Expert Systems with Applications 39 (2012) 328–334

Contents lists available at ScienceDirect

Expert Systems with Applications

journal homepage: www.elsevier.com/locate/eswa

Pickup and delivery problem using metaheuristics techniques

Craig D'Souza [a], S.N. Omkar [b,*], J. Senthilnath [b]

[a] Department of Computer Engineering, NITK, Surathkal, India
[b] Department of Aerospace Engineering, Indian Institute of Science, Bangalore, India

ARTICLE INFO

Keywords:
Dial-a-ride problem
Metaheuristic
Simulated Annealing
Genetic Algorithm
Particle Swarm Optimization
Artificial Immune System

ABSTRACT

Dial-a-ride problem (DARP) is an optimization problem which deals with the minimization of the cost of the provided service where the customers are provided a door-to-door service based on their requests. This optimization model presented in earlier studies, is considered in this study. Due to the non-linear nature of the objective function the traditional optimization methods are plagued with the problem of converging to a local minima. To overcome this pitfall we use metaheuristics namely Simulated Annealing (SA), Particle Swarm Optimization (PSO), Genetic Algorithm (GA) and Artificial Immune System (AIS). From the results obtained, we conclude that Artificial Immune System method effectively tackles this optimization problem by providing us with optimal solutions.

1. Introduction

A lot of work has been done in the pickup and delivery problem (PDP) area which initially started in the 1980s (Psarafis, 1980). The work done in the pickup and delivery problem area can be found in some of the comprehensive literature surveys done over the years (Mitrovi'c-Mini'c, 1998; Parragh, Doerner, & Hartl, 2008a, Parragh, Doerner, & Hartl, 2008b). One of the main problems we face is scalability. When the number of requests are small it is very easy to analyse the problem. When the number of requests increase, the search space increases. The dial-a-ride problem (DARP) is NP-hard as proven by Baugh, Kakivaya, and Stone (1998). The pickup and delivery problem is considered as an optimization problem where the total cost of the provided service is minimized. A lot of literature is available where optimization models are created to solve this problem. Initially dynamic programming was used to solve the PDP problem but did not yield good results due to the presence of local minima's (Psarafis, 1980). To overcome this drawback people employed metaheuristic approaches to solve this optimization problem (Baugh et al., 1998).

One of the first works published in this area using dynamic programming by Psarafis (1980, 1983), which could only solve problem instances where less than 10 customers are involved. After a few years another method was published by Dumas, Desrosiers, and Soumis (1991) which uses a column generation scheme with a constrained shortest path as a sub problem. The above methods could solve only small problem instances. Therefore researchers adopted metaheuristics to tackle large problems where optimal solutions could be achieved. Nanry and Barnes (2000) and Emmanouil, Christos, and Chris (2009) employed a metaheuristic approach to solve the general pickup and delivery problem (GPDP) using a reactive tabu search method. Similarly Cordeau and Laporte (2003) also used classical tabu search to solve the static multi-vehicle DARP. Potter and Bossomaier (1995) used Genetic Algorithms to solve the vehicle routing problem (VRP). Jih and Hsu (1999) used a hybrid method where they combined dynamic programming and Genetic Algorithms to solve the single-vehicle PDP with time windows.

The various challenges we face with respect to travel in Bangalore are the availability of regular, quick and cost effective transport. The cost effectiveness of a provided service is directly proportional to the distance travelled by the vehicle providing that service. Therefore in the service we provide, we not only reduce the distance travelled by the bus/van but also the distance travelled by each and every passenger. Here the bus/van service provided is of the Dial-a-ride type. The passengers decide where they want to be picked up and where they want to be dropped. This service reduces the distance travelled by every passenger as well as the total distance travelled by the bus/van.

In this paper, we consider the DARP optimization problem presented in earlier studies. Since the problem is NP-hard we use the metaheuristic techniques such as Simulated Annealing (SA), Genetic Algorithms (GA), Particle Swarm Optimization (PSO) and Artificial Immune System (AIS). Here we define $dist_{max}$ which is the maximum distance the bus is allowed to traverse. In Keiichi Uchimura, Takahashi, and Saitoh (2002) a Genetic Algorithm + 2-opt method is proposed. This is equivalent to using a Genetic Algorithm with both crossover and mutation operators and on giving them adaptive probabilities of occurring leads to better solutions. We also show that the AIS is good in giving optimal

* Corresponding author. Tel.: +91 080 229 32873; fax: +91 080 236 00134.
E-mail address: omkar@aero.iisc.ernet.in (S.N. Omkar).

0957-4174/$ - see front matter Crown Copyright © 2011 Published by Elsevier Ltd. All rights reserved.
doi:10.1016/j.eswa.2011.07.022

solutions which are better than the results got using Genetic Algorithm. According to the authors knowledge there is no literature present where AIS (Clonal Selection Algorithm) has been used to solve this problem. On finding optimal solutions for the problem at hand we compare the results and find out which method successfully tackles the problem.

The paper is divided into the following sections. Section 2 contains the problem formulation and problem definition. Section 3 contains the methodologies used. Sections 4 and 5 contains the results and conclusion respectively.

2. Problem formulation

Pickup and delivery problems with timing constraints are referred to as pickup and delivery problems with time windows (PDPTW). Here we find optimal routes for the given transportation requests keeping in mind the capacity, time window and precedence constraints. The vehicles start from a depot. Each request has an origin and a destination. The vehicle which is carrying out a certain request has to first go to the origin and then destination. The various constraints are priority, time and capacity. Since in this problem we only deal with single vehicle pickup and delivery we ignore the capacity and time constraints. The objective here is to not only minimize the total distance travelled by the bus but also minimize the distance travelled by every passenger.

In our objective function we first specify the maximum distance ($dist_{max}$) the bus is allowed to traverse. Any distance above this value is taken as it is by the objective function as S. If it is lesser than the maximum value for any particular route the ratio R is calculated, where R is the maximum ratio of distance between pickup point and drop point for passenger i for that particular route to shortest distance between the two points. Therefore the objective function not only minimizes the total distance travelled by the bus but also the individual distance travelled by each passenger.

Mathematically the objective function f is defined as follows:

$$f = \begin{cases} S, & \text{if } S \geqslant \text{dist}_{max} \\ R, & \text{if } S < \text{dist}_{max} \end{cases} \tag{1}$$

$$R = max(P_iD_i/)\overline{P_lD_l} \tag{2}$$

where,

S = total distance travelled by the bus for a particular route
$dist_{max}$ = maximum distance the bus is allowed to travel
P_iD_i = distance between pickup and drop point for passenger i for a given route
$\overline{P_lD_l}$ = shortest distance between pickup and drop point for passenger i

We shall now develop a graph and address the above problem by defining the constraints in a mathematical framework.

We first assume one vehicle is carrying out all the pickup and delivery requests. Consider n pickup points and n drop points for n passengers. The pickup and drop points can be taken as the vertices of an undirected graph $G(V, E)$. Each edge has a particular weight equivalent to the distance between the two vertices joining the edge. We denote c_{ij} as the distance connecting the two vertices i and j. If i is a pickup point then $n + i$ is the drop point for that particular passenger. Then $P_i = \{1,2,\ldots,n\}$, and $D_i = \{n + 1,n + 2,\ldots,2n\}$. The bus starts from the depot labelled as 0 and ends at $2n + 1$. In total the graph $G\,(V,E)$ has $2n + 2$ vertices and is strongly connected in nature.

To evaluate total distance travelled by the bus we use the Boolean variable x_{ij} which is either true or false depending whether the bus travels through the points i and j. Then S is evaluated as follows:

$$S = \sum_{i=0}^{2n+1} \sum_{j=0}^{2n+1} x_{ij} c_{ij} \tag{3}$$

where,

x_{ij} is the Boolean variable
c_{ij} is the distance between the ith and jth point.

We then check whether priority constraints are satisfied by a particular route. For a particular route we should see that the pick-up point for a passenger is visited before the drop point. Consider a route of $2n + 1$ points. Then the priority constraint can be defined as follows:

$$pos(P_i) < pos(D_i) \text{ for passenger } i1 \leqslant i \leqslant n \tag{4}$$

where,
$pos(x)$ gives the position of the point x in the route.

We define fitness of a particular path as f^{-1}. The various steps in evaluating the fitness of a route can be summarised as follows:

1. Choose dist_{max}.
2. Select path to traverse.
3. Check whether path satisfies priority constraints.
4. Calculate f^{-1} for that path.

Evaluate fitness of the path based on the defined objective function. The aim is to find a path having maximum fitness.

3. Methodology

We use four metaheuristic methods to solve the above problem namely Simulated Annealing, Genetic Algorithms, Particle Swarm Optimization and Artificial Immune System. Out of the four metaheuristic methods three are population based algorithms.

3.1. Simulated Annealing (SA)

Simulated Annealing is a metaheuristic used to find maximum or minimum in a given search space (Baugh et al., 1998; Rutenbar, 1989). The cooling function is dependent on the number of iterations. The cooling starts at an initial random state. The initial temperature is taken as T_1K and the final temperature is T_2K where $T_2 < T_1$. The cooling may terminate for two reasons. The first reason is when convergence is achieved and the second reason is when a fixed number of iterations are completed. The main aim of simulated annealing is to go from an initial state having higher energy to a final state having lower energy. The change of state depends on the probability P and T. The probability P depends on the energy of the current state as well as the energy of the neighbouring state.

The initial state is taken as a random path which satisfies the various constraints. The neighbouring function N contains a swap operator which swaps any two points of the current evaluated path. The probability T determines whether the algorithm is greedy or not. As the temperature decreases the probability of choosing a state with lower energy increases i.e. the algorithm tends to be greedy. If the algorithm behaves in a non greedy manner then depending on the probability P the current state changes to the new state or, the old state is retained. P is dependent on the difference between the energy of the current state and the neighbouring state. If there is a large gap between the two states the chance of changing state decreases where as if there is a small difference between the energy of the two states there is a high probability that the current state will change to the neighbouring state. P can be represented as follows:

$$P = 1 - \left[\frac{|E_{neighbouringstate} - E_{currentstate}|}{E_{currentstate}}\right] \quad (5)$$

The variation of T w.r.t. N is as follows:

$$T = T_1 + \left(\frac{T_1 - T_2}{N_{max}}\right)N \quad (6)$$

where,

T_1 = initial temperature.
T_2 = final temperature.
N = number of iterations.
N_{max} = maximum number of iterations.

The problem of getting stuck to local minima is considerably reduced by allowing not only downhill movement but also uphill movement. This is possible because of the probability P.

3.1.1. SA algorithm

1. Initialize the current state i.e. the state is assigned to a random set of pickup and drop points which satisfy the various constraints.
2. Evaluate the energy of the current state i.e. the fitness of the path.
3. By passing the current state to the neighbouring function N we get the neighbouring state whose energy (fitness) is evaluated.
4. Depending on T (i.e. if it is greedy) the current state is replaced with neighbouring state if the energy of the neighbouring state is lesser than the current state else the current state is retained.
5. The current state (i.e. if it is not greedy) shifts to the new state depending on the probability P.
6. The steps 2 to 4 are repeated till the termination condition is achieved.

3.2. Genetic Algorithm (GA)

Genetic Algorithm is a population based metaheuristic (Goldberg, 1989). An initial population P is taken i.e. a random set of routes are assigned to the initial population. The fitness of the population is then evaluated according to the fitness function. A random selection of chromosomes from the current population is made. The selected chromosomes are then subjected to genetic operators where new paths are generated. The new population is then evaluated. This process goes on for N generations so as to achieve convergence. Three genetic operators are used namely 2-point crossover, mutation and reproduction. Each operator has a predefined probability to occur. The operators are also adaptive in nature i.e. their probability of occurrence changes for the next generation.

3.2.1. 2-Point crossover

Consider the two parents P_1 and P_2 undergoing the crossover. Let the length of the chromosomes be n. Two index variables i and j are taken such that $i < j \leqslant n$. The pickup or drop points between i and j are retained in the children C_1 and C_2 respectively. We then traverse from j to $j - 1$ (looping around once we reach the end) in parent P_2 placing points in C_1 from j to i without repeating pickup or drop points in C_1. The similar process is carried out to find C_2 by traversing from j to $j - 1$ in parent P_1. After C_1 and C_2 are found the four chromosomes are evaluated and the best child is selected to be added to the current population while the best parent is retained in the current population. An example of the 2-point crossover is shown below.

P_1 =	2	3	1	5	4
P_2 =	5	3	2	4	1
$i = 2, j = 3$	$n = 5$				
C_1 =	2	3	1	4	5
C_2 =	1	3	2	5	4

After the 2-point crossover the children's fitness values are evaluated and the best child along with the best parent is added to the current generation.

3.2.2. Mutation

Consider a parent P being subjected to the mutation operation. Let the length of the chromosome be n. Two index variables $i$and j are randomly given values such that $i, j \leqslant n$. The values in the ith and jth positions in the selected parent are swapped. The mutated chromosome M is then added to the current generation in the place of the selected parent. An example of mutation is shown below.

P =	2	3	1	5	4
$i = 2, j = 3$	$n = 5$				
M =	2	1	3	5	4

3.2.3. Reproduction

In the reproduction operation the selected parent P is retained in the next generation. Each of the above operations have a fixed initial probability to occur which all add up to 1. Throughout all the generations the sum of probabilities of the different operations remains the same i.e. equal to 1. The probabilities of crossover and mutation are adaptive in nature. As each generation passes the probability of a crossover decreases and the probability of a mutation occurring increases. The probability of reproduction remains the same throughout all generations. The crossover gives the algorithm the ability to find large variations in fitness value whereas mutation allows small variations in a specific path accommodating small changes in fitness value. Crossover reduces the possibility of getting caught up in local minima.

3.2.4. GA algorithm

1. The initial population is initialized with random paths consisting of pickup and drop points and the fitness of each parent is evaluated according to the objective function.
2. A random selection of two parents is made and they are subjected to genetic operations according to the probabilities. This is done for the full generation without selecting a parent twice.
3. The new generation thus formed is then evaluated and the global minimum is recorded.
4. The steps 2 and 3 are repeated for N generations or until convergence is observed.

3.3. Particle Swarm Optimization (PSO)

PSO is a population based metaheuristic method (Kennedy & Eberhart, 1995). Each particle is assigned an initial position and velocity in the given search space. The particles then traverse through the search space depending on its fitness. The movement of the particles are governed by the position of their local bests and the global best in the given search space. The velocity and position after each iteration varies according to the following equations.

$$V_{i+1} = \omega V_i + c_1 r_1 (globalbest - X_i) + c_2 r_2 (localbest_i - X_i) \quad (7)$$

$$X_{i+1} = X_i + V_i \quad (8)$$

where,

c_1, c_2 are constants.
r_1, r_2 are random. vectors.
$\omega$is the inertial constant.
X_i is the position of the ith particle.
V_i is the velocity of the ith particle.
globalbest is the point in the search space where maximum fitness value has been evaluated.
localbest$_i$ is point in the search space where maximum fitness value for the ith particle has been evaluated.

3.3.1. PSO algorithm

1. The n particles are given an initial velocity and position. Each particle is given a string of pickup and drop points.
2. The fitness of each particle is evaluated according to the objective function only if the constraints for the given route are satisfied.
3. The global best and local best for each particle is found and the new velocity and position is calculated.
4. New random routes are assigned to the particles.
5. The steps 2 to 4 are repeated until all the particles converge to a single point.

3.4. Artificial Immune System (AIS)

AIS like GA is a population based metaheuristic method (Farmer, Packard, & Perelson, 1986). The Clonal Selection Algorithm is used for optimization purposes. The Clonal Selection Algorithm uses properties of antibodies and antigens. As time passes the affinity of the antigen towards a particular antibody increases. Initially, equal number of antibodies and antigens are selected and given initial values. The antibody and antigen chains can be binary coded or real coded. Since the problem we are facing is a discrete optimization problem we use real coded chains like the ones used in the previous algorithms. Here each chain is initially given a chain of random pickup and drop points which satisfy the priority and time constraints. The chains are evaluated and the fitness of each route is evaluated. Encoding and decoding functions are used to encode real values to binary values and decode binary to real values. Here since the chains are real coded no encoding or decoding functions are present. Out of the initial population P, n chains with high fitness value are selected and cloned. The cloned chains then undergo the process of hypermutation and reselection.

3.4.1. Hypermutation

This operation is similar to the mutation operation in GA. Suppose n chains are selected for cloning and n_c clones are created per chain then $n \times n_c$ chains are subjected to the hypermutation operation according to the probability pm. The probability pm is adaptive in nature and varies for a certain number of iterations after being set to its original value. This occurs throughout the total number of iterations. The indices$i$ and j are randomly selected and a swap or flip operation is done.

An example of the hypermutation operation is shown below:

P =	2	3	1	5	4
i = 2, j = 4;					
HM =	2	5	1	3	4

3.4.2. Re-selection

In the re-selection process the n best clones are selected from $n \times n_c$ clones and placed in the current population. The replacement can be either greedy or non-greedy. If we follow a greedy method we replace the worst n in the current population P with the n best clones. If we use a non-greedy strategy then we replace the worst in particular intervals. The changed population forms the population for the next iteration.

3.4.3. Clonal Selection Algorithm

1. Initialize antigen and antibody chains i.e. each chain is associated to a string of pickup and drop points which satisfy the various constraints.
2. Evaluate fitness of the current population and select$n$ out of the evaluated chains.
3. Generate n_c clones for each chain.
4. Subject the clones to hypermutation and select n best clones.
5. The n best clones replace the bad chains in the current population based on fitness.
6. Repeat steps 2 to 4 for N iterations.

After N iterations the chain with the best fitness is taken as the solution.

4. Results

The above methods are tested on a real world scenario. We chose 20 locations in Bangalore City (Fig. 1). These locations are chosen from all over the city. We then find out the point-to-point distance between these locations using a Bangalore map and generate an adjacency matrix. This adjacency matrix is used to calculate the objective function. For a testing scenario we chose pickup and drop points from these available twenty locations. We use the same configuration for all the methods so that they can be compared effectively.

The example used is as follows (refer Fig. 1): initially start location is selected to be 1 and stop location as 20. Then pick and drop point is selected i.e,

$$(\text{pick}, \text{drop}) = (2,3); (4,5); (6,7); (8,9); (10,11); (12,13); (14,15); (16,17); (18,19);$$

We then set the maximum distance the bus is allowed to travel to be 150 *Kms*. We then execute each algorithm to note the variation of R and S as shown in Table 2. The results for the four methods mentioned above are compared to see which method gives the optimal value.

4.1. Simulated Annealing

We first start with a random route as the initial state and calculate its energy/fitness. Each route is real coded. The most favourable parameter values are as follows:

1. Initial Temperature = 100K.
2. Final Temperature = 0.5 K.
3. Number of iterations N = 5000.

For every iteration the current state/route gets modified when it is passed to the neighbour function N and the temperature constantly decreases according to the Eq. (6). Therefore the greedy nature of the algorithm increases as it progresses. The fitness value i.e. R and S is evaluated. Since we follow an elitist approach the state having the best value of R and S is recorded throughout the process.

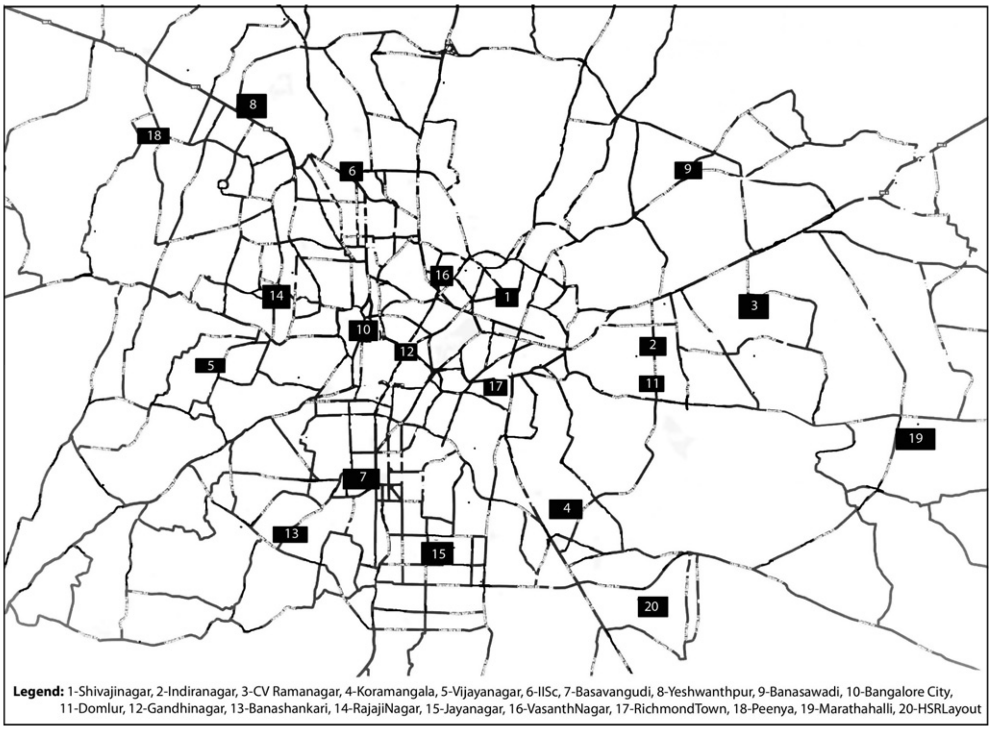
8
18
6
9
16
1
14
3
10
12
2
5
11
17
19
7
4
13
15
20
Legend: 1-Shivajinagar, 2-Indiranagar, 3-CV Ramanagar, 4-Koramangala, 5-Vijayanagar, 6-IISc, 7-Basavangudi, 8-Yeshwanthpur, 9-Banasawadi, 10-Bangalore City, 11-Domlur, 12-Gandhinagar, 13-Banashankari, 14-RajajiNagar, 15-Jayanagar, 16-VasanthNagar, 17-RichmondTown, 18-Peenya, 19-Marathahalli, 20-HSRLayout

Fig. 1. Bangalore city marked with 20 points.

Table 1
Probabilities for various genetic operators.

Operator	Initial probability (first generation)	Final probability (last generation)	Change
2-point crossover	0.7	0.05	0.65 (adaptive-decreasing)
Mutation	0.1	0.75	0.65 (adaptive-increasing)
Reproduction	0.2	0.2	0.0 (no change)
Total	1.0	1.0	

4.2. Genetic Algorithm

Like the previous algorithm we initialize the population with random routes which are real coded. The most favourable parameter values are as follows:

1. Population = 25.
2. Selection is random.
3. Genetic operators: initial rate.
    Crossover rate: 70%.
    Mutation rate: 10%.
    Reproduction rate: 20%.
4. Number of generations $N = 5000$.

Table 1 shows us the adaptive nature of the 2-point crossover and mutation operators. For every generation the fitness of the current population is calculated. Like the previous algorithm we follow an elitist approach in finding the optimal solution. The parent of the current population having the best fitness value is recorded. Therefore after N iterations the global best will have the fitness value of the best parent it has come across. The population is then subjected to genetic operations. The selection of the parents which undergo the various operations is random. The parent which has the highest fitness value in the current generation is retained in the next generation.

4.3. Particle Swarm Optimization

All the particles are initially associated with a random route which is real coded. The experimental conditions are as follows:

1. Number of particles $n = 12$ or 25.
2. Number of iterations $it = 5000$.
3. $c_1 = 1$.
4. $c_2 = 1$.
5. $\omega = 1$.

Initially all the particles are randomly placed in the specified space and given random velocities. Each particle is associated with a string of pickup and drop points. The fitness of the particles i.e. R and S is evaluated and the particle having the best fitness is recorded. The change in velocity and position of the particles is then calculated. The particles are then assigned random new routes and their fitness is evaluated which continues for N iterations. Like the previous methods in this method we use an elitist approach and global best records the best fitness value encountered.

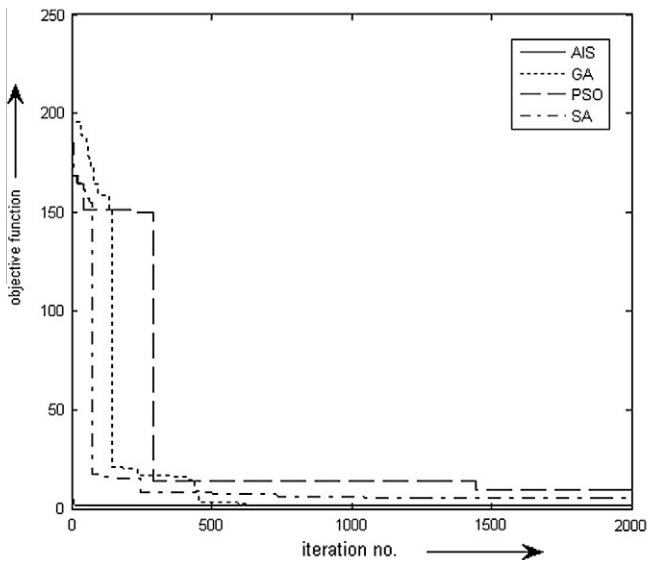

Fig. 2. Convergence graph obtained for AIS, GA, PSO and SA algorithms.

4.4. Artificial Immune System

The Clonal Selection Algorithm is also population based. The antibodies and antigens are initialised using random real coded routes. The most favourable parameter values are as follows:

1. Number of antibodies = 75.
2. Number of antigens = 75.
3. Number of generations *it* = 150.
4. Number of clones per candidate = 6.
5. Hypermutation (flip operation) probability = 0.6.

After initializing the population the fitness of all the chains are calculated. Out of the current population *n* chains having a high fitness value are selected and cloned. The cloned chains are then subjected to the hypermutation operation. Out of all the clones, *n* clones having high fitness values are selected and they replace the chains having lower fitness value in the current population. Thus in every iteration the chain having good values of *R* are retained. Therefore in the convergence graph we see that value of *R* continuously decreases until it converges. The value the graph converges to gives us the optimal value of *R* (Fig. 2).

4.5. Comparison of Metaheuristic Techniques

Since the problem is NP-hard using metaheuristics are an effective way to achieve optimal or near-optimal solutions for real-world problems (Glover & Kochenberger, 2003). In-recent years many metaheuristic methods have been used to solve transportation related problems (Li & Lim, 2003; Sakakibara, Tamaki, & Nishikawa, 2007).

From Table 2, we can observe in comparison with other metaheuristic methods the results obtained using SA is not optimal, as the algorithm is not population based and tends to get caught up in local minima. The neighbour function being only a swap does not have the capability to remove the algorithm out of the local minima although the non greedy nature helps it to some extent. The results got using PSO are not optimal on comparison with GA and AIS. Being a discrete optimization problem every route is represented as a real coded string which is not mapped to a point in space. Therefore the selection of new routes after fitness evaluation is not affected by the current position of the particle. Therefore results got using PSO are not optimal. GA provides us with optimal results as the search space is effectively scanned for optimal results which can be seen in Table 3. The crossover operator helps the algorithm from getting out of local minima and the mutation operator provides us with small changes in fitness value. The adaptive nature of the algorithm also helps in convergence. Similarly from Table 3 we can observe that AIS searches effectively with the help of the hypermutation operation.

The convergence plot (Fig. 2) shows us the rate of convergence of the different algorithms. We see that AIS converges very quickly compared to the other algorithms. On examining the plots of *R* and *S* (Figs. 3, 4) for the various algorithms we can come to the following conclusions. By observing the SA plot for each iteration we see there is an irregular movement. This can be explained by the fact that the algorithm has the capability to behave in a greedy and non-greedy fashion. After *S* becomes lesser than $dist_{max}$, *R* decreases or increases depending on the probability *P*. On examining the plot of *R* and *S* values for GA we see that once *S* falls below $dist_{max}$, *R* continuously decreases while *S* fluctuates below $dist_{max}$, which can be attributed to the carryover of the fittest parent to the next generation. The above nature is also observed in the AIS plot along with a fast convergence. On observing the plot of *S*

Table 2
Comparison of results obtained by PSO, AIS, GA and SA algorithms.

Sl. No.	Particle Swarm Optimization (PSO)		Artificial Immune System (AIS)		Genetic Algorithm (GA)		Simulated Annealing (SA)	
	Value of *S* (Km)	Minimum *R*	Value of *S* (Km)	Minimum *R*	Value of *S* (Km)	Minimum *R*	Value of *S* (Km)	Minimum *R*
1	148.7	6.9998	132.0	1.9593	139.7	3.3721	134.7	4.7143
2	147.3	14.1143	129.2	3.1293	145.5	2.4717	132.7	3.2907
3	142.4	12.2567	139.6	2.3810	139.3	2.3711	131.8	3.3810
4	149.6	8.8086	133.6	2.6122	148.2	1.7317	112.0	3.8027
5	149.6	9.6621	136.0	2.4626	148.3	2.9456	134.8	3.5820

Table 3
Variation of S and R values during the running of PSO, AIS, GA and SA algorithms.

	Particle Swarm Optimization (PSO)		Artificial Immune System (AIS)		Genetic Algorithm (GA)		Simulated Annealing (SA)	
cline8-9	Value of S (Km)	Value of R	Value of S (Km)	Value of R	Value of S (Km)	Value of R	Value of S (Km)	Value of R
Maximum	182.3000	30.2025	163.4000	15.6087	238.8000	29.8571	208.6000	24.0204
Minimum	148.4000	7.8050	127.6000	4.5889	130.2000	2.6279	116.1000	7.1006
Average	150.3183	12.8599	149.4088	4.6230	149.0927	6.3269	127.6782	12.8176
Standard Deviation	0.2742	0.0802	0.2733	0.0481	0.2753	0.0567	0.2527	0.0801

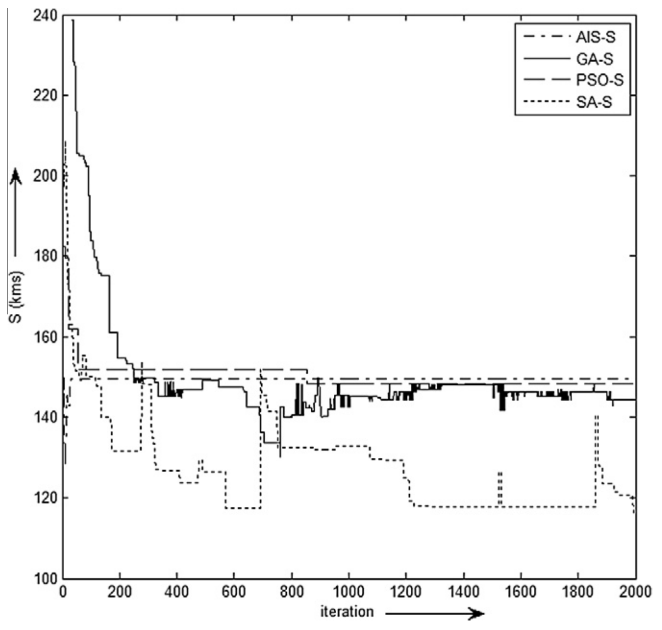

Fig. 3. Variation of *S* in AIS, GA, PSO and SA algorithms.

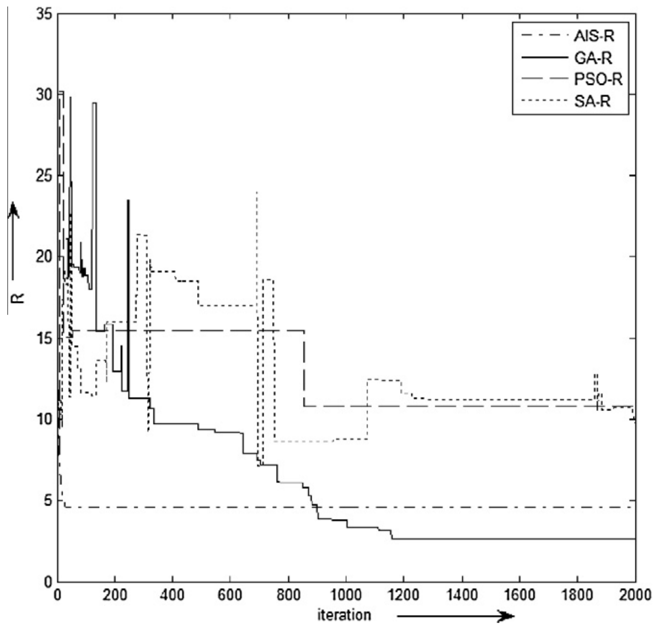

Fig. 4. Variation of *R* in AIS, GA, PSO and SA algorithms.

and *R* values for PSO we see that *S* and *R* values decrease slowly due to the inability to scan the search space effectively when compared with the other methods (Table 3). From Table 2, we see that GA and AIS gives us optimal solutions. The best average solution is provided by AIS.

5. Conclusions

In this paper we use metaheuristics to solve the dial-a-ride problem specific to Bangalore city. We use four different methods to solve the problem so as to decide which method tackles the problem effectively. The four different methods used were Simulated Annealing, Particle Swarm Optimization, Genetic Algorithm and Artificial Immune System. We choose different points in Bangalore city to implement the above four methods. From the results we see that PSO is ineffective at dealing with the problem and SA tends to get into local minima's. GA and AIS tackle the problem effectively giving us optimal solutions. By using these metaheuristic methods we can optimise transport leading to better utilization of time and resources.

References

Baugh, J. W., Jr., Kakivaya, D. K. R., & Stone, J. R. (1998). Intractability of the dial-a-ride problem and a multiobjective solution using simulated annealing. *Engineering Optimization, 30*(2), 91–124.

Cordeau, J. F., & Laporte, G. (2003). A tabu search heuristic for the static multi-vehicle dial-a-ride problem. *Transportation Research Part B* (37), 579–594.

Dumas, Y., Desrosiers, J., & Soumis, F. (1991). The pick-up and delivery problem with time windows. *European Journal of Operational Research* (54), 7–22.

Emmanouil, E. Z., Christos, D. T., & Chris, T. K. (2009). A hybrid metaheuristic algorithm for the vehicle routing problem with simultaneous delivery and pick-up service. *Expert Systems with Applications, 36*(2), 1070–1081.

Farmer, J. D., Packard, N., & Perelson, A. (1986). The immune system, adaptation and machine learning. *Physica D, 22*, 187–204.

Glover, F., & Kochenberger, G. A. (2003). *Handbook of metaheuristics*. Kluwer Academic Publisher.

Goldberg, D. E. (1989). *Genetic algorithms in search. Optimization and machine learning*. Addison–Wesley.

Jih, W.-R., & Hsu, Y.-J. (1999). Dynamic vehicle routing using hybrid genetic algorithms. In *IEEE Computer Society (ed) Proceedings of the 1999 IEEE International Conference on Robotics & Automation* (pp. 453–458).

Kennedy, J., Eberhart, R. (1995). Particle swarm optimization. In *Proceedings of IEEE international conference on neural networks. IV* (pp. 1942–1948).

Li, H., & Lim, A. (2003). A metaheuristic for the pickup and delivery problem with time windows. *International Journal on Artificial Intelligence Tools, 12*(2), 173–186.

Mitrovi'c-Mini'c, S. (1998). Pickup and delivery problem with time windows: A survey. Technical Report. SFU CMPT TR 1998-12. School of Computing Science, Simon Fraser University, Burnaby, BC, Canada.

Nanry, W. P., & Barnes, J. W. (2000). Solving the general pickup and delivery problem with time windows using reactive tabu search. *Transportation Research Part B: Methodological* (34), 107–121.

Parragh, S., Doerner, K., & Hartl, R. (2008a). A survey on pickup and delivery problems: Part I: Transportation between customers and depot. *Journal fürBetriebswirtschaft, 58*(2), 21–51.

Parragh, S., Doerner, K., & Hartl, R. (2008b). A survey on pickup and delivery problems: Part II: Transportation between pickup and delivery locations. *Journal fürBetriebswirtschaft, 58*(2), 81–117.

Potter, T., & Bossomaier, T. (1995). Solving vehicle routing problems with genetic algorithms. In *IEEE Computer Society (ed) Proceedings of the 1995 IEEE international conference on evolutionary computing* (pp. 788–793).

Psarafis, H. (1980). A dynamic programming solution to the single vehicle, many-to-many, immediate request dial-a-ride problem. *Transportation Science* (14), 130–154.

Psarafis, H. (1983). An exact algorithm for the single-vehicle many-to-many dial-a-ride problem with time windows. *Transportation Science* (17), 351–357.

Rutenbar, R. (1989). Annealing algorithms: An overview. *IEEE Circuits and Devices Magazine, 5*(1), 19–26.

Sakakibara, K., Tamaki, H., & Nishikawa, I. (2007). Autonomous distributed approaches for pickup and delivery problems with time windows. In *SICE Annual Conference 2007*, September 17–20, Kagawa University, Japan.

Uchimura, Keiichi, Takahashi, Hiro, & Saitoh, Takashi (2002). Demand responsive services in hierarchical public transportation system. *IEEE Transactions on Vehicular Technology, 51*(4).