Nat Comput (2016) 15:503–517
DOI 10.1007/s11047-015-9509-2

A new approach for solving set covering problem using jumping particle swarm optimization method

S. Balaji[1] · **N. Revathi**[1]

Published online: 29 July 2015

Abstract The set covering problem (SCP) is a well known classic combinatorial NP-hard problem, having practical application in many fields. To optimize the objective function of the SCP, many heuristic, meta heuristic, greedy and approximation approaches have been proposed in the recent years. In the development of swarm intelligence, the particle swarm optimization is a nature inspired optimization technique for continuous problems and for discrete problems we have the well known discrete particle swarm optimization (DPSO) method. Aiming towards the best solution for discrete problems, we have the recent method called jumping particle swarm optimization (JPSO). In this DPSO the improved solution is based on the particles attraction caused by attractor. In this paper, a new approach based on JPSO is proposed to solve the SCP. The proposed approach works in three phases: for selecting attractor, refining the feasible solution given by the attractor in order to reach the optimality and for removing redundancy in the solution. The proposed approach has been tested on the benchmark instances of SCP and compared with best known methods. Computational results show that it produces high quality solution in very short running times when compared to other algorithms.

Keywords Set covering · Swarm optimization · NP-hard problems

✉ S. Balaji
balaji_maths@yahoo.com

N. Revathi
reva17787@yahoo.co.in

[1] Department of Mathematics, SASTRA University, Thanjavur 613 401, India

1 Introduction

Particle swarm optimization (PSO) is a nature-inspired, relatively recent meta heuristic developed by Kennedy and Ebehert (1995). Like genetic algorithms, the PSO is also an optimization technique based on the population i.e. based on trope of social behavior of group of birds (or) banks of fish. Basically PSO is inspired by the continuous movement of the particles that form swarm. Aiming to discrete problems and on several adaptations to those problems, modified PSO known as discrete particle swarm optimization (DPSO) has been proposed by Kennedy and Ebehert (1997).

1.1 Particle swarm optimization

The original PSO considers a swarm S containing n particles ($S = \{1, 2, \ldots, n\}$), each particle i of the swarm has its position vector $x_i = (x_{i1}, x_{i2}, \ldots, x_{ij}, \ldots, x_{im})$ and its velocity vector $v_i = (v_{i1}, v_{i2}, \ldots, v_{ij}, \ldots, v_{im})$ in a m-dimensional continuous solution space. Initially particles positions and its velocities are obtained randomly within some limits. Each iteration, particles update their position and velocity. The position of the particle exclusively depends on its velocity i.e. when considering the kth iteration the position of the particle i is given by means of recurrent equation

$$x_i^k = x_i^{k-1} + v_i^k \tag{1}$$

But when updating the velocity, we must take into account of its velocity, attractiveness of its very own best position (b_i) and the best position (g_i) of its social neighbourhood $N(i)$ because each particle i of the swarm communicate with its social neighbourhood or environment $N(i) \subseteq S$ and

which can change dynamically. One can experience this statement practically, when considering the movement of a fish, it find its best position either by adjusting its position or by his experience and their companions experience about the position. The equation of the velocity update of a particle in the swarm according to Kennedy and Eberhart (1995) is given by

$$v_i^k = c_1\xi v_i^{k-1} + c_2\xi\left(b_i - x_i^{k-1}\right) + c_3\xi\left(g_i - x_i^{k-1}\right) \quad (2)$$

The magnitude of the velocity is controlled by the parameter c_1, represents the effect of inertia, in order to avoid the indefinite increase of the velocity. The parameters c_2 and c_3 are the positive constant weights representing the confidence degree of the particle i in different positions that regulates its movements. ξ is a random number uniformly distributed in [0, 1], generated independently at each iteration.

1.2 Discrete particle swarm optimization

The algorithm described above is the formal particle swarm optimization (PSO) method applicable only for continuous problems. Because of variety of applications of discrete problems and aiming towards their applications, Kennedy and Eberhart (1997) designed discrete binary version of the PSO, referred as discrete particle swarm optimization (DPSO) method, with discrete variables. They defined particles trajectories and velocities in terms of changes of probabilities that a bit will be in one state (or) the other. i.e. position of each particle is a vector $x_i = (x_{i1}, x_{i2}, \ldots, x_{ij}, \ldots, x_{im})$ where x_{ij} assumes the value 1 if the jth binary variable within the position of the ith particle, otherwise it assumes the value 0. However velocity of each particle is still a vector $v_i = \left(v_{i1}, v_{i2}, \ldots, v_{ij}, \ldots, v_{im}\right)$ of the m-dimensional continuous space, $v_i \in \Re^m$. Here v_{ij} indicate the probability of x_{ij} assumes a value 0 (or) 1 in the next iteration. To set the ith particle new position value, each position variable x_{ij} is randomly set with the probability of selecting a value 1 using the sigmoid function $(1 + exp(-v_{ij}))^{-1}$ where v_{ij} is restricted to some typical value $|v_{ij}| < 6.0$, prevents the probability of the x_{ij} assuming either a value of 0 (or) 1 from being too high. Kennedy and Eberhart (1997) have shown that this DPSO capable of optimizing several combinatorial optimization problems.

Few more DPSO techniques for discrete optimization problems are discussed below. Secrest (2001) presented a non-binary version of the DPSO. In this DPSO, the particles in the swarm were represented as linked list of cities and the move of the swarm operated by genetic operators mutation and recombination. Al-kazemi and Mohan (2002) developed a method by adopting the same strategy applied in Kennedy and Eberhart (1997), with the exemption that the coefficients are restricted to assume the values 1 and −1. To obtain the best solution with in the given number of iterations, they used two phase strategy. In the first phase, the coefficient (1, −1, 1) used by each ith particle, by directing the particle movement toward its social neighbourhood best position g_i. In the second phase, the coefficient (1, 1, −1) used by each ith particle, by directing the particle movement toward its best position b_i.

Yang et al. (2004) considered a larger number of combinations of the coefficients, which were referred as quantum states of the particles and they applied principles of quantum computing to update the velocity of the particle. Martinoli (2006), presented a multi-valued PSO (MVPSO). It is described with variables with mulitiple discrete values. In this PSO, position of each particle expressed by means of 3 dimensional array x_{ijk} represents the probability that the ith particle, in the jth iteration, assumes the kth value. Sigmoid distribution used to generate the elements x_{ijk} and they followed the Eqs. (1) and (2).

Another DPSO algorithm was developed by Correa et al. (2006) to tackle the data mining task of attribute selection, in order to classify data sets into classes (or) categories of the same type. This DPSO slightly differs from other PSO by means of swarm which contains particles representing combinations of selected attributes of different size, varies from 1 to λ, the total number of attributes. Other than that of best position of particle and best position among its neighbours, one more factor length of the particle is introduced. The new length of the particle is determined by selecting random number $k \in [0, \lambda]$ and finally new position of the particle is updated by the k attributes with the largest likelihood in the velocity vector.

Tasgetiren et al. (2007) presented a DPSO to solve the generalized traveling salesman problem. In this DPSO, particles best position are updated using three operators which were presented in Pan and Tasgetiren (2008) and global neighbourhood best position updated by the same model applied in Kennedy and Eberhart (1997). Further to improve the solution quality they hybridized the DPSO with a variable neighbourhood descend local search. Liaoa et al. (2007) presented DPSO for flow shop scheduling problem. This DPSO designs "job-to-position" representation for the discrete particle. They define the position and velocity of the ith particle position in the tth iteration as $X_i^t = \left(x_{i11}^t, x_{i12}^t, \ldots, x_{inn}^t\right)$, $x_{ijk} \in \{0, 1\}$, where x_{tijk} equals 1 if job j of particle i is placed in the kth position of the sequence and 0 otherwise and $V_i^t = \left(v_{i11}^t, v_{i12}^t, \ldots, v_{inn}^t\right)$, $v_{ijk}^t \in R$, where v_{ijk}^t is the velocity value for job j of particle i placed in the kth position at iteration t respectively. The velocity V_i^t, called velocity trail, is

inspired by the frequency-based memory which records the number of times that a job visits a particular position.

Al-Sherbaz et al. (2010) used the DPSO method described in Kennedy and Eberhart (1997) for Wimax parameter adaptation through baseband processor, in this DPSO the solution space is the number of bits to represent a particle x_i^t, This number of bits is determined by the range and the necessary precision of the optimized parameters. Shi et al. (2007) presented a novel DPSO based algorithm for Traveling Salesman Problem. In this DPSO, they introduced a permutation and also an uncertainty searching strategy to speed up the convergence speed. The ith particle position $x_i = (x_{i1}, x_{i2}, \ldots, x_{ij}, \ldots, x_{im})$ represents the traveling circle of $x_{i1} \rightarrow x_{i2} \rightarrow \cdots, \rightarrow x_{im} \rightarrow x_{i1}$. Therefore they used the (1) as it is but in (2) they considered the ξ as a real vector whose dimensions are corresponds to the number of transpositions dotted with them. Chen et al. (2006) developed a new hybrid approximation algorithm based on DPSO, this combines global search and local search to search for the optimal results and they also applied simulated annealing to avoid local optima and hence they updated the Eq. (2) using some control parameters. More DPSO based techniques are referred in Aliguliyev (2010), Qiang et al. (2009), Wang and Yang (2007), Zhang et al. (2007).

1.3 Jumping particle swarm optimization

García and Pérez (2008) introduced a new DPSO technique for discrete optimization. In this DPSO particles best position (or) improved solution based on the attraction caused by attractor, one can found this inspiration in frogs in nature; this method is called jumping particle swarm optimization (JPSO). This JPSO technique works without the components of velocity and inertia due to lack of continuous movement in the discrete space. These two are the main components in the original PSO technique. Instead of these components, a random component in the form of jumps is included for the movement of the particles. The position of the particles updated as similar to the velocity update in the general PSO, but in which weights of the update equation are interpreted as probabilities of the movement of the particles by random (x_i) or by guided attraction of its own position (b_i) or by attraction of best position of its social neighbourhood (g_i) or by attraction of the best global position (g^*). These attractions cause some improvement in the position of the particles. The update equation of particles position is given by

$$x_i = c_1 x_i + c_2 b_i + c_3 g_i + c_4 g*$$

where c_1, c_2, c_3 and c_4 are the probability values of the movement of the particles towards their corresponding attracted positions. For the probability values the unit interval [0, 1] is divided into four segments with lengths c_1, c_2, c_3 and $c_4 = 1 - (c_1 + c_2 + c_3)$. A random number generated corresponding to the segments and based on the random number which belongs to the segment, random improvement movement towards the attractor are applied to the position of the particle. The moves by the attraction that do not produce improvement are rejected. After each movement, the number generated at random is multiplied by p and by corresponding coefficients c_i. If this product is greater than 1 a new move is applied, otherwise the movement stops until the next generation, and this move is rejected if it does not give improvement to its position. In addition, after each random or attractor movement, a local search is applied to every particle in the swarm. This local search approach explores set of possible moves starting from random one and the first move found that improve the particles current position. This improvement move stops if it does not give improvement to the current position of the particles.

This new methodology is successfully applied to the vehicle routing problem with time windows (Gutiérrez et al. 2008) and to the minimum labelling steiner tree problems (Consoli et al. 2010). This paper aims to apply JPSO to the Set covering problems (SCP), a well known combinatorial optimization problems, not only due to its spirit of nature but also due to its simplicity, easy implementation, less computational cost and time. Further we wish to compare this method with the other algorithms proposed for SCP to find out its effectiveness for solving SCP.

The rest of the paper is as follows: Sect. 2 explains the set covering problem. Section 3 discusses briefly on different SCP algorithms so far found in the literature. The pseudo-code and description of the proposed Jpso–scp algorithm and brief description of algorithms considered for comparison are discussed in Sect. 4. In Sect. 5 detailed experimental study and computational results are given and finally Sect. 6 ends with our conclusion.

2 Set covering problem

The set covering problem is a well-known combinatorial optimization problem with variety of real time applications in different fields, in particular crew scheduling in railway and airlines (Housos and Elmoth 1997), location problem of facility (Vasko and Wilson 1984) and in industry production planning (Vasko and Wolf 1987). The description of the SCP is as follows:

Given a finite set $I = \{1, 2, \ldots, m\}$ of m elements and the family $J = \{S_1, S_2, \ldots, S_n\}$ of subsets of I such that a non-negative cost c_j is associated with each subset S_j. The

set covering problem is to find the minimum size subset $C \subseteq J$ such that all the members of I covered by the members of C with minimum total cost. i.e. for each $i \in I$, there exist at least one $S_j \in J$ such that i is covered by S_j. Let $A = (a_{ij})$ be the $m \times n$ matrix whose jth column is the characteristic vector of the subset S_j. i.e. $a_{ij} = 1$ if i is covered by S_j or otherwise $a_{ij} = 0$. Then the 0–1 integer programming formulation of SCP is

$$\text{Min} \sum_{j=1}^{n} c_j x_j \quad \text{subject to}$$

$$\sum_{j=1}^{n} a_{ij} x_j \geq 1, \quad i = 1 \text{ to } m.$$

$$x_j \in \{0, 1\}, \quad j = 1 \text{ to } n.$$

The variable x_j takes the value 1 if the subset S_j is selected into the set cover and 0 otherwise. A particular version of the problem was introduced by Toregas et al. (1971), called unicost set covering problem (USCP) or minimum cardinality set covering problem (MCSCP) or location set covering problem (LSCP). In this version the associated cost c_j, for all the elements in J, is equal and all may considered be equal to 1. It is well known that SCP and USCP are NP-hard (Garey and Johnson 1979) and are therefore difficult to solve in the case of large set of instances. A probabilistic version of the set covering problem is addressed in Saxena et al. (2010).

3 Related works

Many exact and heuristic procedures have been developed to solve the SCP. Exact procedure is mainly based on branch and bound (Balas and Carrera 1996), branch and cut and Gomory cut (Fisher and Kedia 1990). Still these procedures are able to solve very limited size instances and also consuming very large amount of time. Because of this many researchers focused on the developing heuristics to get optimal or near optimal solutions in a reasonable amount of time.

The following are the list of some metaheuristic approaches proposed for the SCP or unicost SCP. A very clear literature survey till 2000 on both heuristic and exact approaches has been presented in Caprara et al. (2000). This survey outlined their main characteristics and presented an experimental comparison on the test-bed instances of Beasley's OR Library. Ceria et al. (1998) presented a heuristic based on Lagrangian method for solving large-scale SCP developed from the problem of crew-scheduling at the Italian Railways. Brusco et al. (1999) developed a heuristic for the SCP based on morphing procedure in a simulated annealing (SA). Yagiura et al. (2006) proposed a local search algorithm for SCP with three characteristics and they claimed through computational experience with other existing heuristic algorithms that their algorithm performed quite effectively for large-scale SCP instances. In Li and Kwan (2004), one can see fuzzy evaluation based evolutionary technique for large-scale SCP originated from the public transport industry. Simple solutions of random SCP instances through the average case analysis are discussed in Telelis and Zissimopoulos (2005) and solutions are constructed through an O(nm) algorithm. A Meta-RaPS (Meta-heuristic for Randomized Priority Search) approach to solve the SCP is discussed by Lan et al. (2007) investigates the development of an effective heuristic to solve the SCP and they claimed that their approach finds good quality solution for unicost SCP among compared heuristics. GRASP algorithm to solve unicost set covering problem presented in Bautista and Pereira (2007). This algorithm incorporates a local improvement procedure based on the heuristics to solve binary constraint satisfiability problems (SAT). Azimi et al. (2010) proposed a heuristic algorithm based on the electromagnetism metaheuristic approach to solve the unicost SCP. In this method a local search and iterations movement has been applied using "electromagnetism" theory to generate a pool of solutions from the initial population. Ren et al. (2010) proposed a new approach based on ant colony optimization (ACO) to solve the SCP. In this approach when choosing a new column, it first randomly selects an uncovered row and only considers the columns covering this row, rather than all the unselected columns as candidate solution components. Then a kind of dynamic heuristic information called Lagrangian dual information associated with currently uncovered rows. Finally, a simple local search procedure is developed to improve solutions constructed by ants while keeping their feasibility.

When considering Genetic algorithm approaches for the SCP, we can give importance to the following: Beasley and Chu (1996) developed a genetic algorithm approach for SCP. The method in this procedure modifies the basic genetic procedures by including a fitness-based crossover operator, a variable mutation rate and a heuristic feasibility operator and using these procedures they have designed an optimal algorithm for SCP. Through computational results they have claimed that their approach is capable of finding good quality solution for the SCP in reasonable time. Aickelin (2002) presented a new type of indirect genetic algorithm for the set covering problem. In this approach actual solutions are found by an external decoder function, results can be further improved by adding another indirect optimization layer and then optimized by another hill-

climbing algorithm. A parallel genetic algorithm (PGA) model to solve the set-covering problem is presented in Solar et al. (2002). An experimental study obtained with a binary representation of the SCP, shows that PGA performs better than the sequential model in terms of the number of generations (computational time) needed to achieve solutions.

Li and Cai (2012), Balachander and Kannan (2010) developed gravitational search algorithm (GSA) for solving SCP. This GSA technique is one of the nature based technique like genetic algorithm, simulated annealing and ant colony optimization. It comprises a population based technique induced by Newton law of gravity and the law of motion. The main frame work of the algorithm is: objects are considered with different masses. Every mass can see the position of the other masses and the gravitational force transfers the information among the different masses. The heavy masses corresponds to good solution of the problem and the different masses attracted by the heaviest mass which would give optimum solution in the search space.

Other than these approaches an artificial neural network algorithm is developed by Ohlsson et al. (2001) for SCP based on the mean field feedback procedure. In this approach inequality constraints are encoded in convenient way using a multi-linear penalty function. They claimed through computational results that their algorithm outperformed other approximate methods. Galinier and Hertz (2007) developed three exact algorithms for large-scale SCP. Cormode et al. (2010) presented an algorithm for SCP using modern disk technology and they claimed through computational experiments that their algorithm finds good solutions for larger datasets.

Computational experience of different approximation algorithms with the SCP were given in Grossman and Wool (1997), Pardalos et al. (2006). In which the first paper conducted a comparative study of nine different approximation algorithms for the SCP which includes greedy variants, fractional relaxations, randomized algorithms and a neural network algorithm. Through computational experience on random problems, they identified that the best algorithm among those tested was a randomized greedy algorithm, with the neural network algorithm which is very close in second place. The second paper conducted an empirical study of approximation algorithms of Vertex Cover and the SCPs and produces their strengths, weaknesses, and operation. They have shown through computational experiments that the proven performance of all tested algorithms that did not forecast the empirical performance.

The following are the two problems, which are slightly associated with set covering problem and for these problems PSO based technique has been proposed. Zhan et al. (2012) proposed a binary particle swarm optimization (BPSO) approach to solve the disjoint set covers problem (DSC) in the wireless sensor networks. The DSC problem plays a vital role in sensor networks to increase the life length of sensor nodes. The DSC problem is to divide the sensor nodes into different disjoint sets and schedule them to work one by one in order to save energy while at the same time it should meet the full coverage, and the objective is to maximize the number of disjoint sets. Through simulation and computational results they claimed that the BPSO technique performed well over other approaches in maximizing the disjoint set covers. Moirangthem et al. (2012) proposed an approach based on PSO to identify the break point set in directional loops and multiloop settings and coordination of directional relays in system protection. In this approach they have converted the problem into set covering problem and then determined the break point set using PSO based technique.

4 Algorithms considered

This section describes the algorithms that we consider for the Set Covering problem: Ant_cover + *ls*, an ACO based approach by Ren et al. (2010); Meta-RaPS, a Meta-heuristic by Lan et al. (2007); IGA, an indirect genetic algorithm by Aickelin (2002) and finally the proposed Jpso–scp algorithm based on the Jumping particle swarm optimization.

4.1 Ant_cover + *ls* method

This Ant_cover + *ls* approach to the SCP follows the standard algorithmic scheme of Max–Min Ant system by Stützle and Hoos (2000) with some new features. Strategies used in this scheme include solution construction method, heuristic scheme, phenomerone update rule and a local search procedure. In the solution construction part each ant starts with an empty solution and add columns iteratively until all the rows are covered.

At a step t, an ant chooses column j from all the unselected columns according to probabilistic state transition rule. For this they must use all the possible combinations of state transition rule with the number of columns, and due to this factor its computational running time grows exponentially. Some heuristic rules based on the Lagrangian relaxation (Wolsey 1998) and local search approaches used in order to reduce the running time and computational cost. After running all the iterations, the best solution to date represents the output of the method.

4.2 Meta-RaPS method

This Meta-RaPS approach follows a meta-heuristic developed by Whitehouse et al. (2002). The main work of Meta-RaPS is the use of randomness as a mechanism to avoid local optima. At each iteration it constructs a feasible solution through the utilization of construction heuristic in randomized fashion, and then applies an improvement heuristic to improve the feasible solution.

In the SCP construction heuristic, to select a column for the feasible solution, a priority rule is applied. A greedy score for each column is calculated using some parameter and a column with lower greedy score gets highest priority. After this to improve the solution quality an improvement heuristic based on neighbourhood search procedure is applied. To improve the computational speed a preprocessing method and neighbor search procedure applied. After a number of iterations a best solution is then reported.

4.3 IGA method

The IGA approach to the SCP is a self tuning genetic algorithm and it solves the problem indirectly. This algorithm differs from previous evolutionary approaches by taking indirect route. This can be done by splitting the search into three distinct phases. Initially, the genetic algorithm finds permutations of the rows to be covered along with suitable parameters for the second stage.

The second stage consists of a decoder that builds a solution from the permutations using some provided parameters. These procedures built only exploitable solutions and the obtained solutions further fine tuned by hill-climbing approach. Over all more complicated strategies are applied to get an optimal solution.

4.4 Proposed Jpso–scp method

Even though repetitive advancements in computing, we are to be very wondered by the variety and adaptability of the natural world around us. Bio-inspired optimization and computing techniques covers wide variety of computational approaches motivated from the application of biology to optimization problems and is one of the major subset of natural computation. Making the above statement is worthy, a discrete PSO named Jpso–scp, based on the Jumping particle swarm optimization proposed by García and Pérez (2008), is chosen to deal with the set covering problem. The main interest of using this procedure is not only due to its spirit of nature but also due to its simplicity, easy implementation, low computational cost and time.

The procedure of the proposed Jpso–scp algorithm for the SCP is as follows: a swarm S, containing n particles (random solutions), in the JPSO procedure is considered as initial position of the particles in Jpso–scp too. Particles position x_i (at some iteration i) develop in the solution space, jumping from one solution to another. By the effect of some attractors, at each iteration each particle has a random behaviour or random jumps. The position of a particle encoded as a feasible solution to the SCP. Movement of the particle i is influenced by three attractors:

1. Its own best position to date (b_i)
2. The best position of its social neighbourhood (g_i)
3. The best position to date obtained by all the particles which is called global best position (g^*)

A jump towards the attractor based on the current solution feature and the feature of the attractor and a random jump is based on selecting at random feature of the solution and changing its value. For the SCP, features of a solution are the columns, covers maximum number of rows, to be included in the solution. A particle performs a jump with respect to the selected attractor means by randomly adding some columns to its current position from the selected attractor, or dropping some columns from its current position that are not included in the attractor.

Pseudo-code of the Jpso–scp algorithm given in Algorithm 4.41

Algorithm 4.41: (Jpso-scp)

Input: $m \times n$ matrix of the SCP instance.

Output: A set cover C.

Intialization:

- Let $C \leftarrow \emptyset$ be a set of columns, initially empty set.
  Let
- I: the set of all rows
- J: the set of all columns
- c_i: the set of all columns that cover row i, $i \in I$
- r_j: the set of rows covered by column j, $j \in J$
- n_i: the number of columns that cover row i
- C: the set of columns in a solution
- Set the size n_S of the swarm S.

begin

- Generate the initial swarm S with positions at random;
- $X = [x_0, x_1, \cdots, x_{n_S}] \leftarrow$ Generate-Swarm-At-Random(G);
- Update the vector of the best positions $B = [b_0, b_1, \cdots, b_{n_S}] \leftarrow X$
- Extract the best position among all the particles; $g^* \leftarrow$ Extract-The-Best (S, X);

repeat

$for\ \ i \leftarrow 1$ to $n_S\ \ do$

If $i = 1$ then

- Initialize the best position of the social neighborhood: $g_i \leftarrow (all\ columns)$

else

- Update the best position of the social neighborhood: i: $g_i \leftarrow g_{i-1}$;

end

- Select at random a number between 0 and 1: β = Random(0,1);
  If $\alpha \in [0,0.25[$ then

  - $selected \leftarrow x$;

  $else\ if\ \alpha \in [0.25,0.5[$ then

  - $selected \leftarrow b_i$;

  $else\ if\ \alpha \in [0.5,0.75[$ then

  - $selected \leftarrow g_i$;

  $else\ if\ \alpha \in [0.75,1[$ then

  - $selected \leftarrow g^*$;

  Combine particle i and the selected particle;
  $x_i \leftarrow combine(x_i, selected)$
  $local\ search(i, x_i)$;

$if\ |x_i| < b_i$ then

- Update the best position of the given particle i: $b_i \leftarrow x_i$

end

$if\ |x_i| < g_i$ then

- Update the best position of the social neighborhood of i: $g_i \leftarrow x_i$

end

$if\ |x_i| < b_i$ then

- Update the global position to date i: $g^* \leftarrow x_i$

end

end

until termination condition;

- Set $C \leftarrow g^*$

end

Algorithm 4.41.1: Procedure $merge(x_i, selected$ attractor)

- Select a random integer between 0 and $|x_i|$;
- $\mu \leftarrow$ Random $(0, |x_i|)$;

for $j \leftarrow 1\ to \mu\ \ do$

- Select at random number between 0 and 1;
- $\beta \leftarrow$ Random $(0, 1)$;

if $\beta \leq 0.5$ then

- Select at random a column $c' \in x_i$;
- Delete the column c' from the position of the given particle: $x_i \leftarrow x_i - \{c'\}$;

else

- Select at random a column $c' \in selected$ attractor;
- Add label c' to the position of the given particle i: $x_i \leftarrow x_i \cup \{c'\}$;

end

end

- $n_i = |C \cap c_i|, \forall\ i \in I$
- $U = \{\ i\ /\ n_i = 0, \forall\ i \in I\ \}$
- If $U \neq \emptyset$

- then for each row i in U
- find the first column j (in increasing order) in c_i such that which minimizes $co_j / |U \cap r_j|$
- $C \leftarrow C \cup \{j\}$ and $n_i = n_i + 1,\ \forall i \in r_j$
- $U \leftarrow U - r_j$

else do no operations in C

- for each j $\in C$, if $n_i \geq 2\ \ \forall i\ \in r_j$

  then $S \leftarrow S - \{j\}$ and set $n_i = n_i - 1\ \ \forall i \in r_j$

- C is now feasible solution

Algorithm 4.41.2: Procedure $local\ search(i, x_i)$

Initialization:

- N' - A sequence in which columns of xi are arranged in the increasing order of cost, at the same time columns with equal cost are arranged in the decreasing order of the number of rows that they cover
- $n_i = |C \cap c_i|, \forall i \in I$
- $minC_i$ – column with the minimal cost among all the columns covering row i

for each column $j \in x_i$ (in the reverse order of N')

do

- $R_j = \{i / n_i = 1\ \forall i \in r_j\}$ is the set of rows only covered by column j
- **if** $|R_j| = 0$
  then $C \leftarrow C - \{j\}$ and set $n_i = n_i - 1\ \ \forall i \in r_j$
- **else if** $|R_j| = 1$ and $Co_j \neq minC_{i1}$
  then $C \leftarrow C - \{j\} \cup minC_{i1}$, set $n_i = n_i - 1\ \ \forall i \in r_j$ and $n_i = n_i + 1\ \ \forall i \in r_{minC_{i1}}$
- **else if** $|R_j| = 2$ and $Co_j \neq minC_{i1} = minC_{i2}$
  then $C \leftarrow C - \{j\} \cup minC_{i1}$, set $n_i = n_i - 1\ \ \forall i \in r_j$ and $n_i = n_i + 1\ \ \forall i \in r_{minC_{i1}}$
- **else if** $|R_j| = 2$ and $Co_j \neq minC_{i1} \neq minC_{i2}$ and $Co_{minC_{i1}} + Co_{minC_{i2}} \leq Co_j$
  then $C \leftarrow C - \{j\} \cup minC_{i1} \cup minC_{i2}$, set $n_i = n_i - 1\ \ \forall i \in r_j$ and $n_i = n_i + 1$ $\forall i \in r_{minC_{i1}}$ and $\forall i \in r_{minC_{i2}}$

end if (do no operations if $|R_j| \geq 3$)

end for

Return the solution *C*.

Description of the Jpso–scp follows:

Initial position of the swarm S are generated by using the initial population algorithm given by Beasley and Chu (1996). According to population based model, this is the best choice for the initial positions of the swarm. Here x_i represent one random feasible solution. In the swarm containing n particles (n random solutions), the ith particle position x_i is a 0–1 vector denoting which columns are present in the particle i. Particles positions are updated in every iteration. When considering the kth iteration, ith particle obtain a new position x_i^k from its previous position x_i^{k-1} by using the following update equation

$$x_i^k = c_1 x_i^{k-1} + c_2 b_i + c_3 g_i + c_4 g*$$

i.e. position x_i^k is obtained by making random moves from its current position x_i^{k-1} with probability c_1, approaching

jumps towards b_i with probability c_2, towards g_i with probability c_3, or towards g^* with probability $c_4 = 1 - (c_1 + c_2 + c_3)$. In the SCP, by giving equally likely probability for each of the jumps, the values of c_1, c_2, c_3 and c_4 are set to 0.25. i.e. a random number α between 0 and 1 is generated and if α belongs to $[0, 0.25[$, the particle perform a random jump from its current position x_i. Otherwise if α belongs to $[0.25, 0.5[$, the movement of the particle x_i aiming towards the attractor b_i. Instead, if α belongs to $[0.5, 0.75[$, the particle x_i attractor is selected as g_i for its movement. For the remaining case, i.e. if α belongs to $[0.75, 1[$, g^* is selected as attractor for the movement of the particle x_i.

When the ith particle making the jump towards the selected attractor, the particle x_i either drops some columns from x_i, or randomly adds some columns from the selected attractor based on the procedure in Algorithm 4.41.1. In this procedure to make the solution feasible, we make use of the concept of heuristic feasibility operator (Beasley and Chu 1996), which not only maintains the feasibility of the solutions but also refines the solution towards the optimality. Adding or dropping of some columns of x_i in Algorithm 4.41.1 make use of the following heuristic procedure. In this procedure a random integer μ is selected between 0 and $|x_i|$. Successively, it either drops some columns from x_i or adds some columns from selected attractor until μ columns have been added or deleted from the x_i. After this phase, the set of uncovered rows by x_i are updated. If it is identified, a greedy heuristic approach is applied in order to maintain the feasibility of the solution. i.e. for each uncovered row, correspondingly a column with low cost ratio included and to maintain the optimality a local optimization technique is applied to remove any redundant columns in the solution.

Further to refine the updated solution, as a final phase of the Jpso–scp, a local search procedure developed in Ren et al. (2010), known to be best for SCP, is applied in order to remove redundant columns from x_i and replace them with associated low cost columns that cover corresponding rows at the same time that retaining the feasibility. i.e. this procedure try to remove high cost first whenever it is possible. The detailed description of this phase follows: For each solution C constructed by Jpso–scp, let n_i be the number of columns covering a row i, $i \in I$ and it is ≥ 1 because of the feasibility of the solution. For each $j \in C$, let $R_j = \{i / n_i = 1 \ \forall i \in r_j\}$ be the set of rows only covered by column j. If $|R_j| = 0$, it implies that column j is redundant and it can be removed directly. On the other side if $|R_j| = l > 0$, there are l number of rows covered by column j exclusively and in this case the cost of the column j is compared with total cost of all the columns $minC_i$, $i \in R_j$ where $minC_i$ is the column with the minimal cost among all the columns covering row i. If the cost of the column j is greater, the column j is replaced by all the columns $minC_i$. After these add or drop phase of the columns to the solution C, the value of n_i is updated. As explained in Solar et al. (2002), no operation is done when $|R_j| \geq 3$ because of less improvement in the solution quality and the relatively high computational cost, pseudo-code of this local search procedure given in the Algorithm 4.41.2.

5 Experimental studies

To test the performance of the proposed heuristic, it has been evaluated experimentally on the 65 benchmark instances of SCP from Beasley's OR Library (Beasley 1990). These benchmark instances classified into 11 groups depends on the number of instances, rows and columns and density. Here density represents the number of non-zero entries in the SCP matrix. This information is in the Table 1 in detail. All the experiments were carried out on

Table 1 Brief summary of SCP test instances

Set	No. of instances	No. of rows (m)	No. of columns (n)	Cost interval	Density (%)	Maximum no. of 1s per row	Problem type	Optimal solution
4	10	200	1000	[1, 100]	2	36	Random	Known
5	10	200	2000	[1, 100]	2	60	Random	Known
6	5	200	1000	[1, 100]	5	71	Random	Known
A	5	300	3000	[1, 100]	2	81	Random	Known
B	5	300	3000	[1, 100]	5	192	Random	Known
C	5	400	4000	[1, 100]	2	105	Random	Known
D	5	400	4000	[1, 100]	5	244	Random	Known
NRE	5	500	5000	[1, 100]	10	124	Random	Unknown
NRF	5	500	5000	[1, 100]	20	561	Random	Unknown
NRG	5	1000	10,000	[1, 100]	2	266	Random	Unknown
NRH	5	1000	10,000	[1, 100]	5	572	Random	Unknown

Table 2 Test results for each instance (mCPU = 500 s)

Inst.	Opt.	IGA		Meta-RaPS		Ant_cover + ls		Jpso–Scp	
		O_{avg}	T_{avg} (s)	O_{avg}	T_{avg} (s)	O_{avg}	T_{avg} (s)	O_{avg}	T_{avg} (s)
4.1	**429**	**429.0**	1.62	**429.0**	0.07	**429.0**	0.02	**429.0**	0.01
4.2	**512**	**512.0**	2.23	**512.0**	0.15	**512.0**	0.04	**512.0**	0.02
4.3	**516**	**516.0**	2.13	**516.0**	0.26	**516.0**	0.12	**516.0**	0.06
4.4	**494**	**494.0**	2.14	**494.0**	0.83	**494.0**	0.57	**494.0**	0.17
4.5	**512**	**512.0**	2.38	**512.0**	0.10	**512.0**	0.04	**512.0**	0.04
4.6	**560**	**560.0**	2.54	**560.0**	0.23	**560.0**	0.10	**560.0**	0.04
4.7	**430**	**430.0**	1.66	**430.0**	0.08	**430.0**	0.02	**430.0**	0.01
4.8	**492**	**492.0**	2.02	**492.0**	0.07	**492.0**	0.03	**492.0**	0.01
4.9	**641**	**641.0**	2.21	**641.0**	0.57	**641.0**	0.32	**641.0**	0.17
4.10	**514**	**514.0**	1.85	**514.0**	0.12	**514.0**	0.04	**514.0**	0.02
5.1	**253**	**253.0**	2.29	**253.0**	0.52	**253.0**	0.29	**253.0**	0.17
5.2	**302**	**302.0**	2.34	**302.0**	1.32	**302.0**	0.76	**302.0**	0.35
5.3	**226**	**226.0**	2.09	**226.0**	0.08	**226.0**	0.05	**226.0**	0.02
5.4	**242**	**242.0**	2.18	**242.0**	0.07	**242.0**	0.03	**242.0**	0.01
5.5	**211**	**211.0**	1.57	**211.0**	0.04	**211.0**	0.02	**211.0**	0.01
5.6	**213**	**213.0**	1.78	**213.0**	0.09	**213.0**	0.02	**213.0**	0.01
5.7	**293**	**293.0**	2.12	**293.0**	0.06	**293.0**	0.03	**293.0**	0.01
5.8	**288**	**288.0**	2.33	**288.0**	0.04	**288.0**	0.02	**288.0**	0.01
5.9	**279**	**279.0**	2.20	**279.0**	0.21	**279.0**	0.07	**279.0**	0.02
5.10	**265**	**265.0**	2.02	**265.0**	0.10	**265.0**	0.04	**265.0**	0.02
6.1	**138**	**138.0**	2.65	**138.0**	0.68	**138.0**	0.33	**138.0**	0.18
6.2	**146**	**146.0**	2.76	**146.0**	0.10	**146.0**	0.05	**146.0**	0.02
6.3	**145**	**145.0**	2.94	**145.0**	0.32	**145.0**	0.19	**145.0**	0.08
6.4	**131**	**131.0**	2.53	**131.0**	0.09	**131.0**	0.03	**131.0**	0.01
6.5	**161**	**161.0**	2.97	**161.0**	0.48	**161.0**	0.26	**161.0**	0.12
A.1	**253**	253.3	4.00	**253.0**	2.08	**253.0**	1.28	**253.0**	0.98
A.2	**252**	252.2	4.14	**252.0**	1.96	**252.0**	1.23	**252.0**	0.81
A.3	**232**	232.4	4.13	**232.0**	1.75	232.8	0.96	**232.0**	0.53
A.4	**234**	234.7	4.06	**234.0**	0.37	**234.0**	0.18	**234.0**	0.04
A.5	**236**	236.8	4.08	**236.0**	0.86	**236.0**	0.56	**236.0**	0.23
B.1	**69**	**69.0**	6.21	**69.0**	0.38	**69.0**	0.15	**69.0**	0.06
B.2	**76**	**76.0**	6.32	**76.0**	0.49	**76.0**	0.17	**76.0**	0.07
B.3	**80**	**80.0**	6.66	**80.0**	0.38	**80.0**	0.10	**80.0**	0.03
B.4	**79**	**79.0**	6.45	**79.0**	0.39	**79.0**	0.18	**79.0**	0.03
B.5	**72**	**72.0**	6.13	**72.0**	0.11	**72.0**	0.04	**72.0**	0.01
C.1	**227**	227.1	7.62	227.4	1.34	**227.0**	0.75	**227.0**	0.43
C.2	**219**	219.3	7.15	219.3	1.04	**219.0**	0.53	**219.0**	0.21
C.3	**243**	243.2	8.93	243.1	3.85	**243.0**	2.01	243.2	0.98
C.4	**219**	219.3	8.91	219.2	1.03	**219.0**	0.43	**219.0**	0.15
C.5	**215**	215.2	7.73	215.3	1.21	**215.0**	0.49	**215.0**	0.20
D.1	**60**	**60.0**	8.32	**60.0**	1.23	**60.0**	0.56	**60.0**	0.23
D.2	**66**	**66.0**	9.43	**66.0**	0.96	**66.0**	0.35	**66.0**	0.16
D.3	**72**	**72.0**	9.34	**72.0**	2.00	**72.0**	1.04	**72.0**	0.56
D.4	**62**	**62.0**	9.85	**62.0**	2.12	**62.0**	1.32	**62.0**	0.69
D.5	**61**	**61.0**	8.20	**61.0**	1.07	**61.0**	0.49	**61.0**	0.18
NRE.1	**29**	29.8	18.86	29.2	0.75	**29.0**	0.27	**29.0**	0.11
NRE.2	**30**	31.6	20.32	30.3	3.28	**30.0**	2.14	**30.0**	1.06

Table 2 continued

Inst.	Opt.	IGA		Meta-RaPS		Ant_cover + ls		Jpso–Scp	
		O_{avg}	T_{avg} (s)	O_{avg}	T_{avg} (s)	O_{avg}	T_{avg} (s)	O_{avg}	T_{avg} (s)
NRE.3	**27**	28.9	18.46	27.4	3.14	**27.0**	2.00	**27.0**	1.10
NRE.4	**28**	29.5	20.73	28.2	2.87	**28.0**	1.56	**28.0**	0.76
NRE.5	**28**	29.4	20.64	28.1	1.04	**28.0**	0.34	**28.0**	0.12
NRF.1	**14**	14.8	27.16	14.1	1.92	**14.0**	0.78	**14.0**	0.32
NRF.2	**15**	15.7	22.67	15.2	1.24	**15.0**	0.52	**15.0**	0.21
NRF.3	**14**	14.4	28.45	14.3	3.03	**14.0**	1.14	**14.0**	0.86
NRF.4	**14**	14.6	23.67	**14.0**	3.27	**14.0**	1.23	**14.0**	0.75
NRF.5	**13**	14.2	22.67	13.8	6.23	**13.5**	4.78	**13.0**	2.95
NRG.1	**176**	177.1	30.12	176.3	15.70	**176.0**	12.12	**176.0**	6.15
NRG.2	**154**	156.3	27.66	154.8	18.66	155.1	14.64	**155.0**	7.18
NRG.3	**166**	167.3	30.34	166.7	23.62	167.3	18.56	167.2	8.96
NRG.4	**168**	169.5	27.84	168.6	20.73	168.9	15.62	168.2	9.72
NRG.5	**168**	169.3	27.12	168.7	15.77	168.1	11.72	**168.0**	6.23
NRH.1	**63**	64.1	61.12	63.8	39.64	**64.0**	34.65	**64.0**	15.24
NRH.2	**63**	67.6	51.13	63.9	28.73	63.9	25.83	**63.0**	11.86
NRH.3	**59**	59.8	65.19	59.6	36.94	59.4	29.68	59.2	13.67
NRH.4	**58**	59.7	60.66	58.8	35.63	58.7	26.82	58.3	14.83
NRH.5	**55**	55.8	68.87	55.6	18.73	**55.0**	14.73	**55.0**	6.53

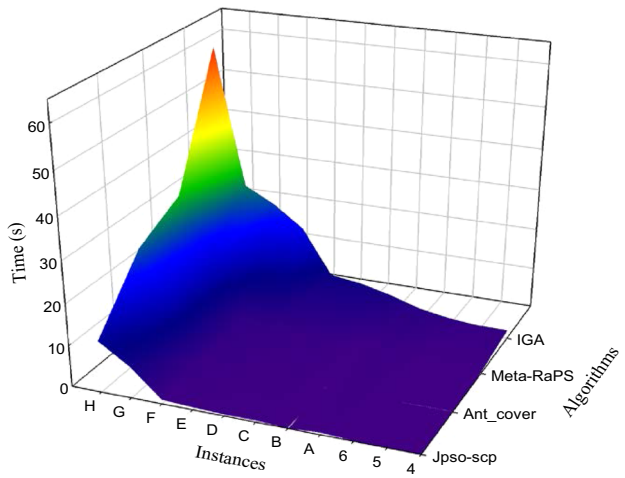

Fig. 1 Average time taken for each set of instances

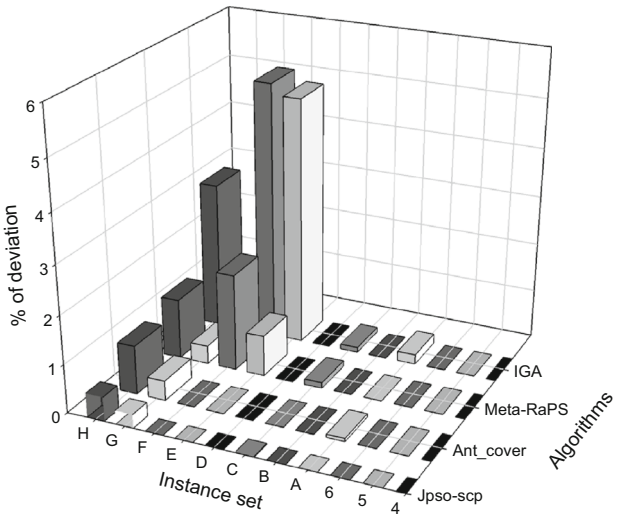

Fig. 2 Mean % of deviation of average values of objective function

an Intel Pentium Core2 Duo Processor PC having 1.6 GHz CPU and 1 GB RAM. All the procedures of the Jpso–scp algorithm have been coded and implemented in MATLAB.

To carry out the effectiveness of the proposed Jpso–scp algorithm, comparison made with the following three best known methods of SCP:

- Ant_cover + *ls*: ACO based approach by Ren et al. (2010)
- Meta-RaPS: Meta-heuristic by Lan et al. (2007)
- IGA: Indirect Genetic Algorithm by Aickelin (2002)

The proposed Jpso–Scp approach is purely based on the spirit of nature, so we wish to test our proposed approach not only with the nature based approaches but also with a well known meta-heuristic approach. In the above mentioned three algorithms first and third methods are nature based technique and the second one is a meta-heuristic approach. In the mean time, it is important to note that all the four algorithms ran on the same platform.

A maximum CPU time allowed is chosen as the stopping criterion for all the four algorithms. This criterion will

Table 3 Pairwise difference between the average ranks of the algorithms

Algorithm (rank)	Jpso–scp (1.29)	Ant_cover + *ls* (2.46)	Meta-RaPS (2.85)	IGA (3.62)
Jpso–scp (1.29)	–	1.17	1.56	2.33
Ant_cover + *ls* (2.46)	–	–	0.39	1.16
Meta-RaPS (2.85)	–	–	–	0.77
IGA (3.62)	–	–	–	–

At 1 % level of significance, for the Nemenyi test, the critical difference value = 1.05

be useful for the direct comparison of all the algorithms with respect to their quality of solution. It is denoted by mCPU in the table.

For each test set, based on the density percentage, 100 SCP test instances are created. All the four algorithms ran on these 100 instances, run for mCPU time and, in each case, the best solution is recorded and the average objective function value is noted.

Results obtained by all the four algorithms are shown in the Table 2, in which first and second column represents instances name and their corresponding best known solution or optimum objective function value, highlighted by bold numbers, correspondingly the instances where the proposed and compared algorithms reaches the optimal solution are highlighted by bold numbers in their columns. In columns three, four, five and six O_{avg} and $T_{avg}(s)$ represents the average objective function value and average time taken in seconds produced by each algorithm to find out the best known solution. More over we identified that the proposed algorithm found optimum solution for 60 instances out of 65 instances in all the 100 runs whereas Ant-Cover + *ls* found optimum for 55 out of 65 instances, Meta-RaPS and IGA found optimum for 53 and 45 instances respectively. In at least one of 100 runs, our proposed Jpso–scp algorithm found optimum solution for all the 65 instances whereas it was 64, 63 and 60 for Ant-Cover + *ls*, Meta-RaPS and IGA respectively.

Further to identify the effectiveness of the proposed algorithm, we have calculated the average of average time taken for each set in the total of 11 set of instances and these results are plotted in the Fig. 1. From this figure we can see that, when compared to other algorithms, the proposed Jpso–scp algorithm took very less average running time for all the 11 set of instances. Further we have calculated the percentage of mean deviation of the average optimum value obtained by all the algorithms from the best known solution or optimum solution using the relation $\{(O_{avg} - \text{Opt}) \times 100\}/\text{Opt}$ and the obtained results are plotted in Fig. 2 and it indicates that the percentage of mean deviation of average objective function value of Jpso–scp algorithm is very small when compared to other three algorithms.

Further in order to confirm the effectiveness of the proposed approach, we follow the rank based statistical analysis; a similar analysis presented in Consoli et al. (2010). For each data set, the ranks of the algorithm are evaluated based on the following performance metric of an algorithm. The performance of an algorithm is considered as better than another one; if in a shortest computational time it obtains either maximum average objective function or an equal average objective function value. The algorithm with best performance assigned with rank 1, rank 2 is assigned to the second best one, and so on. The average ranks of the algorithms among the considered set of instances are IGA = 3.62; Meta-Raps = 2.85; Ant-cover + ls = 2.46 and Jpso–Scp = 1.29. These rank values indicate that the performance of the Jpso–Scp is the best, followed by Ant-cover + ls, Meta-Raps and IGA. This evaluation technique further confirms the superiority of the Jpso–Scp over other compared heuristics.

To analyse the statistical significance of difference between evaluated ranks, we make use of a statistical test for comparison of algorithms. For more detailed study about different tests on comparison of algorithms we can refer (Demśar 2006; Hollander and Wolfe 1973) and the test considered in this work is Nemenyi Post-hoc test (1963). This test is very much useful to identify the performance of two algorithms significantly differs or not. It considers the performance of two algorithms that are significantly different if their corresponding average ranks differ by at least a specific threshold critical difference. Based on this test at 1 % level of significance the critical difference value is 1.05. Table 3 shows that the difference between the average ranks of the algorithms and these results enumerate that the obtained values are greater than the critical difference when other algorithms are compared with Jpso–scp approach. These table values and the smallest rank of Jpso–Scp further insists that the performance of Jpso–scp approach is the best one for solving Set covering problem.

6 Conclusions

In this paper, a new JPSO based approach, Jpso–scp is proposed for solving the set covering problem. This approach finds best solution to SCP in three phases. In the first phase an attractor is selected using row and column cover conditions with low total cost and based on this attractor a feasible solution is then constructed. This

feasiblity is refined for optimality in the second phase and in the third phase redundant columns are removed. i.e. the refined solution in the second phase has been refined further in the third phase. This causes improved quality solution with low computational cost. Extensive computational results show that this approach produces high quality solution in very short running time. Comparison with well known approaches of SCP yield that the proposed Jpso–scp approach based on jumping discrete particle swarm optimization method get tremendous appreciation in solving the SCP by outperforming those approaches. Further, the simplicity, less computational cost and very short running time of the proposed approach imply that the jumping discrete particle swarm optimization based approach is an attractive alternative approach for hard combinatorial optimization problems.

References

Aickelin Uwe (2002) An indirect genetic algorithm for set covering problems. J Oper Res Soc 53(10):1118–1126

Aliguliyev RM (2010) Clustering techniques and Discrete particle Swarm Optimization algorithm for Multi-document summarization. Comput Intell 26:420–448

Al-kazemi B, Mohan CK (2002) Multi-phase discrete particle swarm optimization. In: Fourth international workshop on frontiers in evolutionary algorithms, Kinsale, Ireland

Azimi ZN, Toth P, Galli L (2010) An electromagnetism metaheuristic for the unicost set covering problem. Eur J Oper Res 205:290–300

Balas E, Carrera MC (1996) A dynamic subgradient-based branch-and-bound procedure for set covering. Oper Res 44(6):875–890

Bautista J, Pereira J (2007) A GRASP algorithm to solve the unicost set covering problem. Comput Oper Res 34:3162–3173

Beasley JE (1990) A Lagrangian heuristic for set covering problems. Nav Res Logist 37(1):151–164

Beasley JE, Chu RC (1996) A genetic algorithm for the set covering problem. Eur J Oper Res 94:392–404

Brusco MJ, Jacobs LW, Thompson GM (1999) A morphing procedure to supplement a simulated annealing heuristic for cost and coverage correlated set covering problems. Ann Oper Res 86:611–627

Caprara A, Toth P, Fischetti M (2000) Algorithms for the set covering problem. Ann Oper Res 98:353–371

Ceria S, Nobili P, Sassano A (1998) A Lagrangian-based heuristic for large-scale set covering problems. Math Program 81:215–228

Chen AL, Yang GK, Wu ZM (2006) Hybrid discrete particle swarm optimization algorithm for capacitated vehicle routing problem. J Zhejiang Univ Sci A 7(4):607–614

Consoli S, Pérez JAM, Dowman KD, Mladenović N (2010) Discrete particle swarm optimization for the minimum labelling steiner tree problem. Nat Comput 9:29–46

Cormode G, Karloff H, Wirth A (2010) Set cover algorithms for very large datasets. In: ACM CIKM'10

Correa ES, Freitas AA, Johnson CG (2006) A new discrete particle swarm algorithm applied to attribute selection in a bioinformatic data set. In: Proceedings of the GECCO, pp 35–42

Demśar J (2006) Statistical comparison of classifiers over multiple data sets. J Mach Learn Res 7:1–30

Fisher ML, Kedia P (1990) Optimal solution of set covering/partitioning problems using dual heuristics. Manage Sci 36(6):674–688

Galinier P, Hertz A (2007) Solution techniques for the large set covering problem. Discrete Appl Math 155:312–326

García FJM, Pérez JAM (2008) Jumping frogs optimization: a new swarm method for discrete optimization, Technical Report DEIOC 3/2008, Department of Statistics, O.R and computing, University of La Laguna, Tenerife, Spain

Garey MR, Johnson DS (1979) Computers and intractability: a guide to the theory of NP-completeness. Freeman, San Francisco

Grossman T, Wool A (1997) Computational experience with approximation algorithms for the set covering problem. Eur J Oper Res 101:81–92

Gutiérrez JPC, Silva DL, Pérez JAM (2008) Exploring feasible and infeasible regions in the vehicle routing problem with time windows using a multi-objective particle swarm optimization approach. In: Proceedings of the international workshop on nature inspired cooperatives strategies for optimization, NICSO

Housos E, Elmoth T (1997) Automatic optimization of subproblems in scheduling airlines crews. Interfaces 27(5):68–77

Kennedy J, Eberhart R (1995) Particle swarm optimization. In: Proceedings of the 4th IEEE international conference on neural networks, Perth, Australia, pp 1942–1948

Kennedy J, Eberhart R (1997) A discrete binary version of the particle swarm algorithm. IEEE Conf Syst Man Cybern 5:4104–4108

Lan G, DePuy GW, Whitehouse GE (2007) An effective and simple heuristic for the set covering problem. Eur J Oper Res 176:1387–1403

Li Y, Cai Z (2012) Gravity-based heuristic for set covering problems and its application in fault diagnosis. J Syst Eng Electron 23:391–398

Li J, Kwan RSK (2004) A meta-heuristic with orthogonal experiment for the set covering problem. J Math Model Algorithms 3:263–283

Liaoa C-J, Tseng C-T, Luarn P (2007) A discrete version of particle swarm optimization for flowshop scheduling problems. Comput Oper Res 34:3099–3111

Martinoli PJA (2006) Discrete multi-valued particle swarm optimization. Proc IEEE Swarm Intell Symp 1:103–110

Moirangthem J, Dash SS, Ramas R (2012) Determination of minimum break point set using paricle swarm optimization for system-wide protective relay setting and coordination. Eur Trans Electr Power 22:1126–1135

Ohlsson M, Peterson C, Soderberg B (2001) An efficient mean field approach to the set covering problem. Eur J Oper Res 133:583–595

Pan Q-K, Tasgetiren MF, Liang Y-C (2008) A discrete particle swarm optimization algorithm for the no-wait flowshop scheduling problem. Comput Oper Res 35:2807–2839

Pardalos PM et al (2006) Experimental analysis of approximation algorithms for the vertex cover and set covering problems. Comput Oper Res 33:3520–3534

Qiang L, Na QX, Shi-rang L (2009) A discrete particle swarm optimization algorithm with fully communicated information, ACM GEC, pp 393–400

Raja Balachandar S, Kannan K (2010) A meta-heuristic algorithm for set covering problem based on gravity. WASET 43:504–509

Ren Z-G, Feng Z-R, Ke L-J, Zhang Z-J (2010) New ideas for applying ant colony optimization to the set covering problem. Comput Ind Eng 58:774–784

Saxena A, Goyal V, Lejeune MA (2010) MIP reformulations of the probabilistic set covering problem. Math Program Ser A 121:1–31

Sherbaz AA, Kuseler T, Adams C, Marsalek R, Povalac K (2010) WiMAX parameters adaptation through a baseband processor

using discrete particle swarm method. Int J Microw Wirel Technol 2(2):165–171
Shi XH, Liang YC, Lee HP, Lu C, Wang QX (2007) Particle swarm optimization-based algorithms for TSP and generalized TSP. Inf Process Lett 103:169–176
Solar M, Parada V, Urrutia R (2002) A parallel genetic algorithm to solve the set-covering problem. Comput Oper Res 29:1221–1235
Stützle T, Hoos HH (2000) Max–min ant system. Future Gener Comput Syst 16:889–914
Tasgetiren MF, Suganthan PN, Pan Q-K (2007) A discrete particle swarm optimization algorithm for the generalized traveling salesman problem. In: Proceedings of the GECCO, London, pp 158–165
Telelis OA, Zissimopoulos V (2005) Absolute O(log*m*) error in approximating random set covering: an average case analysis. Inf Process Lett 94:171–177
Toregas C, Swain R, ReVelle C, Bergman L (1971) The location of emergency service facilities. Oper Res Int J 19:1363–1373
Vasko FJ, Wilson GR (1984) Using a facility location algorithm to solve large set covering problems. Oper Res Lett 3(2):85–90
Vasko FJ, Wolf FE (1987) Optimal selection of ingot sizes via set covering. Oper Res 35(3):346–353
Whitehouse GE, DePuy GW, Moraga RJ (2002) Meta-RaPS approach for solving the resource allocation problem. In: Proceedings of the 2002 world automation congress, Orlando, FL
Wolsey LA (1998) Lagrangian duality. In: Wolsey (ed) Integer programming. Wiley, New York, pp 167–181
Hollander M, Wolfe DA (1973) Nonparametric statistical methods, 2nd ed. Wiley, New York
Nemenyi PB (1963) Distribution free multiple comparisons, Ph.D. thesis. Princeton University, New Jersey
Secrest BR (2001) Traveling salesman problem for surveillance mission using Particle Swarm Optimization, Master's Thesis, School of Engineering and Management of the Air Force institute of Technology
Yagiura M, Kishida M, Ibaraki T (2006) A 3-flip neighborhood local search for the set covering problem. Eur J Oper Res 172:472–499
Yang S, Wang M, Jiao L (2004) A quantum particle swarm optimization. In: Proceedings of the CEC2004, the congress on evolutionary computing, vol 1, pp 320–324
Zhan Z-H, Zhang J, Du K, Xiao J (2012) Extended binary particle swarm optimization approach for disjoint set covers problem wireless sensor networks. IEEE Conf Technol Appl Artif Intell 13228691:327–331
Zhang C, Sun J, Wang Y, Yang Q (2007) An improved discrete particle swarm optimization algorithm for TSP, IEEE/WIC/ACM international conferences on web intelligence and intelligent agent technology—workshops, pp 35–38
Zhang H, Sun J, Liu J (2007) A new simplification method for terrain model using discrete particle swarm optimization. In: Proceedings of the 15th international symposium on advances in geographic information systems ACM GIS, pp 1–4