SOLVING COMBINATORIAL OPTIMIZATION PROBLEM USING CHEAPEST SHOP SEEKER ALGORITHM

Supported by tetfund TERTIARY EDUCATION TRUST FUND

Peter Bamidele Shola[1] and Asaju La'aro Bolaji[2]
[1]Department of Computer, University of Ilorin, PMB 1515, Kwara State, Nigeria
[2]Department of Computer Science, Federal University Wukari, PMB 1020, Taraba State, Nigeria
*Corresponding author: shola.bp@unilorin.edu.ng

Received: February 11, 2017 **Accepted:** March 24, 2017

Abstract: In this paper, a discrete version of the cheapest shop seekers algorithm is presented for solving the traveling salesman problem. The cheapest shop seeker, a recently proposed nature-inspired algorithm utilized to solve the global optimization function. It is a population-based metaheuristic inspired by mimicking a group of shoppers cooperatively seeking for the cheapest shop for shopping and proved to be effective when investigated in continuous domain. The performance of the discrete CSS algorithm is evaluated on some benchmark instances from TSPLIB. Experimental results show that the discrete version is found to be effective on small instances where it obtained optimum solution. Similarly, it had comparable performance on the large instance.

Keywords: Cheapest shop seeker, COPs, metaheuristics, population-based algorithm, TSP

Introduction

The cheapest shop seeker algorithm is a newly introduced population-based meta-heuristic algorithm meant for solving optimization problem in a continuous domain (Shola, 2016). It is a stochastic optimizer which is inspired by a group of shoppers cooperatively seeking for the cheapest shop for shopping. The success recorded by the CSS when applied to some benchmark functions in continuous domain motivated the idea of investigating its performance in discrete domain, more specifically to combinatorial optimization problems. A combinatorial optimization problem (COP) is a problem of finding a feasible solution that globally optimizes a given objective function in a discrete finite search space which expands exponentially as the dimension (or size) of the problem increases. COP is an important area of optimization to which many resource management problems belong. Such problem arises in finance, marketing, production, scheduling, inventory control, production, facility location, and in many engineering problems where an optimum design of a structure or product obtainable from the limited available resources, is targeted.

Many methods for obtaining the exact solution to COPs have been proposed. One such method is the branch and bound where feasible solutions are organized in a tree-like structure and a branch is made at each node to explore a path only as far as the bound on the optimum solution (obtained elsewhere) is not exceeded. A backtrack is made to explore another path whenever a bound is exceeded on a path. Few of the studies that employed usage of brand and bound can be found in (Ignall and Scharge 1965; Potts and VanWassenhove 1985; Gendreau *et al.*, 1998; Ronconi, 2005). The cutting plane methods in integer programming have also been used to find the exact solutions to some COPs (Gomory, 1958, 1960). The method iteratively performs the following:

(a) Relaxes the linear programming problem (LPP) by dropping the constraint that the variables be integer type and solve the resulting the LPP, to obtain an optimum solution say $\boldsymbol{x}^*$.

(b) Returns $\boldsymbol{x}^*$ as solution if it is an integer solution otherwise find a cutting (hyper) plane that cuts off a part of the search space in a way that the remaining part contains all the integer feasible solutions of the original search space but not $\boldsymbol{x}^*$. Solve the LPP in this remaining search space for the new optimum solution $\boldsymbol{x}^*$. Repeat step (b).

Other exact methods which have been employed for the COPs include dynamic programming in which the problem is redefined in terms of a set of problems of the same type as the original problem but with each having a smaller search space than the original space (Held and Karp 1962), constraint satisfaction techniques where the COP is reformulated as a constraint satisfaction problem (Brailsford *et al.*, 1999) and some other integer programming techniques like relaxation techniques and decomposition techniques (Benders, 1962). COPs have also been formulated as graph problems in which some graph search algorithms (such as A* search) applied to solve them (Kaya and Uçar, 2009). The exact methods are, however, not suitable for solving a large-sized COPs due to the exponential explosion of the population of their feasible solutions. The approximate solutions (or just good solutions) are all that can be obtained for the COPs and many approximate methods have been devised and applied for such purpose. The two classical examples of approximate methods are local search-based and population-based techniques.

The local search-based optimizer is a single solution local improvement heuristic (technique) that iteratively moves from a initial feasible solution to a neighbouring one based on a given neighbourhood function $N: S -> 2^S$, that defines the set of feasible solutions $N(x) \subset S$ which are neighbours of each feasible solution $\underline{x}$ in the search space S. The effectiveness of a local search depends on the nature of the neighbourhood function (i.e. the size of the neighbourhood it generates and the coverage of the feasible solutions in the search space), the speed of the method employed for searching a neighbourhood (especially if large in size) and the rule for identifying a neighbour to replace the current solution. For instance, the larger the neighbourhood the better may be the solution produces but with more time taken to search the neighbourhood to identify a neighbour to select. The choice of a neighbour also has impact on the computing time: a local search-based method such as hill climbing (that selects just a neighbour better than the current solution) may take less time than the steepest descent (that selects the best neighbour as all the neighbours of the current solution may need be examined to determine the best especially in a discrete space where gradient computation is not valid). Few studies that have applied local search-based to the COPs can be found in (Thompson, 1988; Crauwels, 1998). The problem with the local search-based methods is its inability to escape from local optimum which makes it often returns a local optimum solution. To address this point, the basic local search-based has been improved upon and along many directions. A variable neighbourhood search is a kind of improved local search that employs a set of neighbourhood structures and switches from one to another as the search progresses. Many applications of VNS to numerous COPs can be found in (Divsalar, 2013; Shaw, 1998). The large neighbourhood works by alternatively destroying and rebuilding a solution using an extensive set of destroy and repair heuristics. In adaptive large neighbourhood, the frequency usage of the different destroys and repair heuristics are made a function of their previous performance (Ropke and Pisinger, 2006). The local search-based metaheuristic on the other hand employs a mechanism to reduce its likelihood of getting stuck in a local optimum. Different methods of this kind that have been used to solve many COPs are: (i) simulated annealing which adopts some sort of random move to avoid being trapped in a local optimum (Kirkpatrick, 1983; Matsuo, 1989), (ii) tabu search which keeps memory of past movement and uses this information to identify and escape from local optimum (Glover and Laguna, 1997; Gendreau *et al.*, 1994; Gendreau *et al*., 2006). The population-based technique starts with provisional solutions called population. At each iteration, these solutions are perturbed using the basic components to generate new ones with anticipation that quality will be higher than that of the candidate solutions. Some population-based employed to tackle the COPs are genetic algorithm that uses evolutionary principle of the survival of the fittest as strategy and some form of randomness (Manar and Shameen, 2011; Takeshi and Ryohei, 1995). The issue with these methods is the computing time required to produce the good solutions. Similarly, few examples of prominent nature-inspired population-based metaheuristics originally devised for continuous optimization problems that have adapted to tackle different COPs are particle swarm optimization (PSO) developed by Kennedy and Eberhart (1995) and utilized for the COPs like task assignment (Salman, 2003), classification (Sousa, 2004), orienteering problem (Shanthi and Sarah, 2011), and flowshop scheduling problems (Liao, 2007) and binary discrete version of the PSO is proposed by Kennedy and Eberhart (1997). On the other hand, ant colony optimization (ACO) is originally developed to tackle discrete optimization problems by (Dorigo and Gambardella, 1997). It has been used to tackle many COPs (Besten *et al.*, 2000; Blum and Sampels, 2004). A brief review of nature inspired algorithm can be found in (Fister *et al.*, 2013). In another development, other studies that employed the usage hybrid systems to solve many COPs especially timetabling problem could be found in (Bolaji *et al*., 2014). In hybrid system, two or more optimization techniques are combined to solve an optimization problem. For instance, mat-heuristics combines a metaheuristics and integer programming techniques (Pirkwieser *et al*., 2008, Mezmaz *et al*., 2007) while memetic technique involves the combination of two or more metaheuristics such as local search-based and population-based approaches together (Bouly *et al*., 2008; Burke *et al*., 1996; Sonawane and Leena, 2014; Lin *et al*., 2009). Literatures have also reported some studies that combined metaheuristics with some machine learning techniques such as neural network, fuzzy logic to tackle some optimization problems (Kwon and Moon, 2003). Note that the focus of this paper is to investigate the performance of the proposed Discrete CSS on traveling salesman problem which is one of the classical examples of the combinatorial optimization problems

The paper is organized in the following way. Section 1 presents the introduction while section 2 discusses the classical CSS algorithm. Section 3 presents the framework of the discrete CSS and experimental results is given in section 4. Finally, Section 5 provides conclusion and future research directions.

The cheapest shop seeker Algorithm

The cheapest shop a seeker (CSS) is a population-based stochastic algorithm simulating a group of shoppers cooperatively searches for the cheapest shop to purchase their goods. During the search process, the CSS engages a collection of agents which explore the search space in a cooperative manner in order to obtained solutions to a given optimization problem. The success of the CSS algorithm depends on the capability of the agent in the group to memorize the past experiences (i.e. memorized the best position). Each member of the population cooperatively shared experience in order to achieved common objectives. Each group member competes to survive in the population by searching for the position which could improve the global best position. Similarly, each member of the group has the ability to explore the search independently to improve its own current position.

In CSS algorithm, the solution space is initialized with shops available for shopping where each shop represents a candidate solution to the optimization problem. Furthermore, there is a specified number of shoppers cooperatively searching for the cheapest shop among the shops. Then shoppers interact with each other by sharing their experience where the information received from others and personal experience could be utilized to determine the next shop to visit. A shopper close or near the current cheapest shop could sometimes disregard its personal experience or available information in order to explore search space for the new positions with aims of obtaining a better position than the current global position. Algorithm 1 shows the pseudocoode of the cheapest shop seeker as proposed for solving continuous optimization problems (Shola, 2016).

Algorithm 1: The pseudocode for the cheapest shop seeker

Parameters:

P: number of shops in the populations

N: number of iterations

$c_0,\ c_1$: positive constants usually in the range [2,4].

Dim: the dimension of the problem.

rand(): generates a random number between [0,1]

$\underline{x}_i^k$: vector denoting the position of particle*i* at time *k* (i.e. at k[th] iteration)

$\underline{GB}^k$: vector denoting the globally best position (of all the particles) attained up to time k.

$\underline{LB}_i^k$:vector denoting the best position up to time k attained by particle i

$distance(\underline{u},\ \underline{v})$:the geometric distance of position $\underline{v}$ from $\underline{u}$

$\underline{\text{minx}}$ =(minx$_1$, minx$_1$, ….., minx$_{\text{dim}}$) and

$\underline{\text{maxx}}$ =(maxx$_1$, maxx$_1$, ….., maxx$_{\text{dim}}$)

where minx$_j$ andmaxx$_j$ (j=1,2..,dim) are respectively the lower and upper bounds for the value of component j of $\underline{x}_i^k$

fitValue($\underline{z}$): the fitness value of position $\underline{z}$.

ϵ :the bound on the distance of the shop from the current global position below which particles generate their position randomly.

Initialization step:

(a) INITIALIZE randomly the positions $\underline{x_i}^{(0)}$ of all the shops in the population:

$$\text{for } i = 1, 2, \cdots, N \quad \text{set}$$

$$\underline{x}_i^{(0)} = \underline{\min x} + rand\,() * (\underline{\max x} - \underline{\min x})$$

(b) $\text{for } i = 1, 2, \cdots, N$

COMPUTE the fitness value

$f_i = \text{fitValue}\ (\underline{x}_i^0)$, of shop's position $\underline{x}_i^0$

(c) Set the global best position $\underline{GB}^0$ to the shop position with the best fitness value

Iterative step:

for $k = 1, 2, \cdots, N$ do the following looping

for $i = 1, 2, \cdots, P$ do the following

{ (i)UPDATE $\underline{x}_i^k$ to obtain $\underline{x}_i^{k+1}$,

(a) $\underline{v} = \underline{x}_i^k + \text{rand}\ () * c_0 * (\underline{GB}^k - \underline{x}_i^k)$ 1

(b) $\underline{u} = D * \underline{x}_i^k + c_1 * (\underline{LB}^k - \underline{x}_i^k)$ 2

With any component of $\underline{u}$ or $\underline{v}$ out of interval bound generated randomly as in (a) of initialization step

(c) $if\ (\text{fitValue}(v) > \text{fitValue}(u))\ \ then$

set $\underline{x}_i^{k+1} = \underline{v}$

else set $\underline{x}_i^{k+1} = \underline{u}$

(d) $if\ (\text{distance}(\underline{x}_i^{k+1},\ \underline{GB}^k) < \varepsilon)\ \ then$

$$\underline{x}_i^{(k+1)} = \underline{\min x} + rand\ () * (\underline{\max x} - \underline{\min x})$$

Update step:

Update global best position $\underline{GB}$ to obtain $\underline{GB}^{k+1}$, and the fitness value of $\underline{GB}^k$,

$$if\ (\text{fitValue}(\underline{GB}^k) < \text{fitValue}(\underline{x}_i^{k+1}))\ \ then$$

set $\underline{GB}_i^{k+1} = x_i^{k+1}$

else

set $\underline{GB}_i^{k+1} = \underline{GB}_i^k$

Output the current global best position, $\underline{GB}^N$, and its fitness value, $\text{fitValue}(\underline{GB}^N)$.

The proposed discrete CSS algorithm

In this section, a novel discrete cheapest shop seeker algorithm (DCSS) is proposed for the COPs especially the traveling salesman problem. The continuous nature of the original CSS is adapted in order to handle the discrete search space of the TSP. The definition of the adaptation when utilized to tackle the TSP is provided as follows:

$$\underline{p} \oplus \underline{q} = \underline{r} = \begin{cases} p_j\ if\ rand() > c \\ q_j\ otherwise \end{cases}$$

with the result repaired where necessary

$$\underline{p} \oplus \underline{q} = \text{bestOf}\ (p, q)$$

$$= \text{better of}\ \underline{p}, \underline{q}\ \text{in terms of fitness values}$$

$$c\,\underline{p} = \underline{r} = \begin{cases} \text{mutation of } \underline{p} \text{ obtained by swapping two} \\ \text{entries } p \text{ chosen randomly} \quad if\ \text{rand}() > c \ \text{for} \\ \underline{p} \qquad \text{otherwise} \end{cases}$$

any two vectors $\underline{p} = (p_1, p_2, \cdots, p_N)$, $\underline{q} = (q_1, q_2, \cdots, q_N)$ the shop position updating equations (1) and (2) is converted to

$$\underline{u}_i^{k+1} = D * \underline{x}_i^k \oplus c_1' (\underline{LB}_i^k\ \Theta\ \underline{x}_i^k) \quad (3)$$

$$\underline{v}_i^{k+1} = D * \underline{x}_i^k \oplus c_2' (\underline{GB}_i^k\ \Theta\ \underline{x}_i^k) \quad (4)$$

$$\underline{x}^{k+1} = \text{bestOf}\ (\cdots, \cdots, \underline{u}^{k+1}, \underline{v}^{k+1}) \quad (5)$$

Where $\underline{u}^k, \underline{v}^k, \underline{w}^k$ are repaired when necessary before accepting the best and in a case where $\underline{x}^{k+1}$ equals $\underline{GB}^k$, a feasible solution is randomly generated for $\underline{x}^{k+1}$s.

The procedure of DCSS algorithm is given as

Initialization step:

(a) INITIALIZE randomly the positions $\underline{x}^{(0)}$ of all the shops in the population:

for $i = 1, 2, \cdots, N$ set

$$\underline{x}_i^{(0)} = \underline{\min x} + rand\ () * (\underline{\max x} - \underline{\min x})$$

(b) for $i = 1, 2, \cdots, N$

COMPUTE the fitness value

$f_i = \text{fitValue}\ (\underline{x}_i^0)$, of shop's position $\underline{x}_i^0$

(c) Set the global best position $\underline{GB}^0$ to the shop position with the best fitness value

Iterative step:

for $k = 1, 2, \cdots, N$ do the following looping

for $i = 1, 2, \cdots, P$ do the following

{ (i)UPDATE $\underline{x}_i^k$ to obtain $\underline{x}_i^{k+1}$,

(a) $\underline{v} = c_2' [\underline{GB}_i^k\ \Theta\ \underline{x}_i^k](\underline{x}_i^k)$

(b) $\underline{u} = c_1' [\underline{LB}_i^k\ \Theta\ \underline{x}_i^k](\underline{x}_i^k)$

(c) $\underline{x}_i^{k..+1} = \text{bestOf}\ (\underline{x}^k, u, \underline{v})$

in terms of fitness value

(d) $if\ (\text{equal}\ (\underline{x}_i^{k..+1},\ \ GB^k))\ \ then$

$\underline{x_i}^{(k+1)}$ =generate randomly and**repair** where necessary.

(ii) UPDATE: global best position $\underline{GB}$ to obtain $\underline{GB}^{k+1}$, and the fitness value of $\underline{GB}^k$,

$$if\ (\text{fitValue}(\underline{GB}^k) < \text{fitValue}(\underline{x}_i^{k+1}))\ \ then$$

set $\underline{GB}_i^{k+1} = x_i^{k+1}$

else

set $\underline{GB}_i^{k+1} = \underline{GB}_i^k$

}

Output the current global best position, $\underline{GB}^N$, and its fitness value, $\text{fitValue}(\underline{GB}^N)$.

Experimental Results & Discussions

Here the algorithm is tested on traveling salesman problem (TSP). The choice of travelling salesman problem, as a test case, is made due to the fact that the problem is one of the most significant problems in combinatorial optimization (Schrijver, 2005) and many permutation problems such as scheduling and routing might be case as a traveling salesman problem. In fact TSP has become a standard test case for discrete optimization method. The travelling salesman problem is the problem of finding the shortest route (path) a salesman would take to visit a given number of cities with no city visited more than one time. The discrete versions of the algorithm presented below were tested on data instances from the TSPLIB and some of the results presented below in the table.

Table 1:Experimental results of the proposed method on TSP instances from TSPLIB

Instance	Best Known	Proposed DCSS	Nearest neighbour
GR17	**2085**	**2085**	2187
FRI26	**937**	**937**	1112
bays29	**2020**	**2020**	2267
gr120	**6942**	9028	9351

As shown in Table 1, it can be deduced that the algorithm could able to obtain the exact solution in the problem instances of small dimensions (see instances GR17, FRI26 and bays29 in Table 1). Similarly, the proposed algorithm obtained comparable results in the large instance of the TSP dataset. However, when compared with the solution obtained by the nearest neighbour method as one of the initial solutions for gr120 in the algorithm improved its result from 9028.000000 to 8271 towards the optimum solution 6942.

Conclusion

In this work the cheapest shop optimization technique presented in [1] for continuous domain is adapted for discrete optimization problem. The performance of algorithm is tested on the traveling salesman problem benchmark instances found in TSPLIB. The result of the experiment shows that the proposed discrete cheapest seeker shopping obtained optimal solution in four instances of the dataset and had a comparable performance in the remaining instance. The algorithm seems to be a promising one for discrete optimization from the result. The performance of the proposed method could further investigated on other instances of the TSPLIB in order to justify its performance. Therefore, our future work will focus on improving the technique by integrating with components of other metaheuristic algorithms.

References

Benders JF 1962. Partitioning procedures for solving mixed-variables programming problems. *Numerische Mathematik*, *4*(1): 238-252.

Blum C & Sampels M 2004. An ant colony optimization algorithm for shop scheduling problems.*J. Mathematical Modelling & Algorithms*, 3(3): 285-308.

Bolaji AL, Khader AT, Al-Betar MA & Awadallah MA 2014. University course timetabling using hybridized artificial bee colony with hill climbing optimizer.*J. Computational Sci.*, 5(5): 809-818.

Bouly H, Dang DC & Moukrim A 2008. A memetic algorithm for the team orienteering problem.*Applications of Evolutionary Computing*, 649-658. *Lecture Notes in Computer Science*, Vol. 4974: 649-658.

Brailsford SC, Potts CN & Smith BM 1999. Constraint satisfaction problems: Algorithms and applications. *Eur. J. Operational Res.*, 119(3): 557-581.

Burke E, Newall J & Weare R 1996. A memetic algorithm for university exam timetabling. In:E. Burke, P. Ross(eds) The Practice and Theory of Automated Timetabling,*Lecture Notes in Computer Science*, Vol. 1153: 241-250, Springer Verlag.

Crauwels HAJ, Potts CN & Van Wassenhove LN 1998. Local search heuristics for the single machine total weighted tardiness scheduling problem. *INFORMS Journal on Computing*, 10(3): 341-350.

Den Besten M, Stützle T & Dorigo M 2000. Ant colony optimization for the total weighted tardiness problem. In *Parallel Problem Solving from Nature PPSN VI* (pp. 611-620). Springer Berlin/Heidelberg.

Divsalar A, Vansteenwegen P & Cattrysse D 2013. A variable neighborhood search method for orienteering problems with hotel selection. *Int. J. Production Econ.*, 145(1): 150 – 160.

Dorigo M 1997. Ant colony system: A cooperative learning approach to the travelling salesman problem. *IEEE Transactions on Evolutionary Computation. IEEE*, 1(1): 53-66.

Fister Jr. I, Yang XS, Fister I, Brest J & Fister D 2013. A brief review of nature-inspired algorithms for optimization ELEKTROTEHNISKI VESTNIK 80(3) English edition.

Gendreau M, Hertz A & Laporte G 1994. A tabu search heuristic for the vehicle routing problem.*Management Science*, 40(10): 1276-90.

Gendreau M, Laporte G & Semet F 1998.A branch-and-cut algorithm for the undirected selective traveling salesman problem.*Networks*, 32(4): 263-273.

Gendreau M, Lori M, Laporte G & Martello S 2006. A tabu search algorithm for a routing and container loading problem.*Transportation Science*, 40(3): 342-50.

Glover F & Laguna M 1997. *Tabu Search*, Kluwer Academic Publishers, Boston.

Gomory RE 1958. Outline of an algorithm for integer solution to linear programs. *Bull. Amer. Math. Soc.*, 64: 275–278.

Gomory RE 1960. Solving linear programming problems in integers.*Combinatorial Analysis*, 10: 211-215.

Held M & Karp RM 1962.A dynamic programming approach to sequencing problems.*J. Soc. Indus. & Appl. Maths.*, 10(1): 196-210.

Ignall E & Schrage L 1965. Application of the branch and bound technique to some flow- shop scheduling problems. *Operations Res.*, 13(3): 400-412.

Kaya K & Uçar B 2009. Exact algorithms for a task assignment problem. *Parallel Processing Letters*, 19(03): 451-465.

Kennedy J & Eberhart J 1995 *Particle Swarm Optimization*. In: Proc. IEEE International Conference Neural Networks, Piscataway NJ, Vol. 4: 1942-1948.

Kennedy J & Eberhart R 1997. A discete binary version of the particle swarm optimization [A]. In *Proceeding of the Conference on System, Man, and Cybernetics* (Vol. 100). NJ, USA: IEEE Service Center.

Kirkpatrick S, Gelatt CD & Vecchi MP 1983. Optimization by simulated annealing.*Sciences*, 220(4598): 671-680.

Kwon YK & Moon BR 2003. Daily stock prediction using neuro-genetic hybrids. In *Genetic and Evolutionary Computation–GECCO* 2003 pp. 214-214. Springer Berlin/Heidelberg.

Liao CJ, Tseng CT & Luarn P 2007. A discrete version of particle swarm optimization for flowshop scheduling problems. *Computers & Operations Res.*, 34(10): 3099-3111.

Lin SW, Lee ZJ, Ying KC & Lee CY 2009.Applying hybrid meta-heuristics for capacitated vehicle routing problem. *Expert Systems with Applications*, 36(2), 1505-1512.

Manar H & Shameen F 2011."*Survey of genetic algorithms for university timetabling problem*" International conference on future information technology IPCSIT vol 13 IACSIT Press Singapore.

Matsuo H, Juck SC & Sullivan RS 1989. A controlled search simulated annealing method for the single machine weighted tardiness problem. *Annals of Operations Res.*, 21(1): 85-108.

Mezmaz M, Melab N & Talbi EG 2007. Combining metaheuristics and exact methods for solving exactly multi-objective problems on the grid. *J. Math. Modelling & Algorithms in Operations Res.*, 6(3): 393-409.

Muthuswamy S & Lam S 2011. Discrete particle swarm optimization for the orienteering problem.*Int. J. Indus. Engr: Theory, Applic. & Practice*, 18(2): 92-102.

Pirkwieser S, Raidl GR & Puchinger J 2008. A Lagrangian decomposition/ evolutionary algorithm hybrid for the knapsack constrained maximum spanning tree problem. In: *Recent Advances in Evolutionary Computation for Combinatorial Optimization* pp. 69-85. Springer Berlin Heidelberg.

Potts CN & Van Wassenhove LN 1985. A branch and bound algorithm for the total weighted tardiness problem. *Operations Res.*, 33(2): 363-377.

Reinelt G 1991. TSPLIB – A traveling salesman problem library.*ORSA J. Computing*, 3(4): 376-384.

Ronconi DP 2005.A branch-and-bound algorithm to minimize the makespan in a flowshop with blocking. *Annals of Operations Res.*, 138(1): 53-65.

Ropke S & Pisinger D 2006. An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows. *Transportation Science*, 40(4): 455-472.

Salman A, Ahmad I & Al-Madani S 2002. Particle swarm optimization for task assignment problem.*Microprocessors and Microsystems*, 26(8): 363-371.

Shaw P 1998. Using constraint programming and local search methods to solve vehicle routing problems.In International Conference on Principles and Practice of Constraint Programming (pp. 417-431). Springer Berlin Heidelberg

Shola PB, 2016. The Cheapest Shop Seeker: a new algorithm for optimization problem in a continuous space. *IAES Inter. J. of Art. Intel. (IJ-AI)*, 4(3), 119 - 126.

Sonawane MPA & Ragha L 2014. Hybrid genetic algorithm and TABU search algorithm to solve class time table scheduling Problem. *Int.. J. Res. Studies in Comp. Sci. & Eng.*, 1(4): 19-26.

Sousa T, Silva A &NevesA 2004. Particle swarm based data mining algorithms for classification tasks. *Parallel Computing*, 30(5): 767-783.

Takeshi Y & Ryohei N 1995. *A Genetic Algorithm with Multi-step crossover for Job-shop Scheduling problems*. First IEE/IEEE International conference on Genetic Algorithms in Engineering systems: Innovations and Applications (GALESIA '95), 12-14 September 1995, Sheffield, UK (pp. 146-151).

Thompson PM 1988. *Local search algorithms for vehicle routing and other combinatorial problems*. PhD thesis, Operations Research Center, MIT.