$$

Here, we assume $n > m$ and the system (6) is stable. The fractional-order PID controller has the following transfer function:

$G_C(s) = K_p + T_i s^{-\lambda} + T_d s^{\delta}$

Table 3 Results for the integer order PID

Technique	K_p	T_i	T_d	Step response
FA	500	249.68	500	1.0020
PSO	200	491.68	100	1.0208
Maiti et al. (2008)	214.84	361.57	76.76	1.0339

Table 4 Results for the fractional order PID

Tec.	K_p	T_i	T_d	λ	δ	Step res.
FA	786.99	484.8	308	2	1	1.0079
PSO	857.54	292.98	206.9	1.62	1.75	1.0179
Maiti et al. (2008)	442.68	324.03	115.27	1.5	1.41	1.0288

The optimization problem is summarized as follows:
min $z = z(K_p, T_i, T_d, \lambda, \delta)$.
subject to $L \le K_p, T_i, T_d, \lambda, \delta \le U$.
where the z, L, U is given by the designer. Note that the constraint is introduced to guarantee the stability of the closed-loop system. Also, the values of five design parameters (K_p, T_i, T_d, λ, δ) are directly determined by solving the above optimization problem.

Simulation example

Consider the following the transfer function presented in Maiti et al. (2008):

$G_p(s) = \frac{1}{0.8s^{2.2} + 0.5s^{0.9} + 1}$.

The initial parameters are chosen randomly in the following range: $K_p, [1, 1{,}000]; T_i, [1, 500]; T_d, [1, 500]; \lambda, [0, 2]; \delta, [0, 2]$. We want to design a controller such that the closed loop system has a maximum peak overshoot $M_p = 10\,\%$ and trise = 0.3 s. This translates to $\xi = 0.65$ (damping ratio), $\omega_0 = 2.2\,s^{-1}$ (undamped natural frequency). We then find out the positions of the dominant poles of the closed loop system,

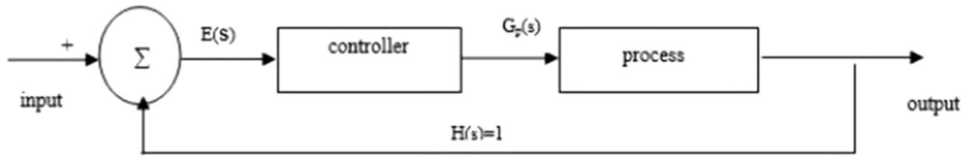
$P_{1,2} = -\xi\omega_0 \pm j\omega_0\sqrt{1-\xi^2}$.

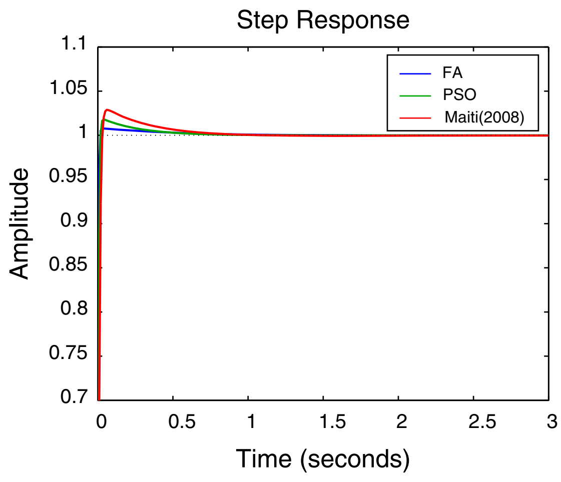
The dominant poles for the closed loop controlled system should lie at $(-1.43 + j1.67)$ and $(-1.43 - j1.67)$. For $p_1 = (-1.43 + j1.67)$, the characteristic equation is:

$1 + \frac{K_p + T_i(-1.43+j1.67)^{\lambda} + T_d(-1.43+j1.67)^{\delta}}{0.8(-1.43+j1.67)^{2.2} + 0.5(-1.43+j1.67)^{0.9} + 1} = 0.$

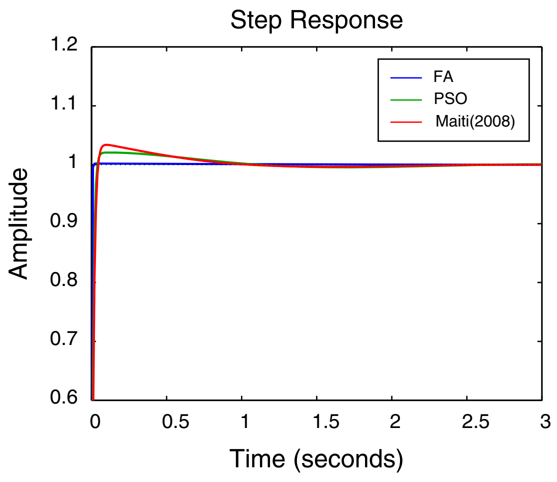
Tables 3 and 4 illustrate the calculated optimized parameters of PID controller using PSO, FA and using the

Fig. 5 Generic closed loop system





(a) Comparison Close Loop step response of the system with FA, PSO, and Mittia (2008) fractional PID controller



(b) Comparison Close Loop step response of the system with FA, PSO, and Mittia (2008) integer PID controller

Fig. 6 Closed loop unit step response

algorithm of Maiti et al. (2008). In Table 3 which gives only the result of the integer-order PID controller parameters, when the variables (λ) and (δ) are set to a value of (1), it could be observed that the optimized parameters calculated using FA algorithm generates the best control step response as illustrate in Fig. 6. It could be concluded also from the same figure that PSO algorithm with tuned parameters introduces a better step response than Maiti et al. (2008). Table 4 introduces optimized parameters of the fractional-order PID controller where the same indicated remarks could be observed as that of the integer order.

Conclusions

The paper presents a new approach to solve FFPs based on two Swarm Intelligence (SI) algorithms. The two types are PSO, and FA. Ten-benchmark problem were solved using the two SI algorithm and many other previous approaches. The results employing the two SI algorithms were compared with the other exact and metaheuristic approaches previously used for handling FPPs. The two algorithms proved their effectiveness, reliability and competences in solving different FPP. The two SI algorithms managed to successfully solve large-scale FPP with an optimal solution at a finite point and an unbounded constraint set. The computational results proved that SI turned out to be superior to other approaches for all the accomplished tests yielding a higher and much faster growing mean fitness at less computational time. A better memory utilization was obtained using the PSO algorithm compared to FA algorithm. Two industrial application problems were solved proving the superiority of FA algorithm over PSO algorithm reaching a better optimized solution. A better optimized ratio was obtained that generated a zero traction error in the gear train application and a better control response was obtained in the PID controller application. In the two applications, the best results were acquired using the two SI algorithms with an advantage to the FA algorithm optimization results and an advantage to the PSO algorithm in the computational time.

Acknowledgments The authors are thankful to the learned referee and editor for their valuable comments and suggestions for the improvement of the paper.

References

Bisoi S, Devi G, Rath A (2011) Neural networks for nonlinear fractional programming. Intern J Sci Eng Res 2(12):1–5
Calvete HI, Galé C, Mateo PM (2009) A genetic algorithm for solving linear fractional bilevel problems. Ann Oper Res 166(1):39–56
Charnes A, Cooper W (1973) An explicit general solution in linear fractional programming. Nav Res Logistics Q 20(3):449–467
Deb K, Srinivasan A (2006) Innovization: innovating design principles through optimization. In: Proceedings of the 8th Annual

Conference on Genetic and Evolutionary Computation, ACM, New York, pp 1629–1636
Dür M, Khompatraporn C, Zabinsky ZB (2007) Solving fractional problems with dynamic multistart improving hit-and-run. Ann Oper Res 156(1):25–44
Farag TB (2012) A parametric analysis on multicriteria integer fractional decision-making problems. PhD thesis, Faculty of Science, Helwan University, Helwan, Egypt
Hasan MB, Acharjee S (2011) Solving lfp by converting it into a single lp. Intern J Oper Res 8(3):1–14
Hezam IM, Raouf OA (2013a) Particle swarm optimization approach for solving complex variable fractional programming problems. Intern J Eng 2(4)
Hezam IM, Raouf OA (2013b) Employing three swarm intelligent algorithms for solving integer fractional programming problems. Intern J Sci Eng Res (IJSER) 4:191–198
Hezam IM, Raouf MMH (2013c) Osama Abdel: solving fractional programming problems using metaheuristic algorithms under uncertainty. Intern J Adv Comput 46:1261–1270
Hosseinalifam M (2009) A fractional programming approach for choice-based network revenue management. PhD thesis, Universite de Montreal, Montreal, Canada
Jaberipour M, Khorram E (2010) Solving the sum-of-ratios problems by a harmony search algorithm. J Comput Appl Math 234(3):733–742
Jiao H, Guo Y, Shen P (2006) Global optimization of generalized linear fractional programming with nonlinear constraints. Appl Math Comput 183(2):717–728
Jiao H, Wang Z, Chen Y (2013) Global optimization algorithm for sum of generalized polynomial ratios problem. Appl Math Model 37(1):187–197
Maiti D, Biswas S, Konar A (2008) Design of a fractional order pid controller using particle swarm optimization technique. arXiv, preprint arXiv:0810.3776
Mehrjerdi YZ (2011) Solving fractional programming problem through fuzzy goal setting and approximation. Appl Soft Comput 11(2):1735–1742
Pal A, Singh S, Deep K (2013) Solution of fractional programming problems using pso algorithm. In: 2013 IEEE 3rd International Advance Computing Conference (IACC), pp 1060–1064
Sameeullah A, Devi SD, Palaniappan B (2008) Genetic algorithm based method to solve linear fractional programming problem. Asian J Info Technol 7(2):83–86
Shen P, Chen Y, Ma Y (2009) Solving sum of quadratic ratios fractional programs via monotonic function. Appl Math Comput 212(1):234–244
Shen P, Ma Y, Chen Y (2011) Global optimization for the generalized polynomial sum of ratios problem. J Global Optim 50(3):439–455
Stancu-Minasian I (1997) Fractional programming: theory, methods and applications, vol 409. Kluwer academic publishers, Dordrecht
Wang C-F, Shen P-P (2008) A global optimization algorithm for linear fractional programming. Appl Math Comput 204(1):281–287
Wolf H (1985) A parametric method for solving the linear fractional programming problem. Oper Res 33(4):835–841
Xiao L (2010) Neural network method for solving linear fractional programming. In: 2010 International Conference On Computational Intelligence and Security (CIS), pp 37–41
Yang X-S (2011) Nature-inspired metaheuristic algorithms. Luniver Press, United Kingdom
Zhang Q-J, Lu XQ (2012) A recurrent neural network for nonlinear fractional programming. Math Prob Eng 2012