17

Particle Swarm Optimization Algorithm for Transportation Problems

Han Huang[1] and Zhifeng Hao[2]
[1]*School of Software Engineering, South China University of Technology*
[2]*College of Mathematical Science, South China University of Technology*
P. R. China

1. Brief Introduction of PSO

Particle swarm optimization (PSO) is a newer evolutionary computational method than genetic algorithm and evolutionary programming. PSO has some common properties of evolutionary computation like randomly searching, iteration time and so on. However, there are no crossover and mutation operators in the classical PSO. PSO simulates the social behavior of birds: Individual birds exchange information about their position, velocity and fitness, and the behavior of the flock is then influenced to increase the probability of migration to regions of high fitness. The framework of PSO can be described as Figure 1.

1. Initialize K Particles $X_1, X_2, ..., X_K$, calculating $pbest_1, ..., pbest_K$ and $gbest$, $t = 1$;
2. For $i = 1, ..., K$ and $j = 1, ..., N$, update particles and use X_i to refresh $pbest_i$ and $gbest$; (shown as equation 1 and 2)
3. $t = t + 1$; If $t > \max\_gen$, output $gbest$ and exit; else, return Step 2.

Figure 1. The framework of classical PSO

In the optimal size and shape design problem, the position of each bird is designed as variables x, while the velocity of each bird v influences the incremental change in the position of each bird. For particle d Kennedy proposed that position x^d be updated as:

$$x^d{}_{t+1} = x^d{}_t + v^d{}_{t+1} \tag{1}$$

$$v^d{}_{t+1} = v^d{}_t + c_1 r_1 (p^d{}_t - x^d{}_t) + c_2 r_2 (p^g{}_t - x^d{}_t) \tag{2}$$

Here, $p^d{}_t$ is the best previous position of particle d at time t, while $p^g{}_t$ is the global best position in the swarm at time t. r_1 and r_2 are uniform random numbers between 0 and 1, and $c_1 = c_2 = 2$.

2. Particle Swarm Optimization for linear Transportation Problem

2.1 Linear and Balance Transportation Problem

The transportation problem (TP) is one of the fundamental problems of network flow optimization. A surprisingly large number of real-life applications can be formulated as a TP. It seeks determination of a minimum cost transportation plan for a single commodity from a number of sources to a number of destinations. So the LTP can be described as: Given there are n sources and m destinations. The amount of supply at source i is a_i and the demand at destination j is b_j. The unit transportation cost between source i and destination j is c_{ij}. x_{ij} is the transport amount from source i to destination j, and the LTP model is:

$$\begin{aligned} \min \quad & z = \sum_{i=1}^{n}\sum_{j=1}^{m} c_{ij}x_{ij} \\ \text{s.t.} \quad & \sum_{j=1}^{m} x_{ij} \le a_i \qquad i = 1,2,...,n \\ & \sum_{i=1}^{n} x_{ij} \ge b_j \qquad j = 1,2,\ldots,m \cdot \\ & x_{ij} \ge 0; \qquad i = 1,2,...,n; j = 1,2,...,m \end{aligned} \tag{3}$$

TP has been paid much attention to and classified into several types of transmutation. According to the nature of object function, there are four types: (1) linear TP and nonlinear TP. (2) single objective TP and multi objective. Based on the type of constraints, there are planar TP and solid TP. The single object LTP dealt with in this paper is the basic model for other kinds of transportation problems.

A special LTP called balanced LTP is considered as follows:

$$\begin{aligned} \min \quad & z = \sum_{i=1}^{n}\sum_{j=1}^{m} c_{ij}x_{ij} \\ \text{s.t.} \quad & \sum_{j=1}^{m} x_{ij} = a_i \qquad i = 1,2,...,n \\ & \sum_{i=1}^{n} x_{ij} = b_j \qquad j = 1,2,...,m \cdot \\ & \sum_{i=1}^{n} a_i = \sum_{j=1}^{m} b_j \\ & x_{ij} \ge 0; \qquad i = 1,2,...,n; j = 1,2,...,m \end{aligned} \tag{4}$$

The fact that a $n \times m$ LTP can be changed into a $n \times (m+1)$ balanced LTP, can be found in operational research because the demand at destination $m+1$ could be calculated by $b_{m+1} = \sum_{i=1}^{n} a_i - \sum_{j=1}^{m} b_j$ with the condition $\sum_{i=1}^{n} a_i \geq \sum_{j=1}^{m} b_j$.

2.2 Initialization of PSO for Linear and Balance Particle Swarm Optimization

A particle $X = \begin{bmatrix} x_{11} & \cdots & x_{1m} \\ \cdots & \cdots & \cdots \\ x_{n1} & \cdots & x_{nm} \end{bmatrix}$ stands for a solution to LTP. There are nm particles initialized to form nm initial solutions in the initialization. Every element of set N can be chosen as the first assignment to generate the solutions dispersedly, which is good for obtaining the optimal solution in the iteration.
If a LTP is balanced, the following procedure can be used to obtain an initial solution:

```
program GetOnePrimal (var X: Particle, first: int)
  var   i,j,k: integer;
        N: Set of Integer;
  begin
    k :=0;
    N := {1,2,…,nm};
    repeat
        if k=0 then
            k:=first ;
        else
            k:= a random element in N;
        i := ⌊ (k - 1 )/m ⌋ + 1 ;
        j := ((k-1) mod m) +1;
        xij := min {ai, bj};
        ai := ai - xij;
        bi := bi - xij;
        N := N \ {k};
    until N is empty
  end.
```

And the initialization of PSO-TP can be designed as follows:

```
program Initialization
  var   i : integer;
  begin
   Get a balanced LTP;
        i := 1;
        repeat
            GetOnePrimal (Xi, i);
            i := i+1;
        until i > nm
  end.
```

2.3 Updating Rule of PSO for Linear and Balance Particle Swarm Optimization

In PSO algorithm, a new solution can be obtained by using the position updating rule as equations 2 and 3. However, the classical rule is unable to meet such constraints of TP as $\sum_{j=1}^{m} x_{ij} = a_i$ and $\sum_{i=1}^{n} x_{ij} = b_j$. A new rule is designed to overcome this shortcoming. For particle d, we propose that position X^d $(n \times m)$ be updated as

$$V^d{}_{t+1} = \begin{cases} \varphi_1(P^d{}_t - X^d{}_t) + \varphi_2(P^g{}_t - X^d{}_t) & t = 0 \\ \lambda_1 V^d{}_t + \lambda_2[\varphi_1(P^d{}_t - X^d{}_t) + \varphi_2(P^g{}_t - X^d{}_t)] & t > 0 \end{cases} \tag{5}$$

$$X^d{}_{t+1} = V^d{}_{t+1} + X^d{}_t \tag{6}$$

where $P^g{}_t \neq X^d{}_t$ and $P^d{}_t \neq X^d{}_t$.

If $P^g{}_t = X^d{}_t$ and $P^d{}_t \neq X^d{}_t$, $\varphi_1 = 1$. If $P^g{}_t \neq X^d{}_t$ and $P^d{}_t = X^d{}_t$, $\varphi_2 = 1$. If $P^g{}_t = X^d{}_t$ and $P^d{}_t = X^d{}_t$, $\lambda_1 = 1$.

$P^d{}_t$ $(n \times m)$ is the best previous position of particle d at time t, while $P^g{}_t$ $(n \times m)$ is the global best position in the swarm at time t. φ_1 and φ_2 are uniform random numbers in (0, 1), meeting $\varphi_1 + \varphi_2 = 1$, while λ_1 is a uniform random number between [0.8, 1.0) and $\lambda_2 = 1 - \lambda_1$.

$$if \quad t = 0,\ X^d{}_{t+1} = V^d{}_{t+1} + X^d{}_t$$

$$= \varphi_1(P^d{}_t - X^d{}_t) + \varphi_2(P^g{}_t - X^d{}_t) + X^d{}_t = (\varphi_1 P^d{}_t + \varphi_2 P^g{}_t) - (\varphi_1 X^d{}_t + \varphi_2 X^d{}_t) + X^d{}_t$$

$$\sum_{j=1}^{m} x^{d}{}_{ij}(t+1)$$

$$= \varphi_1 \sum_{j=1}^{m} p^{d}{}_{ij}(t) + \varphi_2 \sum_{j=1}^{m} p^{g}{}_{ij}(t) - (\varphi_1 \sum_{j=1}^{m} x^{d}{}_{ij}(t) + \varphi_2 \sum_{j=1}^{m} x^{d}{}_{ij}(t)) + \sum_{j=1}^{m} x^{d}{}_{ij}(t)$$

$$= \varphi_1 a_i + \varphi_2 a_i - (\varphi_1 a_i + \varphi_2 a_i) + a_i = a_i$$

$$\sum_{i=1}^{n} x^{d}{}_{ij}(t+1)$$

$$= \varphi_1 \sum_{i=1}^{n} p^{d}{}_{ij}(t) + \varphi_2 \sum_{i=1}^{n} p^{g}{}_{ij}(t) - (\varphi_1 \sum_{i=1}^{n} x^{d}{}_{ij}(t) + \varphi_2 \sum_{i=1}^{n} x^{d}{}_{ij}(t)) + \sum_{i=1}^{n} x^{d}{}_{ij}(t)$$

$$= \varphi_1 b_j + \varphi_2 b_j - (\varphi_1 b_j + \varphi_2 b_j) + b_j = b_j$$

$$if \quad t > 0,$$

$$X^{d}{}_{t+1} = V^{d}{}_{t+1} + X^{d}{}_{t}$$

$$= \lambda_1 V^{d}{}_{t} + \lambda_2 [\varphi_1 (P^{d}{}_{t} - X^{d}{}_{t}) + \varphi_2 (P^{g}{}_{t} - X^{d}{}_{t})] + X^{d}{}_{t}$$

$$= \lambda_1 V^{d}{}_{t} + \lambda_2 [(\varphi_1 P^{d}{}_{t} + \varphi_2 P^{g}{}_{t}) - (\varphi_1 X^{d}{}_{t} + \varphi_2 X^{d}{}_{t})] + X^{d}{}_{t}$$

$$= \lambda_1 (X^{d}{}_{t} - X^{d}{}_{t-1}) + \lambda_2 [(\varphi_1 P^{d}{}_{t} + \varphi_2 P^{g}{}_{t}) - (\varphi_1 X^{d}{}_{t} + \varphi_2 X^{d}{}_{t})] + X^{d}{}_{t}$$

$$\sum_{j=1}^{m} x^{d}{}_{ij}(t+1) = \lambda_1 (\sum_{j=1}^{m} x^{d}{}_{ij}(t) - \sum_{j=1}^{m} x^{d}{}_{ij}(t-1)) + \lambda_2 (\varphi_1 a_i + \varphi_2 a_i - (\varphi_1 a_i + \varphi_2 a_i)) + a_i$$

$$= \lambda_1 (a_i - a_i) + 0 + a_i = a_i$$

$$\sum_{i=1}^{n} x^{d}{}_{ij}(t+1)$$

$$= \lambda_1 (\sum_{i=1}^{n} x^{d}{}_{ij}(t) - \sum_{i=1}^{n} x^{d}{}_{ij}(t-1)) + \lambda_2 (\varphi_1 b_j + \varphi_2 b_j - (\varphi_1 b_j + \varphi_2 b_j)) + b_j$$

$$= \lambda_1 (b_j - b_j) + 0 + b_j = b_j$$

Therefore, $X^d{}_{t+1}$ would meet the condition that $\sum_{j=1}^{m} x^d{}_{ij}(t+1) = a_i$ and $\sum_{i=1}^{n} x^d{}_{ij}(t+1) = b_j$ with the function of Formulae 5 and 6. However, the new rule cannot ensure the last constraint that $x_{ij} \geq 0, i = 1,..,n, j = 1,...,m$. In the following section, an extra operator is given to improve the algorithm.

2.4 Negative Repair Operator

A particle of PSO-TP (Formula 7) will be influenced by the negative repair operator if $x_{ki} < 0, k = 1,...,n, i = 1,...,m$, which is indicated as follows:

$$X = \begin{bmatrix} x_{11} & \cdots & x_{1i} & \cdots & x_{1m} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ x_{k1} & \cdots & x_{ki} & \cdots & x_{km} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ x_{l1} & \cdots & x_{li} & \cdots & x_{lm} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ x_{n1} & \cdots & x_{ni} & \cdots & x_{nm} \end{bmatrix} \quad (7)$$

---

program RepairOnePos (var X: Particle, k,i: int)
begin

select the maximum element signed as x_{li} in Col. i;

$x_0 := x_{ki}$, $x_{li} := x_{li} - |x_0|$, $x_{ki} := 0$;

change elements in Row.k into $x_{kj} := \begin{cases} x_{kj} & x_{kj} = 0 \\ x_{kj} - \dfrac{|x_0|}{u} & x_{kj} > 0 \end{cases}$;

(u is the number of times when the following condition $x_{kj} > 0, j = 1,...,m$ is met)

change elements in Row. l into $x_{lj} := \begin{cases} x_{lj} & x_{kj} = 0 \\ x_{lj} + \dfrac{|x_0|}{u} & x_{kj} > 0 \end{cases}$;

end.

As a result, the procedure of negative repair operator can be described as:

```
program NegativeRepair (var X: Particle)
var   i,j: integer;
begin
   if some element of X is negative then
       repeat
         If x_ij<0 is found then
              RepairOnePos (X, i, j);
       until Every element of X is not negative
end.
```

2.5 PSO Mutation

Mutation is a popular operator in Genetic Algorithm, and a special PSO mutation is designed to help PSO-TP change the partial structure of some particles in order to get new types of solution. PSO-TP cannot fall into the local convergence easily because the mutation operator can explore the new solution.

```
program PSOMutation (var X: Particle)
begin
  Obtain p and q randomly meeting 0<p<n and 0<q<m;
   Select p rows {i1,…ip} and q lines {j1,…jq} randomly from matrix X to form a small matrix
Y (yij,i=1,…,p,j=1,…,q);
```

$$a^{y}{}_{i} = \sum_{j \in \{j_1,\ldots,j_q\}} x_{ij} \qquad (i = i_1,\ldots,i_p)$$

$$b^{y}{}_{j} = \sum_{i \in \{i_1,\ldots,i_p\}} x_{ij} \qquad (j = j_1,\ldots,j_q)$$

```
   Use a method like the one in initialization to form the initial assignment for Y;
  Update X with Y;
end.
```

2.6 The Structure of PSO-TP

According to the setting above, the structure of PSO-TP is shown as:

```
program PSO-TP (problem: balanced LTP of n×m size, pm: float)
var t:integer;
begin
  t:=0;
  Initialization;
```

Obtain $Pg_0(n \times m)$ and $Pd_0(n \times m)$(d=1,…,n×m);

repeat

t:=t+1;

Calculate X^d_t with Formula 5 and 6 (d=1,…,n×m);

NegativeRepair(X^d_t)(d=1,…,n×m);

Carry out PSOMutation(X^d_t) by the probability pm;

Update $Pg_t(n \times m)$ and $Pd_t(n \times m)$(d=1,…,n×m);

until meeting the condition to stop

end.

3. Numerical Results

There are two experiments in this section: one is comparing PSO-TP with genetic algorithm (GA) in some integer instances and the second is testing the performance of PSO-TP in the open problems. Both of the experiments are done at a PC with 3.06G Hz, 512M DDR memory and Windows XP operating system. GA and PSO-TP would stop when no better solution could be found in 500 iterations, which is considered as a virtual convergence of the algorithms. The probability of mutation in PSO-TP is set to be 0.05.

Problem\ five runs	PSO-TP Min	PSO-TP Ave	GA Min	GA Ave	PSO-TP Time(s)	GA Time(s)
P1 (3*4)	152	152	152	153	0.015	1.72
P2 (4*8)	287	288	290	301	0.368	5.831
P3 (3*4)	375	375	375	375	0.028	0.265
P4 (3*4)	119	119	119	119	0.018	1.273
P5 (3*4)	85	85	85	85	0.159	0.968
P6*(15*20)	596	598	-	-	36.4	-

Table 1. Comparison Between PSO-TP and GA

As Table 1 shows, both the minimum cost and average cost obtained by PSO-TP are less than those of GA. Furthermore, the time cost of PSO-TP is much less than that of GA. In order to verify the effectiveness of PSO-TP, 9 real number instances are computed and the results are shown in Table 2. Since GA is unable to deal with the real number LTP directly, only PSO-T is tested.

Problem\five runs	Optimal Value	PSO-TP Average	PSO-TP Time(s)
No.1	67.98	67.98	0.02
No.2	1020	1020	0.184
No.4	13610	13610	0.168
No.5	1580	1580	0.015
No.6	98	98	0.023
No.7	2000	2000	0.015
No.8	250	250	<0.001
No.9	215	215	0.003
No.10	110	110	0.012

Table 2. Performance of PSO-TP in open problems

According to the results in Table 2, PSO-TP can solve the test problems very quickly. The efficiency of PSO-TP may be due to the characteristic of PSO algorithm and the special operators. Through the function of the new position updating rule and negative repair operator, the idea of PSO is introduced to solve LTP successfully. The nature of PSO can accelerate the searching of the novel algorithm, which would also enable PSO-TP to get the local best solution. What's more, the PSO mutation as an extra operator can help PSO-TP to avoid finishing searching prematurely. Therefore, PSO-TP can be a novel effective algorithm for solving TP.

4. Particle Swarm Optimization for Non-linear Transportation Problem

4.1 Non-linear and Balance Transportation Problem

The unit transportation cost between source i and destination j is $f_{ij}(x_{ij})$ where x_{ij} is the transportation amount from source i to destination j, and TP model is:

$$\min \quad z = \sum_{i=1}^{n}\sum_{j=1}^{m} f_{ij}(x_{ij})$$

$$\text{s.t.} \sum_{j=1}^{m} x_{ij} \le a_i \qquad i = 1,2,...,n \tag{8}$$

$$\sum_{i=1}^{n} x_{ij} \ge b_j \qquad j = 1,2,\ldots,m \cdot$$

$$x_{ij} \ge 0; \qquad i = 1,2,...,n; j = 1,2,...,m$$

According to the nature of object function, there are four types: linear TP in which the function $f_{ij}(x_{ij})$ is linear and nonlinear TP in which $f_{ij}(x_{ij})$ is non-linear, as well as single objective and multi-objective TP. Based on the types of constraints, there are planar TP and solid TP. The single object NLTP is dealt with in this paper. In many fields like railway transportation, the relation between transportation amount and price is often non-linear, so NLTP is an important for application.

4.2 Framework of PSO for Non-linear TP

In the population of PSO-NLTP, an individual $X_i = \begin{bmatrix} x_{11} & \cdots & x_{1m} \\ \cdots & \cdots & \cdots \\ x_{n1} & \cdots & x_{nm} \end{bmatrix}$ stands for a solution to NLTP (Exp. 2), where $n \times m$ is the population size. There are $n \times m$ individuals initialized to form $n \times m$ initial solutions in the initialization. The initialization and mutation are the same as the ones in PSO-LTP (Section 2.2 and 2.5).
And the framework of PSO-NLTP is given:

Algorithm: PSO-NLTP
Input: NLTP problem (Exp. 8)
begin
  Initialization;
  Setting parameters;
  repeat
    Updating rule;
    Mutation;
    Updating the current optimal solution
  until meeting the condition to stop
end.
Output: Optimal solution for NLTP

In the parameter setting, The parameters of PSO-NLTP are all set adaptively: as the population size is $n \times m$, the size of mutation matrix Y is set randomly meeting 0<p<n and 0<q<m and the mutation probability P_m is calculated by $P_m = 0.005 \times N_t$, where $N_t = 1$ when $X_{best}^{(t)}$ is updated and $N_t = N_t + 1$ when $X_{best}^{(t)}$ remains the same as $X_{best}^{(t-1)}$.

4.3 Updating Rule of PSO-NLTP

As one of the important evolutionary operator, recombination is designed to optimize the individuals and make them meet the constraints of supply and demand as $\sum_{j=1}^{m} x_{ij} = a_i$ and

$\sum_{i=1}^{n} x_{ij} = b_j$ (Exp. 4). At the beginning of an iteration, every individual is recombined by the following expression.

$$X_i^{(t+1)} = \varphi_1 X_i^{(t)} + \varphi_2 X_{best}^{(t)} + \varphi_3 X_{random}^{(t)} \tag{9}$$

$X_{best}^{(t)}$ is the best particle found by PSO-NLTP form iteration 0 to t. $X_{random}^{(t)}$ is the particle formed randomly (by sub-algorithm GetOnePrimal in section 2.2) for the updating rule of $X_i^{(t)}$. φ_1, φ_2 and φ_3 are the weight terms meeting $\varphi_1 + \varphi_2 + \varphi_3 = 1$, which are calculated as Exp 4-6 show, where $f(X_i^{(t)})$ is the cost of the solution for TP (Exp. 4).

$$\varphi_1 = f(X_i^{(t)})^{-1} / \left(f(X_i^{(t)})^{-1} + f(X_{best}^{(t)})^{-1} + f(X_{random}^{(t)})^{-1} \right) \tag{10}$$

$$\varphi_2 = f(X_{best}^{(t)})^{-1} / \left(f(X_i^{(t)})^{-1} + f(X_{best}^{(t)})^{-1} + f(X_{random}^{(t)})^{-1} \right) \tag{11}$$

$$\varphi_3 = f(X_{random}^{(t)})^{-1} / \left(f(X_i^{(t)})^{-1} + f(X_{best}^{(t)})^{-1} + f(X_{random}^{(t)})^{-1} \right) \tag{12}$$

$X_i^{(t+1)}$ can be considered as a combination of $X_i^{(t)}$, $X_{best}^{(t)}$ and $X_{random}^{(t)}$ based on the their quality, and proved to meet the constraints of supply and demand.

$$\sum_{j=1}^{m} x_{ij}^{(t+1)} = \sum_{j=1}^{m} (\varphi_1 x_{ij}^{(t)} + \varphi_2 x_{i,j,best}^{(t)} + \varphi_3 x_{i,j,random}^{(t)})$$

$$= \varphi_1 \sum_{j=1}^{m} x_{ij}^{(t)} + \varphi_2 \sum_{j=1}^{m} x_{i,j,best}^{(t)} + \varphi_3 \sum_{j=1}^{m} x_{i,j,random}^{(t)}$$

$$= \varphi_1 a_i + \varphi_2 a_i + \varphi_3 a_i = a_i \quad (i = 1, \ldots, n)$$

$$\sum_{i=1}^{n} x_{ij}^{(t+1)} = \sum_{i=1}^{n} (\varphi_1 x_{ij}^{(t)} + \varphi_2 x_{i,j,best}^{(t)} + \varphi_3 x_{i,j,random}^{(t)})$$

$$= \varphi_1 \sum_{i=1}^{n} x_{ij}^{(t)} + \varphi_2 \sum_{i=1}^{n} x_{i,j,best}^{(t)} + \varphi_3 \sum_{i=1}^{n} x_{i,j,random}^{(t)}$$

$$= \varphi_1 b_j + \varphi_2 b_j + \varphi_3 b_j = b_j \quad (j = 1, \ldots, m)$$

Furthermore, the recombination rule can also ensure the positive constraint that $x_{ij}^{(t+1)} = \varphi_1 x_{ij}^{(t)} + \varphi_2 x_{i,j,best}^{(t)} + \varphi_3 x_{i,j,random}^{(t)} \geq 0, i = 1,..,n, j = 1,...,m$.

4.4 Numerical Results

There are 56 NLTP instances computed in the experiment, of which the results are shown in this section. The experiment is done at a PC with 3.06G Hz, 512M DDR memory and Windows XP operating system. The NLTP instances are generated by replacing the linear cost functions of the open problems with the non-linear functions. The methods which are effective for linear TP cannot deal with NLTP for the complexity of non-linear object function. The common NLTP cost functions are indicated in Table 1.

Problem	Transportation Cost Functions
No.1	$f_{ij}(x_{ij}) = c_{ij} x_{ij}^2$
No.2	$f_{ij}(x_{ij}) = c_{ij}\sqrt{x_{ij}}$
No.3	$f_{ij}(x_{ij}) = \begin{cases} c_{ij}(\frac{x_{ij}}{S}) , & if \ \ 0 \leq x_{ij} < S \\ c_{ij} , & if \ \ S < x_{ij} \leq 2S \\ c_{ij}(1 + \frac{x_{ij} - 2S}{S}) , & if \ \ 2S < x_{ij} \end{cases}$
No.4	$f_{ij}(x_{ij}) = c_{ij} x_{ij}[\sin(x_{ij} \frac{5\pi}{4S}) + 1]$

Table 3. NLTP cost functions [15]

The comparison between PSO-NLTP and EP with penalty strategy only indicates whether the recombination of PSO-NLTP is better at dealing with the constraints of NLTP (Exp. 8) than penalty strategy of EP. There cannot be any conclusion that PSO-NLTP or EP is better than the other because they are the algorithms for different applications. The three algorithms are computed in 50 runs independently, and the results are in Table 4 and Table 5. They would stop when no better solution could be found in 100 iterations, which is considered as a virtual convergence of the algorithms.

NLTP instances in Table 4 are formed with the non-linear functions (shown in Table 3) and the problems. And the instances in Table 5 are formed with the non-linear functions and the problems. We set $S = \sum_{i=1}^{n} a_i / 10$ in function No.3 and $S = 1$ in function No.4 in the experiment.

Problem	PSO-NLTP Average	GA Average	EP Average	PSO-NLTP Time(s)	GA Time(s)	EP Time(s)
No.1-1	8.03	8.10	8.36	0.093	0.89	0.109
No.1-2	112.29	114.25	120.61	0.11	0.312	0.125
No.1-4	1348.3	1350.8	1476.1	0.062	0.109	0.078
No.1-5	205.9	206.3	216.1	0.043	0.125	0.052
No.1-6	12.64	12.72	13.53	0.062	0.75	0.078
No.1-7	246.9	247.6	256.9	0.088	0.32	0.093
No.1-8	84.72	84.72	87.5	<0.001	0.015	<0.001
No.1-9	44.64	44.65	46.2	<0.001	0.046	<0.001
No.1-10	24.85	24.97	25.83	<0.001	0.032	<0.001
No.2-1	155.3	155.3	168.5	<0.001	0.016	<0.001
No.2-2	2281.5	2281.5	2696.2	<0.001	0.015	<0.001
No.2-4	28021	28021	30020.2	<0.001	0.015	<0.001
No.2-5	3519.3	3520.4	3583.1	<0.001	0.015	<0.001
No.2-6	264.9	266.5	314.4	<0.001	0.015	<0.001
No.2-7	4576.9	4584.5	5326.0	0.009	0.052	0.012
No.2-8	432.8	432.8	432.8	<0.001	0.015	<0.001
No.2-9	386.3	386.3	386.3	<0.001	0.031	<0.001
No.2-10	195.3	195.3	226.0	<0.001	0.006	<0.001
No.3-1	309.9	310.0	346.6	<0.001	0.093	0.001
No.3-2	4649.2	4650	5415.2	<0.001	0.921	0.012
No.3-4	65496.7	66123.3	68223.3	<0.001	0.105	<0.001
No.3-5	7038.1	7066.6	7220.9	<0.001	1.015	0.001
No.3-6	540	540	672.5	0.001	0.062	0.002
No.3-7	9171.0	9173.2	9833.3	<0.001	0.312	<0.001
No.3-8	1033.4	1033.4	1066.7	<0.001	0.012	<0.001
No.3-9	933.3	933.4	1006.4	0.002	0.147	0.015
No.3-10	480	480	480	0.016	0.046	0.004
No.4-1	107.6	107.8	118.2	0.063	0.159	0.078
No.4-2	1583.5	1585.2	1622	0.062	0.285	0.093
No.4-4	19528.4	19531.3	20119	0.075	0.968	0.068
No.4-5	2466.9	2468.2	2880.2	0.072	0.625	0.046
No.4-6	151.7	152.1	161.9	0.093	1.046	0.167
No.4-7	3171.1	3173.8	3227.5	0.047	0.692	0.073
No.4-8	467.1	467.1	467.1	<0.001	0.036	<0.001
No.4-9	376.3	376.3	382.5	<0.001	0.081	0.003
No.4-10	205.9	205.9	227.6	0.026	0.422	0.031

Table 4. Comparison I between PSO-NLTP, GA and EP with penalty strategy

Problem	PSO-NLTP Average	GA Average	EP Average	PSO-NLTP Time(s)	GA Time(s)	EP Time(s)
No.1-11	1113.4	1143.09	1158.2	0.031	0.065	0.046
No.1-12	429.3	440.3	488.3	0.187	1.312	0.203
No.1-13	740.5	740.5	863.6	0.09	2.406	0.781
No.1-14	2519.4	2529.0	2630.3	0.015	0.067	0.016
No.1-15	297.2	297.9	309.2	0.046	0.178	0.058
No.1-16	219.92	220.8	234.6	0.040	1.75	0.060
No.2-11	49.7	51.9	64.2	<0.001	0.001	<0.001
No.2-12	78.4	78.4	104.5	0.001	0.025	<0.001
No.2-13	150.2	150.4	177.9	<0.001	0.015	<0.001
No.2-14	118.6	118.2	148.4	<0.001	0.001	<0.001
No.2-15	64.5	64.5	64.5	<0.001	0.031	<0.001
No.2-16	47.1	47.8	53.4	<0.001	0.015	<0.001
No.3-11	13.3	13.3	13.3	0.015	0.734	0.031
No.3-12	21.0	21.0	26.3	0.018	0.308	0.036
No.3-13	37.2	37.4	43.5	0.171	1.906	0.156
No.3-14	37.5	37.8	46.7	0.011	0.578	0.008
No.3-15	28.3	28.1	33	0.009	0.325	0.013
No.3-16	22.5	23.0	29.6	<0.001	0.059	0.015
No.4-11	8.6	8.8	37.4	0.001	0.106	0.001
No.4-12	20.0	23.1	40.8	0.253	2.328	0.234
No.4-13	49.0	52.3	72.1	0.109	2.031	0.359
No.4-14	47.7	51.2	82.2	0.003	0.629	0.006
No.4-15	11.97	12.06	36.58	0.019	0.484	0.026
No.4-16	2.92	3.08	8.1	0.031	0.921	0.045

Table 5. Comparison II between PSO-NLTP, GA and EP with penalty strategy

As Table 4 and Table 5 indicate, PSO-NLTP performs the best of three in the items of average transportation cost and average computational cost. The NLTP solutions found by EP with penalty strategy cost more than PSO-NLTP and GA, which indicates recombination of PSO-NLTP and crossover of GA handle the constraints of NLTP (Exp. 4) better than the penalty strategy. However, EP with penalty strategy cost less time than GA to converge because the crossover and mutation operator of GA is more complicated. PSO-NLTP can cost the least to obtain the best NLTP solution of the three tested methods. Its recombination makes the particles feasible and evolutionary for optimization. The combination of updating rule and mutation operators can play a part of global searching quickly, which makes PSO-NLTP effective for solving NLTPs.

5. Discussions and Conclusions

Most of the methods that solve linear transportation problems well cannot handle the non-linear TP. An particle swarm optimization algorithm named PSO-NLTP is proposed in the present paper to deal with NLTP. The updating rule of PSO-NLTP can make the particles of the swarm optimally in the feasible solution space, which satisfies the constraints of NLTP. A mutation operator is added to strengthen the global optimal capacity of PSO-NLTP. In the experiment of computing 56 NLTP instances, PSO-NLTP performs much better than GA and EP with penalty strategy. All of the parameters of PSO-NLTP are set adaptively in the iteration so that it is good for the application of the proposed algorithm. Moreover, PSO-NLTP can also solve linear TPs.

The design of the updating rule of PSO can be considered as an example for solving optimization problems with special constraints. The operator is different from other methods such as stochastic approach, greedy decoders and repair mechanisms, which are to restrict the searching only to some feasible sub-space satisfying the constraints. It uses both the local and global heuristic information for searching in the whole feasible solution space. Furthermore, through the initial experimental result, it performs better than the penalty strategy which is another popular approach for handling constraints.

6. References

Papamanthou C., Paparrizos K., and Samaras N., Computational experience with exterior point algorithms for the transportation problem, *Applied Mathematics and Computation*, vol. 158, pp. 459-475, 2004. [1]

Vignaux G.A. and Michalewicz Z., A genetic algorithm for the linear transportation problem, *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 21, no. 2, MARCWAPRIL, pp.445-452, 2004. [2]

Hitchcock F., The distribution of a product from several sources to numerous location, *Journal of Mathematical Physics*, vol. 20, pp. 224-230, 1941. [3]

Michalewicz Z., et al, A non-Standard Genetic Algorithm for the Nonlinear Transportation Problems, *ORSA Journal on Computing*, vol. 3, no. 4, pp.307-316, 1991. [4]

Li Y.Z., Ida K.C. and Gen M., Improved genetic algorithm for solving multi objective solid transportation problem with fuzzy numbers, *Computers ind. Engng*, vol. 33, no.3-4, pp. 589-592, 1997. [5]

Gen M., et al, Solving bicriteria solid transportation problem by genetic algorithms, *Proceedings of the 16th International Conference on computers and industrial engineering*, Ashikaga, Japan, pp.572-575, 1994. [6]

Dantzig G.B., Application of the simplex method to a transportation problem, in: T.C. Koopmans (Ed.), *Activity of production and application*, John Wiley & Sons, NY, pp. 359-373, 1951. [7]

Orlin J.B., Plotkin S.A. and Tardos E., Polynomial dual network simplex algorithms, *Math. Program*, vol. 60, pp. 255-276, 1993. [8]

Paparrizos K., An exterior point simplex algorithm for general linear problems, *Ann. Oper. Res.*, vol. 32, pp. 497-508, 1993. [9]

Papamanthou C., Paparrizos K. and Samaras N., Computational experience with exterior point algorithms for the transportation problem, *Applied Mathematics and Computation*, vol. 158, pp. 459-475, 2004. [10]

Sharma R.R.K. and Sharma K.D., A new dual based procedure for the transportation problem, *European Journal of Operational Research*, vol. 122, pp. 611-624, 2000. [11]

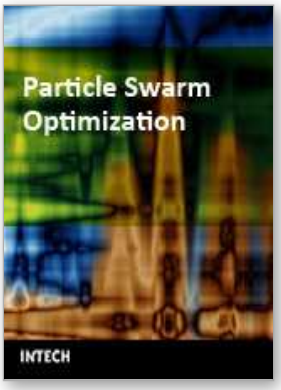
Yang X. and Gen M., Evolution program for bicriteria transportation problem, *Proceedings of the 16th International Conference on computers and industrial engineering*, Ashikaga, Japan, pp. 451-454, 1994. [12]

Kennedy J. and Eberhart R.C., Particle swarm optimization, in *Proc IEEE Int. Conf. Neural Networks*, Perth, Austalia,pp. 1942-1948, Nov. 1995. [13]

Kennedy J., The particle swarm: Social adaptation knowledge, in *Proc. 1997 Int. Conf. Evolutionary Computation*, Indianapolis, IN, pp. 303-308, Apr. 1997. [14]

Fourie P.C. and Groenwold A.A., The particle swarm optimization algorithm in size and shape optimization, *Struct Multidisc Optim*, vol. 23, pp. 259-267, 2000. [15]

Shi Y.H. and Eberhart R.C., A modified particle swarm optimizer, *Proc. Int. Conf. On Evolutionary Computation*, pp.69-73, 1998. [16]



Particle Swarm Optimization

Edited by Aleksandar Lazinica

ISBN 978-953-7619-48-0
Hard cover, 476 pages
Publisher InTech
Published online 01, January, 2009
Published in print edition January, 2009

Particle swarm optimization (PSO) is a population based stochastic optimization technique influenced by the social behavior of bird flocking or fish schooling.PSO shares many similarities with evolutionary computation techniques such as Genetic Algorithms (GA). The system is initialized with a population of random solutions and searches for optima by updating generations. However, unlike GA, PSO has no evolution operators such as crossover and mutation. In PSO, the potential solutions, called particles, fly through the problem space by following the current optimum particles. This book represents the contributions of the top researchers in this field and will serve as a valuable tool for professionals in this interdisciplinary field.

How to reference

In order to correctly reference this scholarly work, feel free to copy and paste the following:

Han Huang and Zhifeng Hao (2009). Particle Swarm Optimization Algorithm for Transportation Problems, Particle Swarm Optimization, Aleksandar Lazinica (Ed.), ISBN: 978-953-7619-48-0, InTech, Available from: http://www.intechopen.com/books/particle_swarm_optimization/particle_swarm_optimization_algorithm_for_transportation_problems

INTECH
open science | open minds

InTech Europe
University Campus STeP Ri
Slavka Krautzeka 83/A
51000 Rijeka, Croatia
Phone: +385 (51) 770 447
Fax: +385 (51) 686 166
www.intechopen.com

InTech China
Unit 405, Office Block, Hotel Equatorial Shanghai
No.65, Yan An Road (West), Shanghai, 200040, China
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元
Phone: +86-21-62489820
Fax: +86-21-62489821