This Chapter presents strategy of particle swarm optimization (PSO) algorithm introduced by Kennedy and Eberhart (1995) and Clerc M. (2006) for solving fractional programming problems. In this Chapter, possibility of using particle swarm optimization algorithm for solving fractional programming problems has been considered. The particle swarm optimization technique has been tried on a set of 12 test problems taken from the literature whose optimal solutions are known. A penalty function approach (Deb,K,2000) is incorporated for handling constraints of the problem. It has been shown that PSO algorithm can be effectively used to solve fractional programming problems also.

4.1 INTRODUCTION

A special class of optimization programming problems, where the objective function is the ratio of two given functions. These ratios represent some kind of efficiency measure for a system and arise whenever optimization of ratios such as performance/cost, income/investment, return/risk, output/input etc are required.

The general Mathematical model of an FPP is given as

\[
\text{Optimize (maximize or minimize) } \frac{f(x)}{g(x)}
\]

subject to the constraints:

\[
h_i(x) \leq or \geq or = b_i, \quad i = 1,2, \ldots, m.
\]

where \( x = (x_1, x_2, \ldots, x_n) \) is called a decision vector and \( x_1, x_2, \ldots, x_n \) are called unknown variables or decision variables. The above Mathematical programming is called a linear fractional programming problem if all the functions i.e. \( f, h_i \), \( g \) are linear. However if one or more of these functions are non linear, then the problem is known as a non linear fractional programming problem.

There are numbers of methods for finding the solution to such problems. Carnes and Cooper (1962) made fundamental contribution to the class of linear fractional programming
problems by establishing that solving the linear fractional programming problem is equivalent to solving at most two linear programming problems.

Kanti Swarup (1967) has given a straightforward generalization of Simplex method for solving linear fractional programming problems under the assumptions that the solution set is regular and the denominator of the objective functions is strictly positive throughout the feasible region.

Jaganathan (1966) and Dinkelbach (1967) have suggested techniques for solving the non-linear fractional programming problems and have established many important results in this area. Fractional programs have been used with increasing frequency as realistic models in a variety of decision making situations and consequently have received considerable attention in this field.

Jiao H et al (2006) presented a global optimization algorithm for solving generalized linear fractional programming problems. In the algorithm an equivalent non-convex programming problem was introduced firstly and then they constructed the linear lower bounding functions about the objective and constraint functions of the problem. The algorithm is applied to a number of test problems and convergence to the global minima was achieved.

S. Tantawy (2008) proposed an interactive method for solving linear fractional programming problems and shown that this method can be used for sensitivity analysis when a scalar parameter is introduced in the objective coefficients.

Joshi Vishwash Deep et al (2011) investigated a method to calculate the lower and upper bounds of the total fractional transport cost when the supply and demand quantity are varying. A set of two-level transportation problems is transformed into the one level Mathematical programs to find out objective value.

E. Sohrab et al (2012) introduced an interval valued linear fractional programming with interval coefficients in the objective function and proved that an IVLFP can be converted to an optimization problem with interval valued objective function in which its bounds are linear fraction functions.

In this Chapter we have considered the possibility of solving different types of fractional programming problems using Particle Swarm Optimization Technique.
4.2 PSO ALGORITHM AND ITS CONCEPT

Particle Swarm Optimization (Kennedy, J. and Eberhart, R, 1995) is one of Swarm intelligence algorithm. It is proposed by American social psychologist Kennedy and electrical engineer Russell Eberhart in 1995. The thought come from the idea of information sharing system (Eberhart, R. C. and Shi, Y, 2001) in Biological communities. Because of its easy concept and profound background, PSO has huge advantage in scientific research and engineering application. So once the PSO was proposed it has been the focus of most scholars. But compared with the GA both the mathematical basis and convergence analysis PSO has shortage in foundation. PSO is lack of strict Mathematical foundation. The search of PSO theory has two direction macro and micro. In the direction of micro the focal point is the search about trajectory of particles in the solution space. In the direction of macro focus the whole group and give the mathematical model discuss the convergence of the algorithm.

Initially PSO was designed for continuous optimization problems, but later a wide variety of challenging engineering and science applications came into being.

In PSO, particle is the basic component of solution in the solution space. The dimension of the solution is \( D \). The \( i^{th} \) particle is given as \( X_i = (x_{i1}, x_{i2}, \ldots, x_{iD}) \), where \( x_{ik} \) is the position of the \( i^{th} \) particle in the solution space of \( k \) i.e. in the \( k \)-th direction.

Particle swarm is made up with \( N \) particles and particle swarm stands for the \( N \) candidate solution . A population is formed of a group of \( n \) particles. Population \( X(t) = [X_1(t), X_2(t), \ldots, X_n(t)] \), where \( X_i \) is the position of the \( i^{th} \) particle in the population.

The velocity of the particle is the variation of one iteration and it stands for displacement of the solution in the space \( D \). \( V_i = (v_{i1}, v_{i2}, \ldots, v_{iD}) \) where \( v_{ik} \) is the velocity of the \( i^{th} \) particle in the solution space of \( k \) i.e. in the \( k \)-th direction.

The standard Mathematical description of PSO (Jiao H et al, 2006) is as follow.

The search space is \( D \) and the number of particles is \( n \). The position of the \( i^{th} \) particle is \( X_i = (x_{i1}, x_{i2}, \ldots, x_{iD}) \), the velocity of the \( i^{th} \) particle is \( V_i = (v_{i1}, v_{i2}, \ldots, v_{iD}) \), the best position of the \( i^{th} \) particle is \( p_i = (p_{i1}, p_{i2}, \ldots, p_{iD}) \) named as pbest, the best position of the population is \( p_g = (p_{g1}, p_{g2}, \ldots, p_{gD}) \) named as gbest. The expression of \( i^{th} \) particle in the direction of \( D \) (Shi, Y. and Eberhart, R. C, 1998) is given as
\begin{align*}
    v_{id}(t + 1) &= wv_{id}(t) + c_1 r_1 (p_{id}(t) - x_{id}(t)) + c_2 r_2 (p_{gd}(t) - x_{id}(t)) \quad \ldots \ldots (4.1) \\
    x_{id}(t + 1) &= x_{id}(t) + v_{id}(t + 1) \quad \ldots \ldots (4.2)
\end{align*}

where \( 1 \leq i \leq n, 1 \leq d \leq D \)

c_1, c_2 are normal numbers named as acceleration factors, \( r_1, r_2 \) are the random numbers in \([0,1]\) and \( w \) is the inertial factor. The change range of position and velocity is \([-x_{d_{\max}}, x_{d_{\max}}]\) and \([-v_{d_{\max}}, v_{d_{\max}}]\).

If \( v_{id} > v_{d_{\max}} \), then

\( v_{id} = v_{d_{\max}} \). Particles change their positions by using (4.2) and (4.3). When the particles find their best position the iteration stop. For minimum of \( f(x) \), the best position of \( i \)-th particle is given by

\begin{align*}
    p_i(t + 1) &= \begin{cases} 
    p_i(t), & f(x_i(t + 1)) \geq f(p_i(t)) \\
    x_i(t + 1), & f(x_i(t + 1)) < f(p_i(t)) 
\end{cases} \quad \ldots \ldots (4.3)
\end{align*}

The initial experiments (Shi, Y. and Eberhart, R. C, 1998) suggested that a value between 0.8 and 1.2 provided good results. Later work (Eberhart and Shi, 2000) indicates that the optimal strategy is to initially set \( w \) to 0.9 and reduce it linearly to 0.4, allowing initial exploration followed by acceleration toward an improved also available to adjust the inertia weight. For example, in (Eberhart and Shi, 2000) the adaptation of \( w \) using a fuzzy system was reported to significantly improve PSO performance. Another effective strategy is to use an inertia weight with a random component, rather than time decreasing. For example, (Eberhart and Shi, 2001) successfully used \( w = u(0.5, 1) \), a uniformly distributed random number between 0.5 and 1. There are also studies, e.g., (Zheng et al., 2003), in which an increasing inertia weight was used.

The PSO algorithm is shown below

For \( t = 1 \) to the max. bound on the max_no_ of_iterations,

For \( i = 1 \) to the swarm_size.

For \( j = 1 \) to the problem_dimensionality.

Apply the velocity update equation (4.2)

Update position using equation (4.3)

End-for-j;

53
Compute fitness of updated position;
   If needed, update historical information for pbest
   & gbest;
   End-for-i;
   Terminate if gbest meets problem requirements;
   End-for-t;
   End algorithm.

4.3. DISCUSSION ON RESULTS

   We use the acceleration factors $c_1$ and $c_2$ equals to 2. Although other settings were
   also be used in the literature but $c_1$ and $c_2$ are usually in the range from [0,4]. The swarm
   size i.e. number of particles we used here (5 to 10)* dimension. There is no hard and fast rule
   for it. For most of the problems 10-50 particles are large enough to get good result. The range
   of the particle we set as Vmax to keep the swarm under control for example as $x$ belongs to
   [-10,10] then Vmax =20. One another approach in the literature is given as Vmax=[upper
   limit-lower limit]/5. An inertial factor $w=0.721348$ is used (Eberhart, R. C. and Shi, Y.,2000)
   in order to overcome the problem of premature convergence of PSO. For solving all the
   problems in this paper a C++ code has been developed and compiled in Microsoft visual C++
   compiler and the following are recorded :
   (i) ANE= Average no. of functions evaluations of successful runs.
   (ii) Success Rate(SR)= (No of successful runs/Total runs)*100
   (iii) AET i.e. average execution time of successful runs.
   (iv) K=1 to 3 (neighborhood size of particles).
Table 4.1: (Comparison of results between PSO and known optimal Solutions from literature of FPP)

<table>
<thead>
<tr>
<th>Pb No</th>
<th>Optimal Solution by PSO $(x_1, x_2, \ldots, x_n; f)$</th>
<th>Known Optimal Solution $(x_1, x_2, \ldots, x_n; f)$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>PSO parameters used (SS, NR, ANE, AET, SR%)</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>(0.3,975; 0.21)</td>
<td>(0.1,5.12; 0.21)</td>
</tr>
<tr>
<td></td>
<td>(10,20,434,247,100)</td>
<td>(Animesh Biswas et al, 2011)</td>
</tr>
<tr>
<td>2</td>
<td>(11.18,0.752)</td>
<td>(11.16, 0.11; 0.75)</td>
</tr>
<tr>
<td></td>
<td>(10,20,13958,171,90)</td>
<td>(Animesh Biswas et al, 2011)</td>
</tr>
<tr>
<td>3</td>
<td>(10,20,60,60,24,0,118,145,60,44,60,1.05)</td>
<td>(0,0,60,45,30,60,75,45,30,601,1.06)</td>
</tr>
<tr>
<td></td>
<td>(50,20,335,42937,60)</td>
<td>(Joshi et al, 2011)</td>
</tr>
<tr>
<td>4</td>
<td>(30; 3.714)</td>
<td>(30, 0; 3.7143)</td>
</tr>
<tr>
<td></td>
<td>(20,20,861,93,100)</td>
<td>(S. Tantawy, 2008)</td>
</tr>
<tr>
<td>5</td>
<td>(4.5; 1.5)</td>
<td>(2.8; 9.999993499)</td>
</tr>
<tr>
<td></td>
<td>(20,20,471,62,100)</td>
<td>(Jiao H et al, 2006)</td>
</tr>
<tr>
<td>6</td>
<td>(1,1.02362; 1.347222)</td>
<td>(1.0000000458, 1.0; 1.347219844)</td>
</tr>
<tr>
<td></td>
<td>(20,20,222,31,100)</td>
<td>(Jiao H et al, 2006)</td>
</tr>
<tr>
<td>7</td>
<td>(1.1; 0.336037)</td>
<td>(1.000000688, 1.000000458; 0.336037)</td>
</tr>
<tr>
<td></td>
<td>(10,20,10000,15359,80)</td>
<td>(Jiao H et al, 2006)</td>
</tr>
<tr>
<td>8</td>
<td>(1.1; 1.346382)</td>
<td>(1.000000, 1.000000; 1.346382)</td>
</tr>
<tr>
<td></td>
<td>(10,20,10000,2406,100)</td>
<td>(Jiao H et al, 2006)</td>
</tr>
<tr>
<td>9</td>
<td>(1.1; -0.72882929647)</td>
<td>(1.000000, 1.000000; -2882929647)</td>
</tr>
<tr>
<td></td>
<td>(10,20,10000,203,100)</td>
<td>(Jiao H et al, 2006)</td>
</tr>
<tr>
<td>10</td>
<td>(0, 0.75; 0.146154)</td>
<td>(0, 0.75; 0.146154) (E. Sohrab et al, 2012)</td>
</tr>
<tr>
<td></td>
<td>(10,20,238,31,100)</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>(1.6667, 11.3332; 4.0454)</td>
<td>(1.6667, 11.3332; 4.0454)</td>
</tr>
<tr>
<td></td>
<td>(20,20,5982,78,100)</td>
<td>(E. Sohrab et al, 2012)</td>
</tr>
<tr>
<td>12</td>
<td>(30, 7.544657)</td>
<td>(5.1053, 2.4394; 7.544657)</td>
</tr>
<tr>
<td></td>
<td>(10,20,2276,31,100)</td>
<td>(S. Tantawy, 2008)</td>
</tr>
</tbody>
</table>
4.4 CONCLUSIONS

In this Chapter, we present our experience of using particle swarm optimization algorithm for solving different types of fractional programming problems taken from the literature. The performance of the PSO algorithm has shown that this algorithm can also be used for solving different types of fractional programming problems with or without constraints very effectively. Results are checked on a set of 12 test problems taken from literature and given in the Appendix B of the Thesis. Our results show that the PSO algorithm in solving fractional programming problems performs best in most of the cases as shown in table 4.1. In future we intend to apply this strategy of PSO to solve the larger real life optimization problems.