A Modified ACO with K-OPT for Restricted Covering Salesman Problems in Different Environments

Prasanta Dutta (dutta1900@yahoo.com)
Debra Thana S.K.S Mahavidyalaya
Indadul Khan
Chandrakona Vidyasagar MV https://orcid.org/0000-0002-4052-551X
Krishnendu Basuli
West Bengal State University
Manas Kumar Maiti
Mahishadal Raj College

Research Article

Keywords: Covering Salesmen Problem, Ant Colony Optimization, K-opt operation, Fuzzy Simulation, Rough Simulation

Posted Date: January 10th, 2022

DOI: https://doi.org/10.21203/rs.3.rs-1060695/v1

Version of Record: A version of this preprint was published at Soft Computing on April 8th, 2022. See the published version at https://doi.org/10.1007/s00500-022-06978-0.

A MODIFIED ACO WITH K-OPT FOR RESTRICTED COVERING SALESMAN PROBLEMS IN DIFFERENT ENVIRONMENTS

Abstract. In this study, the ant colony optimization (ACO) algorithm is modified with the K-opt operation to solve the covering salesman problem(CSP) under one restriction in crisp and imprecise (fuzzy, rough) environments. A CSP involves two phases- the division of cities into groups with the selection of the visiting cities and searching of the Hamiltonian circuit through the visiting cities. But, none of the studies in the literature is made following the direct approach. Also, none of the studies in the literature gives attention to reduce the total travel distance of the unvisited cities from the visited city of a group. Moreover, there is no algorithm in the literature which provides the solution of a CSP with the specified coverage range r. Also, none has introduced any algorithm to solve CSPs in imprecise environments. Though algorithms are available to solve the Traveling Salesman Problems in the imprecise environments, the approach cannot deal with the problems involving fuzzy data with non-linear membership functions or the problems involving rough data where the rough estimation can not be done using Lebesgue measure. The well establish algorithm for any routing problem is the ACO, but not much attention has been paid to solve the CSP using ACOs. To overcome these limitations on the studies of the ACO on the CSPs, here, an algorithm is proposed for the division of groups of the set of cities depending upon the maximum number of cities in a group and the total number of groups. Then ACO is used to find the shortest/minimum-cost path of the problem by selecting only one visiting the city from each group without violating the restriction of the specified coverage range r of the location of the unvisited cities. K-opt operation is applied periodically at the end of ACO operation to improve the quality of the best found solution so far by the ACO algorithm and to arrest any premature convergence. For the restricted problems paths are searched in such a manner that the total distance/travel cost of different unvisited cities of a group from the visited city of the group should not exceed a predefined upper limit. To solve the problem in an imprecise environment some approach is followed so that the tour is searched without transferring the imprecise optimisation problem into an equivalent crisp optimisation problem. Also, the simulation approaches in fuzzy and rough environments are proposed to deal with the CSPs with any type of estimation of the imprecise data set. Algorithm is tested with the standard benchmark crisp problems available in the literature. To test the algorithm in the imprecise environments, the imprecise instances are derived randomly from the standard crisp instances using a specified rule. Test results imply that the proposed algorithm is efficient enough in solving the CSPs in the crisp as well as in the imprecise environments.

Keywords: Covering Salesmen Problem; Ant Colony Optimization; K-opt operation; Fuzzy Simulation; Rough Simulation.

1. Introduction

A covering salesman problem (CSP) is a complete weighted graph, consists of a set of vertices, called cities, and a weight matrix, called distance (cost) matrix which consists of the distance (travelling cost) between any two cities. The goal of the problem is to find the different clusters (groups) of the cities and then find a minimum cost Hamiltonian circuit visiting only one city in each cluster so that each unvisited city in a cluster should be located in a predefined coverage range of the visiting city. A maximum number of cities in a group is provided in the problem. The clustering and the searching of Hamiltonian circuit is made in such a manner that the travelling distance (cost) through the circuit is a minimum. The CSP has various real life applications, like, telecommunication [14], health care [9], disaster management[21], humanitarian relief[2], transportation[28], etc. For the relief of effected areas of different natural disasters, like,

flood, earthquake, tsunami, etc, the locations of the relief camps are selected depending upon the human resources in such a manner that all the effected peoples can get different services with the minimum effort and expenditure. Mobile communication tower points are selected in such a way that covers all the customers' area with the minimum expenditure. So, the CSP has a wide range of applications in the real life problems.

As CSP belongs to the class of NP-hard problems, exact (analytical approach) algorithms are not suitable to find the solution of a CSP in a usable time window. So heuristic approaches are applicable to find a compromise solution in a specific time window. The initial study in this direction was made by current and schilling[1] in 1989. They proposed the problem and the problem was solved by the authors using a two phase heuristics approach. In the first phase of the algorithm, the set covering approach is followed to find the clusters of the vertex set. Then one node from each cluster is selected as the facility node. Then a traveling salesman problem (TSP) is formed through the selected nodes. In the second phase, this TSP is solved using any suitable heuristics approach. These two phases are repeated iteratively to search the best path.

Using Lin-Kernighan procedure [16], Golden et al. [7] proposed two heuristics search algorithms, LS_1 and LS_2 for solving different CSPs in 2012. Both the algorithms start with a randomly generated set of feasible solutions of the problem. Then LS_1 uses some stochastic approach to replace some nodes from the path by some new nodes to improve the quality of the solution in such a manner that the feasibility of the solution persists. The authors used mutation operations to prevent any local optima and used some uphill moves to introduce diversity in the algorithm. On the other hand LS_2 uses two iterative procedures, namely, the Improvement Procedure and the Perturbation Procedure to find the best feasible solution.

Using the features of integer linear programming (ILP) Salari and Naji-Azimi[24] proposed a Heuristics method for solving CSP in 2012. The algorithm starts with a randomly generated set of feasible solutions of the problem and the tour is improved iteratively using two procedures, namely, Heuristic based improvement and ILP based improvement. An Extraction-Reassignment procedure is used to decrease the tour length in the heuristic phase. At first, a subset of the vertices visited by the tour is extracted and then this vertices are reassigned into a new tour by solving the ILP model for the improvement of the objective function. Another heuristic algorithm by Lin-Kernighan[16] is also used for the possible improvement of the tour. Also, a perturbation phase is introduced in the algorithm to escape from the local optima.

Similar to the CSP, another problem, named, Covering Tour Problem (CTP), was proposed by Gendreau et al.[6] in 1992. In this problem the set of vertices, V, is partitioned into to two subsets V_1 and V_2 and the goal is to search a minimum cost Hamiltonian circuit through V_1 so that the distance of each of the vertices of V_2 from the circuit is located within a prescribed distance. They first formulate the problem as an ILP problem and then an exact branch and cut algorithm is developed to solve the problem. They have also proposed a heuristics approach for it. In a recent study, Singh et al. [22] proposed a mixed integer programming model for a bi-objective generalised covering salesman problem, where simultaneously, the total covered demand is maximised and the total tour length is minimised.

Combining the features of ant colony optimization (ACO) algorithm and the dynamic programming technique, Salari et al[25] proposed a heuristic approach to solve CSPs. Together with 3-opt, they introduced two perturbation approaches vertex-removal and vertex-addition for searching quality solutions. Venkatesh and Singh [29] modified the Artificial Bee Colony (ABC) algorithm with different perturbation techniques to solve the CSPs in 2019. Venkatesh

et al.[29] proposed a local search heuristic for the CSPs using multiple perturbation strategy. In another study, Pandiri et al.[22] modified two heuristic approaches ABC and Genetic Alorithm (GA) to solve CSPs in 2020. They proposed four approaches for selecting neighbour solutions in both the algorithms- GA and ABC, namely, Subset Neighbour, Local search with two neighbour structures, General Exchange and Permutation neighbour. Zang et al.[31] defined Bilevel CSP(BCSP) and proposed two algorithms based on, parallel variable neighbourhood search (PVNS), namely, synchronous master slave PVNS and asynchronous cooperative PVNS, for the same. A crossover procedure using GPX was proposed by Tripathy et al. [27] and using it a GA was developed by them for the CSPs in 2017.

Though a significant amount of studies is made on the CSPs after its introduction by current and schilling[1] there are some lacunas in the existing literature, which are summarised below:

- The well-established algorithm for any routing problem is the ACO, but not much attention has been paid to solve the CSPs using ACO [25].
- It is known that, a CSP involves two phases- division of cities into groups with the selection of visiting cities and searching of the minimum cost Hamiltonian circuit through the visiting cities. But, none of the studies in the literature is made following this direct approach.
- Moreover, none of the studies gives attention to reduce the total travel distance of the unvisited cities from the visited city of a group. Though in real life problem it is crucial as different facilities are provided from the visited cities only.
- Also, there is no algorithm in the literature, which gives results of the CSP instances with the specified coverage range r. All the studies have been made where groups are created with the nearest nodes from the selected centres of the groups.
- All the studies upto till date proposed different heuristics for CSPs for the betterment of the existing results, which in turn motivates us for the further betterment.
- In all the above studies, it is observed that the problems are considered in the crisp environment only, i.e., the cost of travel between any two specific cities is fixed, which is unrealistic in any real life situation. In fact, the cost of travel between any two places varies with several factors, like, road conditions, type of vehicle used, the route used, etc. This cost is actually imprecise in nature and can be estimated as rough or fuzzy number using experts' opinion. This type of estimation is less error prone as it is done using experts' opinion. Though there are algorithms to solve basic TSPs and its variants in the imprecise environments[11, 12, 13], none has introduced any algorithm to solve the CSPs in any imprecise environment.
- Though algorithms are available to solve TSPs in the imprecise environments [12, 13], the approach cannot deal with the problems involving fuzzy data with non-linear membership functions (e.g., Parabolic Fuzzy Number(PFN)) or TSPs involving Rough data where the estimation can not be done using Lebesgue measure.

To overcome the above mentioned lacunas, in this study an attempt has been made to introduce a better heuristic approach to solve the CSPs in the crisp and as well as in the imprecise environments. Simulation approaches in the imprecise environments are proposed to deal with the CSPs in the imprecise environments for any type of imprecise data sets. Moreover, a restricted problem is introduced and solved where the total travel distance of different unvisited cities from the visited city of the corresponding group should not exceed a prescribed limit.

The heuristic algorithm proposed here consists of three phases. In the first phase of the algorithm, the set of cities is partitioned into groups depending upon the maximum number of cities in a group and the total number of groups. A procedure is proposed for this purpose where initially a centre of each group is selected and then other cities of the group are selected in such a manner that all other cities are located within the specified coverage range r. The centre of the next group is selected from the unselected cities which is nearest to this center. In the second phase of the algorithm, the ACO [4] is used to find the shortest/minimum-cost path of the problem by visiting only the centre of each group. In the last phase, the K-opt algorithm is used to improve the quality of the solution obtained by the ACO and to resist premature convergence. For the restricted problem, the path is searched in such a manner that the total distance/cost of different unvisited cities of a group from the visited city of the corresponding group should not exceed a predefined upper limit. To solve the problem in an imprecise environment an approach is followed, where, the tour is searched without transferring the imprecise optimisation problem into any equivalent crisp optimisation problem. Moreover, fuzzy simulation and rough simulation approaches are proposed to deal with the problem with any general type imprecise data set.

The rest of the paper is organized as follows: in section 2, the background and the main challenges of solving the CSPs in different environments are discussed. The required assumptions and different notations for the mathematical representation of the problem and the development of the proposed algorithm are listed in section 3. The mathematical representation of the problem and some technical prerequisites for the proposed solution approach are presented in section 4. Optimization in imprecise environments and the simulation approaches are discussed in section 5. The ACO algorithm for the CSP is described in section 6. In section 7, the K-Opt algorithm is presented. Proposed approach is presented in section 8. Experimental results are discussed in section 9. A brief conclusion is drawn in section 10. At length, the references are listed.

2. Background of the CSP and challenges

CSP was defined by Current and Schilling[1] in 1989. They formulated the problem as a zero-one linear programming problem (LPP) and proposed a heuristic approach to solve the same. In their approach, in the first step, the subset of cities to cover all the cities are determined using the approach of solving a set covering problem (SCP). For each of the solution of SCP, a TSP is generated and solved. The minimum cost tour is the solution of the CSP. As SCP and TSP, both are NP-hard problems, so, it is not possible to solve a moderate size CSP in a reasonable time window following this approach.

Combining the features of dynamic programming technique, ACO and amalgamating different perturbation techniques, like, 3-opt, vertex removal and vertex addition techniques, etc., Salari et al.[25] developed a novel heuristic to solve CSPs in 2015. In this study paths are created by selecting successive cities using ACO until all the cities are covered. Then, the vertex removal and vertex addition techniques are used to improve the tour cost. Finally, 3-opt operation is applied on the path for further possible improvements. This study ignores the restriction of the upper limit, r, of the coverage range of the unvisited cities and consider only the nearer nodes of any visiting city as the covered nodes for searching the path.

In a recent study, Pandiri et al. [22] proposed two meta-heuristics for the CSPs. In the first approach, ABC algorithm is modified for the CSPs using four perturbation operations on a tour. One of these approach is the removal of a visited vertex and addition of an unvisited

vertex which can cover the removed vertex and they named it LS2N. Another approach is the permutation of neighbour, where, some permutation is made on the positions of the visited cities in a tour to improve the quality. The third approach is the general exchange (GE) of an unvisited city with a visited city, such that the feasibility of the tour holds. The fourth approach is named subset neighbour (SSBN) and in this approach some cities of a tour are removed with some probability and some unvisited cities are added so the tour becomes feasible as well as the quality is improved. Using these perturbation approaches they, developed a tour improvement algorithm and is used in the employed bee phase and in the onlooker bee phase of the basic ABC to make it useful for the CSPs. For the generation of initial solution set for the ABC as well as for the GA approach they followed the same approach as Salari et al.[25]. In GA cyclic crossover operation is used and for the mutation process they proposed two new approaches. They have also used the perturbation approaches of the ABC algorithm in the GA for the further possible improvements. Performances of both these approaches are similar to the hybrid ACO proposed by Salari et al.[25]. So the approaches of Pandiri et al. [22] did not give better solution than that of Salari et al.[25]. Moreover, this study also ignores the restriction of the upper limit, r, of the coverage range of the unvisited cities and consider only the nearer nodes of any visiting city as the covered nodes for searching the path.

All the above studies as well the studies mentioned in the introduction section, the initial paths are created using successive addition of visiting cities to cover all the cities following the approaches of set covering problem, but none has mentioned the covering range in their test instances, i.e., the maximum distance of an unvisited city from its nearest visiting city, though the tour cost mostly depends on it and it is also the basic restriction of the problem. Also, none of these studies tries to reduce the total group covering distance of a visiting city, though it is realistic one for any real life problem. Since, the CSP is a routing problem, selection of the visiting cities and the determination of the optimal rout through these cities using any suitable routing algorithm, may produce better result (as proposed by Current and Schilling[1]). But the approach is overlooked by the researchers for developing any heuristic approach. Moreover, none have studied the CSPs in imprecise environments. There are some studies on the TSPs and GTSPs with fuzzy cost matrices and rough cost matrices [11, 12, 13], using the credibility measure on fuzzy events and the trust measure on rough events, where TFNs are used as the fuzzy parameters and the Lebesgue measure is used for the estimation of rough parameters. Their approach can not deal with such problems involving fuzzy cost matrices with non-linear membership functions. In fact, when more than one fuzzy parameters with non-linear membership function (e.g., PFN) are added then it is not possible to determine the analytical form of the membership function of the resultant fuzzy number. So following their approach it is not possible to determine the credibility measure of the fuzzy events involving the process when the membership functions of the fuzzy parameters are non-linear type. The same problem has to be face in the rough environment also, if the Lebesgue measure is not possible to use for the rough estimation.

To overcome these limitations in this study fuzzy simulation approach and rough simulation approaches are proposed to deal with such crucial situations. Moreover, the CSP is solved with an approach where initially the cities are divided into groups selecting one possible visiting city of each group. Then the routing heuristic ACO is used to determine the optimal route through the visiting cities. K-opt operation is also used periodically for the possible improvement of the best path found so far and to resist any premature convergence. In a particular case, the CSP is solved with the restriction limit of the total group coverage range as well as incorporating the normal restriction on the coverage range, r.

3. Assumptions and Notations

Following assumptions and notations are used to formulate the problem:

(i) n is the number of cities involved in the problem.
(ii) NC is the maximum number of unvisited cities covered by a visiting city.
(iii) NB is the maximum number of cluster of cities.
(iv) c_{ij} is the travelling distance/cost between city i and city j.
(v) r is the maximum distance between a visiting city and the furthest unvisited city covered by that visiting city.
(vi) $(x_{ij})_{n\times n}$ is the decision variable matrix for the mathematical representation of the problem

$$\text{where } x_{ij} = \begin{cases} 1 & \text{if the salesman moves to city } j \text{ from city } i \\ 0 & \text{otherwise} \end{cases}$$

(vii) Another array variable $(v_i)_{1\times n}$, is also used for the mathematical representation of the problem, where,

$$v_i = \begin{cases} 1 & \text{if the salesman visits city } i \\ 0 & \text{otherwise} \end{cases}$$

(viii) Also the coverage matrix $(y_{ij})_{n\times n}$ is defined as

$$y_{ij} = \begin{cases} 1 & \text{if } c_{ij} \le r \\ 0 & \text{otherwise} \end{cases}$$

(ix) Z is the total tour cost/travel distance.
(x) The ascent ˜ is used over a symbol to indicate fuzzy quantity.
(xi) The ascent ˇ is used over a symbol to indicate rough quantity.

4. Technical Background

Covering Salesman Problem: The basic CSP consists of a set of cities $V = \{1, 2, ...n\}$ where the travelling distance/ cost between any two cities i and j is known and fixed, say c_{ij}. The goal of the problem is to search a minimum distance Hamiltonian circuit through a subset of cities so that the position of every unvisited city is located within a predefined coverage range r of a visited city. Then the problem mathematically takes the following form

$$\text{Minimize} \qquad Z = \sum_{i\in V}\sum_{j\in V} c_{ij}x_{ij} \tag{1}$$

$$\text{Subject to} \qquad \sum_{j\in V} x_{ij} + \sum_{j\in V} x_{ji} = 2v_i \ \ \forall i \in V \tag{2}$$

$$\sum_{j\in V} y_{ji}v_j \ge 1 \ \ \forall i \in V \tag{3}$$

$$\sum_{i\in S}\sum_{j\in V\setminus S} x_{ij} \quad + \quad \sum_{i\in V\setminus S}\sum_{j\in S} x_{ij} \ge 2(v_l + v_k - 1),\ S \subset V, 2 \le |S| \le n-2, \forall l \in V,\ k \in V \setminus S \tag{4}$$

In the formulation, the objective function Z is the total travel distance of the tour, which is to be minimized. Constraints set (5) implies that if the salesman visits a city he must left the city after his visit. Constraints set (6) implies that every vertex in the vertex set V is located in a covering distance r from a visited city. Finally, equation set (7) implies that the tour should not contain any sub-tour.

In the existing literature, every problem consists of a predefined number of clusters (NB) and upper limit (NC) of the number of cities in a cluster. A salesman should visit only one city of each cluster so that the problem constraints are satisfied.

To solve the problem here following steps are followed:

- At first randomly select a city c_1 to be visited, and search $NC-1$ cities nearest to the selected city. If these cities are located inside the covering distance r, then this set of vertices is named as the first cluster. If it is not true then another vertex is selected and the same process is continued until a cluster is created.
- After the formation of a cluster, the nodes in the clusters are eliminated from the node set and let it be V_1.
- Select next city c_2 to be visited from V_1 which is nearest to c_1 and form a cluster following the same procedure as first cluster. If fails then the next nearest node of V_1 from c_1 is selected as c_2 and the process is continued.
- Following the same procedure, different clusters are created. In this formation, in any step if it fails to create a cluster then the process again starts from the formation of first cluster.

A salesman at first partitioned the cities into some clusters according to the constraints. Then the salesman visits one of the cities of each group which is located within the coverage range r of all the unvisited cities for the selling/canvassing/distribution of some products to its customers of the different cities of the groups. The goal of the problem is threefold:

- First partition the cities into clusters depending upon the maximum number of cities in a cluster and clusters limit.
- Search a minimum cost path for the traveller through all the clusters by visiting only one city of each cluster.
- Every unvisited city must lies within a predefined covering distance from a visited city.

Fuzzy Number: A fuzzy set $\tilde{F}$ is defined on an universe U such that every $u \in U$ belongs to the set $\tilde{F}$ with some degree of membership $\mu_F(u) \in [0,1]$. The function $\mu_F : U \to [0,1] \subset \mathbb{R}$ is called the membership function of the fuzzy set $\tilde{F}$. In other words the function μ_F completely defines the fuzzy set $\tilde{F}$ on U. A fuzzy set normally denoted by using an ascent˜over the name. A fuzzy number $\tilde{A}$ is a fuzzy set on $\mathbb{R}$ such that its membership function μ_A satisfies the following two conditions [30]:

- $\exists\, a \in \mathbb{R}$ such that $\mu_A(a) = 1$.
- $\forall\, a,\, b \in \mathbb{R}$ and $\lambda \in (0,1),\ \mu_A(\lambda a + (1-\lambda)b) \geq \min\{\mu_A(a), \mu_A(b)\}$

Triangular Fuzzy Number (TFN): A TFN $\tilde{A}$ having membership function μ_A has positive membership in an interval [a,c] and has membership value 1 at only one point $b \in [a,c]$. It is denoted by $\tilde{A} = (a,b,c)$ and its membership function $\mu_{\tilde{A}}$ is given by

$$\mu_{\tilde{A}}(x) = \begin{cases} \dfrac{x-a}{b-a} & \text{for} \quad a \leq x \leq b \\ \dfrac{c-x}{c-b} & \text{for} \quad b \leq x \leq c \\ 0 & \text{otherwise} \end{cases} \tag{5}$$

Parabolic fuzzy number (PFN).: A PFN $\tilde{A}$ having membership function μ_A has positive membership in an interval [a,c] and has membership value 1 at only one point $b \in [a,c]$. It is

denoted by $\tilde{A} = (a, b, c)$ and its membership function $\mu_{\tilde{A}}$ is given by

$$\mu_{\tilde{A}}(a) = \begin{cases} 1 - \dfrac{(b-x)^2}{(b-a)^2} & \text{for} \quad a \le x \le b \\ 1 - \dfrac{(x-b)^2}{(c-b)^2} & \text{for} \quad b \le x \le c \\ 0 & \text{otherwise} \end{cases} \tag{6}$$

Fuzzy extension principle: Let $f : (X_1 \times X_2 \times ... \times X_n) \to Y$ be a function and $\tilde{A}_1$, $\tilde{A}_2$,..., $\tilde{A}_n$ are fuzzy sets defined on X_1, X_2,..., X_n, respectively. Then $\tilde{B} = f(\tilde{A}_1 \times \tilde{A}_2 \times ... \times \tilde{A}_n)$ is a fuzzy set defined on Y whose membership function is defined by the extension principle and the set is given by:

$$\tilde{B} = \{(y, \mu_B(y)) | \mu_B(y) = \max_{x_1 \in X_1, x_2 \in X_2, ..., x_n \in X_n} \{\min(\mu_{A_1}(x_1), \mu_{A_2}(x_2), ..., \mu_{A_n}(x_n)), y = f(x_1, x_2, ..., x_n)\}\} \tag{7}$$

α-cut of a fuzzy number: Let $\tilde{F}$ be a fuzzy number with membership function μ_F. Then α-cut of $\tilde{F}$ is a subset of $\mathbb{R}$ denoted by $\tilde{F}_\alpha$ and defined as

$$\tilde{F}_\alpha = \{x \in \mathbb{R} : \mu_F(x) \ge \alpha\} \tag{8}$$

Credibility Measure(Cr): If $\tilde{A}$ and $\tilde{B}$ be two fuzzy numbers. Then $\tilde{A} \star \tilde{B}$ is a fuzzy event, where $* \in \{<, >, \le, \ge, =\}$. Considering the level of uncertainty as the semantics of fuzzy numbers, credibility measure of the event $\tilde{A} \star \tilde{B}$ is denoted by $Cr(\tilde{A} \star \tilde{B})$ and is defined as

$$Cr(\tilde{A} \star \tilde{B}) = \frac{1}{2}[Pos(\tilde{A} \star \tilde{B}) + Nes(\tilde{A} \star \tilde{B})] \tag{9}$$

$$\text{where } Pos(\tilde{A} \star \tilde{B}) = \sup\{\min(\mu_{\tilde{A}}(a), \mu_{\tilde{B}}(b)) : a, b \in \mathbb{R}, \text{ such that } a * b \text{ holds }\} \tag{10}$$

$$\text{and } Nes(\tilde{A} \star \tilde{B}) = 1 - Pos(\overline{\tilde{A} \star \tilde{B}}), \text{ where } \overline{\tilde{A} \star \tilde{B}} \text{ denotes the complement of } \tilde{A} \star \tilde{B} \tag{11}$$

Lemma 1[10]: For any two TFNs $\tilde{A} = (A_1, A_2, A_3)$ and $\tilde{B} = (B_1, B_2, B_3)$

$$Pos(\tilde{A} > \tilde{B}) = \begin{cases} 0 & A_3 \le B_1 \\ \frac{A_3 - B_1}{B_2 - B_1 + A_3 - A_2} & A_3 \ge B_1 \text{ and } A_2 \le B_2 \\ 1 & A_2 \ge B_2 \end{cases} \tag{12}$$

$$\text{and } Nes(\tilde{A} > \tilde{B}) = \begin{cases} 0 & A_2 \ge B_2 \\ \frac{A_2 - B_2}{B_3 - B_2 + A_2 - A_1} & B_2 \le A_2 \text{ and } A_1 \le B_3 \\ 1 & A_1 \ge B_3 \end{cases} \tag{13}$$

Lemma 2[10]: For any two PFNs $\tilde{A} = (A_1, A_2, A_3)$ and $\tilde{B} = (B_1, B_2, B_3)$

$$Pos(\tilde{A} > \tilde{B}) = \begin{cases} 0 & A_3 \le B_1 \\ 1 - (\frac{B_2 - A_2}{A_3 - A_2 + B_2 - B_1})^2 & A_3 \ge B_1 \text{ and } A_2 \le B_2 \\ 1 & B_2 \le A_2 \end{cases} \tag{14}$$

$$\text{and } Nes(\tilde{A} > \tilde{B}) = \begin{cases} 0 & B_2 \ge A_2 \\ 1 - (\frac{A_2 - B_2}{B_3 - B_2 + A_2 - A_1})^2 & B_2 \le A_2 \text{ and } A_1 \le B_3 \\ 1 & A_1 \ge B_3 \end{cases} \tag{15}$$

Lemma 3:If for any fuzzy event $\tilde{A} * \tilde{B}$, $Nes(\tilde{A} * \tilde{B}) > 0$ then $Cr(\tilde{A} * \tilde{B}) > 0.5$.

Proof $Nes(\tilde{A} * \tilde{B}) > 0 \Rightarrow 1 - Pos(\overline{\tilde{A} * \tilde{B}}) > 0 \Rightarrow Pos(\overline{\tilde{A} * \tilde{B}}) < 1 \Rightarrow Pos(\tilde{A} * \tilde{B}) = 1$

$\Rightarrow$ $Cr(\tilde{A} \star \tilde{B}) = \frac{1}{2}[Pos(\tilde{A} \star \tilde{B}) + Nes(\tilde{A} \star \tilde{B})] \Rightarrow Cr(\tilde{A} * \tilde{B}) > 0.5$

Lemma 4[20]: For any fuzzy event $\tilde{A} * \tilde{B}$, $Cr(\tilde{A} * \tilde{B}) + Cr(\overline{\tilde{A} * \tilde{B}}) = 1$.

Lemma 5[10]: For any two TFNs $\tilde{A} = (A_1, A_2, A_3)$ and $\tilde{B} = (B_1, B_2, B_3)$

$$Cr(\tilde{A} < \tilde{B}) = \begin{cases} 0 & A_3 \leq B_1 \\ \frac{1}{2}(1 + \frac{B_2 - A_2}{A_3 - A_2 + B_2 - B_1}) & A_2 \leq B_2 \text{ and } B_1 < A_3 \\ \frac{1}{2}(\frac{B_3 - A_1}{B_3 - B_2 + A_2 - A_1}) & B2 < A2 \text{ and } B_3 > A_1 \\ 0 & \mathrm{B}_3 < A_1 \end{cases} \tag{16}$$

Rough variable[19]**:** A rough variable $\hat{y}$ is a quantifiable function within the rough space $(\Lambda, \Delta, \kappa, \pi)$ to the set of real numbers $\mathbb{R}$. That is for every Borel set B of $\mathbb{R}$, we have $\{\lambda \in \Lambda | \hat{y}(\lambda) \in B\} \in \kappa$. The lower and upper approximations of the rough variable, $\hat{y}$, are defined as $\underline{\hat{y}} = \{\hat{y}(\lambda) | \lambda \in \Delta\}$ and $\overline{\hat{y}} = \{\hat{y}(\lambda) | \lambda \in \Lambda\}$ respectively.

Trust measure[19]**:** Let $(\Lambda, \Delta, \kappa, \pi)$ be a rough space. Then the trust measure of a rough event R is denoted by $Tr(R)$ and is defined as

$$Tr(R) = \frac{1}{2}[\underline{Tr}(R) + \overline{Tr}(R)]$$

where $\underline{Tr}(R) = \frac{\pi(R \cap \Delta)}{\pi(\Delta)}$ and $\overline{Tr}(R) = \frac{\pi(R)}{\pi(\Lambda)}$ are the lower and upper trust measure of the rough event R respectively. A rough set normally denoted by using an ascent ˆ over the name.

Lemma 6[23]**:** Let Lebesgue measure is used for trust measure and $\hat{R}_1 = ([p_1, q_1][r_1, s_1])$, $\hat{R}_2 = ([p_2, q_2][r_2, s_2])$ be two rough variables. Then trust measure of the rough event $\hat{R}_1 < \hat{R}_2$ is denoted by $Tr(\hat{R}_1 < \hat{R}_2)$ and is given by

$$Tr(\hat{R}_1 < \hat{R}_2) = \begin{cases} 0 & \text{for } \ s_2 - r_1 \leq 0 \\ \frac{s_2 - r_1}{2(s_2 - r_1 - r_2 + s_1)} & \text{for } \ q_2 - p_1 \leq 0 \leq s_2 - r_1 \\ \frac{1}{2}\left(\frac{s_2 - r_1}{s_2 - r_1 - r_2 + s_1} + \frac{q_2 - p_1}{q_2 - p_1 - p_2 + q_1}\right) & \text{for } \ p_2 - q_1 \leq 0 \leq q_2 - p_1 \\ \frac{1}{2}\left(\frac{s_2 - r_1}{s_2 - r_1 - r_2 + s_1} + 1\right) & \text{for } \ r_2 - s_1 \leq 0 \leq p_2 - q_1 \\ 1 & \text{for } \ r_2 - s_1 \geq 0 \end{cases} \tag{17}$$

Lemma 7[20]: For any rough event $\hat{A} * \hat{B}$, $* \in \{<, \leq, >, \geq\}$, $Tr(\hat{A} * \hat{B}) + Tr(\overline{\hat{A} * \hat{B}}) = 1$.

CSP in imprecise environment: In the problem (4) when some of the cost parameters are imprecise(Fuzzy/Rough etc.) in nature then the goal of the problem reduces to the determination of optimal path under minimization of imprecise objective. So the problem reduces to optimization in the imprecise environment(Fuzzy/Rough etc.)

5. Optimization in imprecise environment

Let us consider an optimization problem in the form:

$$\left.\begin{array}{ll} \text{Maximize} & f(x, y) \\ \text{subject to} & \phi_i(x, y) \leq 0, i = 1, 2, ...p \end{array}\right\} \tag{18}$$

Here $x = (x_1, x_2, ..., x_n)$ is a decision vector having n variables, $y = (y_1, y_2, ..., y_m)$ is a vector representing, m, parameters of the problem and $\phi_i(x, y)$, $i = 1, 2, ...p$ are constraint functions.

In any classical optimization problem, the parametric values are crisp in nature. When the parameters are imprecise in nature, the problem (18) reduces to optimization in imprecise environment.

5.1. **Optimization in fuzzy environment.** In the problem (18), if y is a vector of fuzzy numbers, $\tilde{y} = (\tilde{y}_1, \tilde{y}_2, ..., \tilde{y}_m)$, the problem (18) reduces to an optimization problem in fuzzy environment having n crisp decision variables, $x_1, x_2, ..., x_n$, and is presented below:

$$\left.\begin{array}{ll} \text{Maximize} & \tilde{f}(x, \tilde{y}) \\ \text{Subject to} & \tilde{\phi}_i(x, \tilde{y}) \leq 0, i = 1, 2, ...p \end{array}\right\} \tag{19}$$

clearly here objective function as well as the constraints are fuzzy in nature. Optimization in fuzzy environment is not properly defined till date. Moreover, there is no proper guideline to check whether the value of a decision vector is feasible with respect to the constraints involve in the problem. In problem(19) the value x_0 of the decision vector x may said to be feasible if the following condition holds

$$Cr(\tilde{\phi}_i(x, \tilde{y}) \leq 0) > 0.5, \forall\, i \in \{1, 2, ...p\} \tag{20}$$

It is a valid approach as $Cr(\tilde{\phi}_i(x, \tilde{y}) \leq 0) + Cr(\tilde{\phi}_i(x, \tilde{y}) > 0) = 1$(§Lemma 4). Similarly, a feasible solution $x^{'}$ is better than another feasible solution $x^{''}$ if

$$Cr(\tilde{f}(x^{'}, \tilde{y}) > \tilde{f}(x^{''}, \tilde{y})) > 0.5 \tag{21}$$

If the analytical form of the membership functions of the constraint functions $\tilde{\phi}_i(x, \tilde{y})$, $i =$ 1, 2, ..., p are available then similar as, Lemma 5, one can easily determine the values of $Cr(\tilde{\phi}_i(x, \tilde{y}) \leq 0)$, $i = 1, 2, ..., p$ for the determination of the feasibility of any value of the decision vector, x. Similarly if the analytical form of the membership function of the fuzzy objective $\tilde{f}(x, \tilde{y})$ is available then one can easily determine the values of $Cr(\tilde{f}(x^{'}, \tilde{y}) > \tilde{f}(x^{''}, \tilde{y}))$ for the determination of the better option between two feasible values, $x^{'}$ and $x^{''}$ of the decision vector, x. But if the analytical form of the membership function of any constraint or the analytical form of the membership function the objective function is not available then fuzzy simulation approach can be used for the validation of any constraint in (20) or (21). It can be done using the following algorithm.

Algorithm 1. Algorithm to verify $Cr(\tilde{\phi}_i(x, \tilde{y}) \leq 0) > 0.5$ for a particular value of the decision vector x.

From Lemma-3, it can be stated that, for a decision vector x if $Nes(\tilde{\phi}_i(x, \tilde{y}) \leq 0) > 0$ then $Cr(\tilde{\phi}_i(x, \tilde{y}) \leq 0) > 0.5$. Also $Nes(\tilde{\phi}_i(x, \tilde{y}) \leq 0) > \alpha \Rightarrow Pos(\tilde{\phi}_i(x, \tilde{y}) \geq 0) < 1 - \alpha$. So $Cr(\tilde{\phi}_i(x, \tilde{y}) \leq 0) > 0.5 \Leftrightarrow Pos(\tilde{\phi}_i(x, \tilde{y}) \geq 0) < 1 - \alpha$ for any $\alpha > 0$. The following algorithm, checks the feasibility of the statement $Pos(\tilde{\phi}_i(x, \tilde{y}) \geq 0) < 1 - \alpha$, i.e., feasibility of the statement $Cr(\tilde{\phi}_i(x, \tilde{y}) \leq 0) > 0.5$.

1. Input x
2. Initialize N
3. $j \leftarrow 1$
4. Select randomly one vector y_0 from $[\tilde{y}]_{1-\alpha}$
5. If $\phi_i(x, y_0) \geq 0$
6. Return Infeasible
7. End If
8. $j \leftarrow j + 1$
9. If $j \leq N$
10. Go to step 4.
11. End If

12. Return Feasible
13. End Algorithm

5.2. **Optimization in rough environment.** In the problem (18), if y is a vector of rough numbers, $\hat{y} = (\hat{y}_1, \hat{y}_2, ..., \hat{y}_m)$, the problem (18) reduces to an optimization problem in rough environment having n crisp decision variables, $x_1, x_2, ..., x_n$, and is presented below:

$$\left.\begin{array}{ll} \text{Maximize} & \hat{f}(x, \hat{y}) \\ \text{Subject to} & \hat{\phi}_i(x, \hat{y}) \leq 0, i = 1, 2, ...p \end{array}\right\} \tag{22}$$

clearly here objective function as well as the constraints are rough in nature. Optimization in rough environment is not properly defined till date. Moreover, there is no proper guideline to check whether the value of a decision vector is feasible with respect to the rough constraints involve in the problem. In problem(22) the value x_0 of the decision vector x may said to be feasible if the following conditions hold

$$Tr(\hat{\phi}_i(x_0, \hat{y}) \leq 0) > 0.5, \forall\, i \in \{1, 2, ...p\} \tag{23}$$

It is a valid approach as $Tr(\hat{\phi}_i(x_0, \hat{y}) \leq 0) + Tr(\tilde{\phi}_i(x_0, \tilde{y}) > 0) = 1$(§Lemma 7). Similarly, a feasible solution $x^{'}$ is better than another feasible solution $x^{''}$ if

$$Tr(\hat{f}(x^{'}, \hat{y}) > \hat{f}(x^{''}, \hat{y})) > 0.5 \tag{24}$$

If the analytical form of the constraint functions $\hat{\phi}_i(x, \tilde{y})$, $i = 1, 2, ..., p$ are available then similar as, Lemma 6, one can easily determine the values of $Tr(\hat{\phi}_i(x, \hat{y}) \leq 0),\ i = 1, 2, ..., p$ for the determination of the feasibility of any value of the decision vector, x. Similarly if the analytical form of the rough objective $\hat{f}(x, \hat{y})$ is available then one can easily determine the values of $Tr(\hat{f}(x^{'}, \hat{y}) > \hat{f}(x^{''}, \tilde{y}))$ for the determination of the better option between two feasible values, $x^{'}$ and $x^{''}$ of the decision vector, x. But if the analytical form of any constraint or the analytical form of the objective function is not available then rough simulation approach can be used for the validation of any constraint in (23) or (24). It can be done using the following algorithm.

Algorithm 2. Algorithm for the determination of $Tr(\hat{\phi}_i(x, \hat{y}) \leq 0)$ for a particular value of the decision vector x.

Let $\hat{y}$ be an m dimensional rough vector on the rough space $(\Lambda, \Delta, \kappa, \pi)$ then $\hat{\phi}(x, \hat{y})$ is also a rough vector on that space. To find the trust measure of the rough event $\hat{\phi}(x, \hat{y}) \leq 0$, the following algorithm can be used:

1. Let $n_1 = n_2 = 0$ and set value of N, sufficiently large.
2. Generate $\underline{\lambda}$ uniformly from Δ according to the measure π
3. If $\phi(x, y(\underline{\lambda})) \leq 0$ then $n_1 = n_1 + 1$.
4. Generate $\overline{\lambda}$ uniformly from κ according to the measure π
5. If $\phi(x, y(\overline{\lambda})) \leq 0$ then $n_2 = n_2 + 1$.
6. Repeat the steps 2-5 for N times.
7. $Tr(\hat{\phi}(x, \hat{y}) \leq 0) = (n_1 + n_2)/(2N)$
8. End algorithm

6. Ant Colony Optimization

Following different behaviours of ants for searching their food sources, the ACO algorithm was proposed by Dorigo et al.[4] in 1997 to solve TSPs. In the algorithm, the path of an ant from the nest to the food source is analogous to the path of a salesman in TSP. Ant algorithm are multi-agent system in which the behaviour for each single agent, called artificial ant or ant, follows real ants's behaviour. Nowadays, a large no of algorithms on ant base has been available in the literature. The purpose of the algorithm is to find a minimum distance path

from the source to the destination. Analogous to the real ant, in the algorithm, every artificial ant has a chemical called pheromone. When the ant travels from one place to another, it is left on the path. The path in which density of pheromone is maximum is the shortest path in reality. In the algorithm also this phenomenon is used to find the shortest path for the salesman.

In the algorithm, in the iteration, t, an ant k, which is currently located at node i, selects the next node j, depending on a probability, $P_{ij}^k(t)$, using some probabilistic selection process, e.g., for Roulette Wheel selection process[15]:

$$P_{ij}^k(t) = \begin{cases} \frac{\tau_{ij}^\alpha(t)\eta_{ij}^\beta(t)}{\sum_{u \in N_i^k} \tau_{iu}^\alpha(t)\eta_{iu}^\beta(t)} & \text{if} \quad j \in N_i^k(t) \\ 0 & \text{if} \quad j \notin N_i^k(t). \end{cases} \tag{25}$$

where τ_{ij} represents the pheromone value and η_{ij} represents the heuristics value of the move from node i to j at time step t. $N_i^k(t)$ represent the set of nodes which are not yet visited by ant k (when it is at node i). α and β are positive real parameters whose values determine the relative importance of the pheromone versus the heuristics information. η_{ij} is calculated by following equation,

$$\eta_{ij} = \frac{1}{d_{ij}} \tag{26}$$

where d_{ij} is the distance (cost) between the node i and j.

Pheromone Evaporation. At the time of visiting the nodes by an ant, some amount of pheromone are evaporated from each edge and some pheromone are deposited on the edges through which the ant move. For each edge (i, j), evaporation takes place using the following rule:

$$\tau_{ij}(t+1) \quad = \quad \rho\tau_{ij}(t) \tag{27}$$

with $\rho \in [0, 1]$. ρ is the constant, that specifies pheromone evaporate rate.

Pheromone update rule. After completion of a tour (path) from the source to the destination by all the ants, the pheromone on each edge (*i*,*j*) through which any ant moves is updated (due to deposition of pheromone) as

$$\tau_{ij}(t+1) \quad = \quad \tau_{ij}(t) + \sum_{k=1}^{n_k} \triangle\tau_{ij}^k(t) \tag{28}$$

where $\triangle\tau_{ij}^k(t)$ is the amount of pheromone deposited by the ant k on the edge(*i*,*j*) at time step t and here $\triangle\tau_{ij}^k(t)$ is taken as

$$\triangle\tau_{ij}^k(t) = \begin{cases} \frac{1}{f(X_k)} & \text{if k-th ant passes through the edge (i,j)} \\ 0 & \text{otherwise} \end{cases} \tag{29}$$

where X_k is the path of k-th ant and $f(X_k)$ is the distance of the path. For detail algorithm of ACO please see [10].

7. K-Opt perturbation Operation for CSP

K-Opt [26] is mainly used to improve the tour cost of a CSP. It is also called a tour improvement algorithm. The mechanism of K-Opt operation is to break a feasible tour into K-parts, rejoin different permutations of the parts to create different feasible tours and select the best among them for the improvement of the original tour. Here K-Opt operation is applied on a complete tour of a CSP for its possible improvement. For detailed of K-Opt operation, one can refer [11].

2-Opt algorithm removes one edge form the tour, and reconnects the two sub-tours' combination and reverses of their path to find the better movement. It is continued until no improvement is found using the process. Similarly in the case of 3-Opt, breaking 2 edges of a tour, eight possible new combinations can be found. In this way we continue to break edges form tour i.e. K = 1, 2, 3..., n and get new algorithm, like 2-Opt, 3-Opt, 4-Opt and so on. But the increase of K increases the time complexity of the algorithm. For that reason, here, the 3-Opt operation is applied and it is found that it acts better than the 2-Opt operation for large size problems.

7.1. **K-Opt Operation on a complete tour :** The detailed algorithm of K-Opt operation for $K = 3$ is presented below. Here a one-dimension array $bestpath_i$ is used to represent i-th path/tour of the CSP obtained using ACO. Then two cities of the path are randomly selected to divide the path into three sub-tour and let these be $\text{bestpath}_{ij}, j = 1, 2, 3$. The reverse of these sub-tours are denoted by $\text{bestpath}^r_{ij}, j = 1, 2, 3$ respectively. Then eight new tours, $\text{nbestpath}_{ij}, j = 1, 2, ..8$, can be formed using $\text{bestpath}_{ij}, j = 1, 2, 3$ and $\text{bestpath}^r_{ij}, j = 1, 2, 3$ as below:

nbestpath_{i1}=$\text{bestpath}_{i1} \rightarrow \text{bestpath}_{i2} \rightarrow \text{bestpath}_{i3}$
nbestpath_{i2}=$\text{bestpath}_{i1} \rightarrow \text{bestpath}^r_{i2} \rightarrow \text{bestpath}_{i3}$
nbestpath_{i3}=$\text{bestpath}_{i1} \rightarrow \text{bestpath}_{i2} \rightarrow \text{bestpath}^r_{i3}$
nbestpath_{i4}=$\text{bestpath}_{i1} \rightarrow \text{bestpath}^r_{i3} \rightarrow \text{bestpath}^r_{i2}$
nbestpath_{i5}=$\text{bestpath}_{i1} \rightarrow \text{bestpath}_{i3} \rightarrow \text{bestpath}^r_{i2}$
nbestpath_{i6}=$\text{bestpath}_{i1} \rightarrow \text{bestpath}^r_{i3} \rightarrow \text{bestpath}_{i2}$
nbestpath_{i7}=$\text{bestpath}_{i1} \rightarrow \text{bestpath}^r_{i2} \rightarrow \text{bestpath}^r_{i3}$
nbestpath_{i8}=$\text{bestpath}_{i1} \rightarrow \text{bestpath}^r_{i3} \rightarrow \text{bestpath}_{i2}$

Among these new tours minimum cost tour is the best found movement using 3-Opt operation once. If cost of this tour is less than the parent tour then it replaces the parent tour.

8. Proposed algorithm for Covering Salesman problem

As stated earlier, the proposed algorithm consists of two procedures. The first procedure divides the node set of the problem into groups in such a manner that all the nodes of any group located within the restricted covering distance r (of the problem) from the center of the respective group. The second procedure determines the shortest route through the centres of the group. In the algorithm N represents number of nodes/cities, NB represents number of groups, NC represents maximum number of cities in a group. At first, equal no of nodes are selected in all the groups. The remaining nodes are successively added to different groups depending upon the distance of the node (nearest to the centre element of the group) so that cardinality of a group does not exceed NC. NE is the equal number of nodes in each group, i.e., $NE = [N/NB]$, where $[N/NB]$ represents an integral part of N/NB. $G[i]$ represents i-th group, l is the cardinality of the group, i.e., $G[i].l$ is the length of group i, an array $G[i].S$ represents the node set of group i and $max\_iteration$ is the maximum number of iterations and N is the no of nodes. $(d_{ij})_{N\times N}$ represents the distance matrix of the problem.

1. **Start Algorithm**
2. Set NC, NB, $max\_iteration$, $max\_iteration2$, N.
3. Input $(d_{ij})_{N\times N}$
4. **For** $i = 1$ to $max\_iteration$ **do**
5. Set $G[k].l = 0$ for $k = 1, 2, ..., NB$
6. Set $NS = \{V_1, V_2, ...V_N\}$
7. Set $G[i].S = \{\}$
8. v_c= a randomly selected element from NS
9. $G[1].S = G[1].S \cup \{v_c\}$

$NS = NS - \{v_c\}$
$G[1].l = G[1].l + 1$
For $j = 2$ to NE **do**
  v= Nearest city of v_c in NS
  $G[1].S = G[1].S \cup \{v\}$
  $G[1].l = G[1].l + 1$
  $NS = NS - \{v\}$
End For
For $k = 2$ to NB **do**
  v_c= Nearest city of v_c in NS
  $G[k].S = G[k].S \cup \{v_c\}$
  $NS = NS - \{v_c\}$
  $G[k].l = G[k].l + 1$
  For $j = 2$ to NE **do**
    v= Nearest city of v_c in NS
    $G[k].S = G[k].S \cup \{v\}$
    $G[k].l = G[k].l + 1$
    $NS = NS - \{v\}$
  End For
End For
set $count = 0$
While $NS \neq \{\}$ **do**
  For $k = 1$ to NB **do**
    If $G[k].l \leq NC$ **then**
      $count = count + 1$
      $b_{count} = k$ where b is an array
    End if
  End For
  v= first element from NS
  $NS = NS - \{v\}$
  $min = d_{vG[b_1].S[G.[b_1].l]}$
  $pos = b_1$
  For $k = 2$ to $count$ **do**
    If $(d_{vG[b_k].S[G.[b_k].l]} < min)$ **then**
      $min = d_{vG[b_k].S[G.[b_k].l]}$
      $pos = b_k$
    End If
  End For
  $G[pos].S = G[pos].S \cup \{v\}$
  $G[pos].l = G[pos].l + 1$
End While
For $k = 1$ to $maxiteration2$ **do**
  set $flag = 0$
  For $j = 1$ to NB **do**
    z=a randomly selected integer in the range $[1, G[j].l]$
    $G[j].center = G[j].S[z]$
    For $k = 1$ to $G[j].l$ **do**
      If $(d_{G[j].S[k]G[j].center} > r)$ **then**
        $flag = 1$
        break

End If
End For
If $flag = 1$ **then**
break
End If
End For
if $(flag = 0)$ **then**
ACO algorithm is used to find the best path according the centres of the groups.
K-opt operation is applied on the best path for further possible improvement.
The best solution found so far, $solution_b$, is updated.
End If
End for
End for
Output $solution_b$
End algorithm

8.1. **Exploration and exploitation.** For any CSP instance, a particular division of the groups of cities and the selection of visiting nodes may converge to local optima. To explore different possible paths, the ACO is applied on the different selections of the set of visiting cities. In this study, K-opt operation is applied at the end of the ACO algorithm to avoid convergence of the path at any local optima. If the ACO converges to a local optima, then the application of 3-opt on the obtained path of ACO will find a better path and the repeated application of 3-opt on the improved paths may obtain the global optimal path. In this way, the exploration and exploitation is made in the proposed algorithm.

8.2. **Implementation and testing.** The algorithm is implemented in Dev C++ in a computer having Intel core-i3 first generation processor and 2 GB RAM. The algorithm is tested against a set of benchmark test problems from TSPLIB with significantly large sizes (size up to 654 nodes). The performance of the algorithm is compared with different existing successful algorithms for CSP in the literature. It is observed that the efficiency of the proposed algorithm is comparatively better with respect to the other existing algorithms used for the comparison.

9. Numerical Illustration

So far the author's knowledge go, in the literature, there is no algorithm that presents results of benchmark CSP instances with specified coverage range r. All the studies have been made where groups are created with the nearest nodes from the selected centres of the groups. Due to this reason, here, to measure the efficiency of the proposed algorithm, some standard benchmark test instances are used that are proposed by Golden et al. [7]. These instances are generated form TSPLIB [5]. The test problems have been divided into small, medium, and large size problems according to their sizes. The small size and medium size problems contain 51 to 200 number of nodes where each node can cover nearest 7, 9, 11 number of nodes and the large size problems contain 532 to 654 number of nodes where each node can cover 3, 5, 7 number of nearest nodes. Other notations and symbols are the same as previously stated.

The problems are solved using the proposed algorithm for CSPs and the results are tabulated in Table-1 and table-2. The goal is to minimize the total tour cost of the Hamiltonian path through the visiting nodes. It is observed from the tables that in most of the considered instances tour cost increase with the number of groups, which agrees with reality. In very few cases, the algorithm gives higher tour cost for larger group size, due to the division of the groups. But in those cases also the proposed algorithm gives better results compared to the other well-established

algorithms used for the comparison. The convergence graphs of the best found solutions for some instances are presented in figure-1.

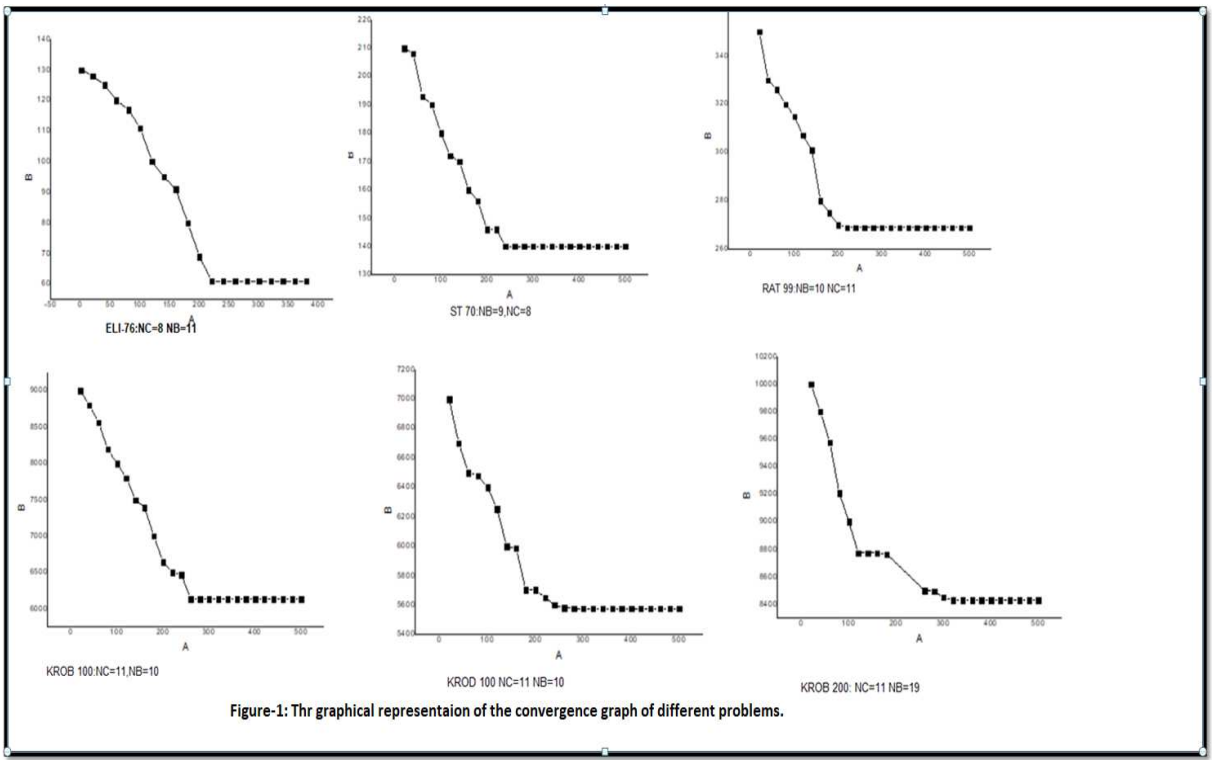

Figure-1: Thr graphical representaion of the convergence graph of different problems.

Table-3 and Table-4 present the comparative study of the computational results of the proposed algorithm with respect to seven other state-of-the-art algorithms for the CSPs. Table-3 represents the best found cost of the different instances obtained by the different well-established algorithms for CSPs in the literature along with the proposed algorithm. Table-4 represents the average cost and the standard deviation of the results obtained by the different algorithms in five different runs for the different instances. It is observed from the tables that the proposed algorithm obtains the best tour cost as well as the minimum average cost for each of the instances with different numbers of NCs. From these observations, it can be concluded that the efficiency of the proposed algorithm is better compared to the other algorithms in the literature for solving the CSPs.

Table-5 and Table-6 present the results due to the different values of the coverage range r for some large size test instances. It is observed from the tables that for each of the instances, the tour cost increases with the decrease of r. In Table-7 also, the results due to the different values of the coverage range r are presented for all the considered test instances. But, here, at first result due to a fixed coverage range, r, is obtained, which is presented in the column marking 100%. Then results are obtained by decreasing the coverage range by taking its value 80%, 70%, 60% and 50% of r. In some cases, the algorithm could not find any result. In those cases result positions in the table are kept $BLANK$. It is observed from Table-7 that the tour costs increases gradually as the covering range decreases gradually. Clearly, this observation also agrees with reality.

Again results are obtained for the restricted problems where the Hamiltonian path of a CSP is

searched under the restriction of a specific coverage range of different groups, i.e., total distance of different cities of a group from the visiting city should not exceed a predefined level. For this study the group coverage range is considered as $NC * r/3$ for any problem, where the symbols are previously defined and the value of r is mentioned in the table. Results are obtained for different problems and are presented in Table-8. If, for any case, solution does not exists, then the solution position of such problems are kept $BLANK$ in the table.

As this is the initial study on imprecise CSP, no standard test problems on fuzzy CSP exists in the literature. Due to this reason, test problems with fuzzy cost parameters are generated from the previously considered standard benchmark test problems of CSPs with crisp parameters. To test the proposed algorithm in a fuzzy environment these fuzzy CSPs are used. In a crisp CSP, if travel cost between node i and node j is d_{ij} then following Khan et al. [12, 13], the corresponding fuzzy cost is generated as $\tilde{d_{ij}} = (d_{ij1}, d_{ij2}, d_{ij3})$, where $d_{ij2} = d_{ij}$, $d_{ij1} = d_{ij} - R1$, $d_{ij3} = d_{ij} + R2$, where $R1$ and $R2$ are randomly generated between $(0, R \times d_{ij}/100)$. Here R is the percentage of fuzziness and its value is considered as 0.5, i.e., here, maximum 0.5% fuzziness is considered for the fuzzy costs. Table-9 presents the computational results of these problems using the proposed algorithm in fuzzy environment. As cost matrices are generated with 0.5% fuzziness the tour costs of the best found paths are near to the corresponding crisp problems and it implies that the proposed algorithm is efficient enough to solve CSPs with fuzzy cost matrices. Results are obtained following both the direct credibility measure approach and the simulation approach for the same and the same results are obtained. As a result only one result table is presented, as, the same results are obtained using two approaches.

Due to the same reason as the fuzzy CSPs, there are no standard test problems on rough CSPs in the literature. Due to this reason, test problems with rough cost parameters are generated from the previously considered standard benchmark test problems of CSPs with crisp parameters. To test the proposed algorithm in a rough environment these generated CSPs with rough cost parameters are used. If in a crisp CSP, travel cost between node i and node j, is d_{ij}, then following Khan et al. [13], the corresponding cost in the rough environment is generated as $\tilde{d_{ij}} = (d_{ij1}, d_{ij2}, d_{ij3}, d_{ij4})$, where $d_{ij1} = d_{ij}$, $d_{ij2} = d_{ij} + R1$, $d_{ij3} = d_{ij} - R2$, $d_{ij3} = d_{ij1} + R3$ and $R1$, $R2$, $R3$ are randomly generated in $(0, R \times d_{ij}/100)$. Here R is the percentage of roughness and its value is considered as 0.5, i.e., here maximum 0.5% roughness is considered for the rough costs. Table-10 presents the computational results of these problems using the proposed algorithm in rough environment. As cost matrices are generated with 0.5% roughness the tour costs of the best found paths are near to the corresponding crisp problems and it implies that the proposed algorithm is efficient enough to solve CSPs with rough cost matrices. Results are obtained following both the direct trust measure approach and the simulation approach and the same results are obtained. As a result only one result table is presented, as, the same results are obtained using two approaches.

10. Conclusion

Here for the first time, the ACO algorithm is modified with K-opt operation to develop an efficient and consistent algorithm for CSPs under one restriction in crisp and imprecise (fuzzy, rough) environments. An algorithm is proposed for the division of groups of the cities depending upon the maximum number of cities in a group and the total number of groups. The ACO is used to find the shortest/minimum-cost path of any problem by selecting only one city from each group in such a manner that all the unvisited cities are within a predefined coverage range r. K-opt algorithm is used at the end of ACO operation for the possible improvement of the quality of the solution. Here, study is made only in two imprecise environments- fuzzy environment and rough environment. To solve the problem in imprecise environment some approach is followed

where tour is searched without transferring the imprecise optimisation problem into a crisp optimisation problem. Moreover, fuzzy and rough simulation approaches are proposed to solve the problem for type of fuzzy data set and any measure of rough data set. For the restricted problems path are searched in such a manner that the total distance/cost of different cities of a group from the visited city of the group should not exceed a predefined limit. The algorithm is tested with standard benchmark crisp problems available in the literature. To test the algorithm in the imprecise environment, imprecise instances are derived randomly from the standard crisp instances using a specified rule. Different numerical studies and the comparison studies with different well-established algorithms for CSPs establish the efficiency of the proposed approach for solving the CSPs in the crisp as well as in the imprecise environments. The approaches presented in this study is applicable in any routing problems, like, TSP, GTSP, CTP, BCSP, Vehicle Routing Problem, etc., in different environments. The simulation approaches presented here can be applied to solve different real life optimisation problems in science and technology. In future work, stochastic simulation approach can be used to deal with the problem with the random data sets. Also, the study can be done with the mixed data sets, i.e., the data sets involving different types of imprecise data. Multi-objective CSPs in imprecise environments can also be studied in the future.

Table-1: The computational results of the proposed algorithm

Instance	NC	Best found tour cost						
berlin52	**7**	NB	**8**	9	10	11	12	13
		Best cost	**1459**	1540	2157	1639	2361	2607
	9	NB	**6**	7	8	9	10	11
		Best cost	**1112**	1180	1455	1528	2157	1647
	11	NB	5	**6**	7	8	9	10
		Best cost	1029	**953**	1122	1455	1547	2157
st70	**7**	NB	**10**	11	12	13	14	15
		Best cost	**172**	185	205	212	235	243
	9	NB	**8**	9	10	11	12	13
		Best cost	**140**	158	176	181	204	222
	11	NB	**7**	8	9	10	11	12
		Best cost	**108**	131	164	176	181	201
eil76	**7**	NB	**11**	12	13	14	15	16
		Best cost	**118**	130	139	147	152	155
	9	NB	**9**	10	11	12	13	14
		Best cost	**90**	93	115	125	140	152
	11	NB	7	**8**	9	10	11	12
		Best cost	62	**61**	89	108	109	125
rat99	**7**	NB	**15**	16	17	18	19	20
		Best cost	**382**	400	386	411	439	428
	9	NB	**12**	13	14	15	16	17
		Best cost	**201**	341	359	392	404	377
	11	NB	**10**	11	12	13	14	15
		Best cost	**269**	313	332	327	365	379
kroA100	**7**	NB	15	**16**	17	18	19	20
		Best cost	**7365**	7857	7760	8347	8774	9349
	9	NB	12	**13**	14	15	16	17
		Best cost	6641	**6236**	7616	7804	8150	7772
	11	NB	**10**	11	12	13	14	15
		Best cost	**5651**	6485	6460	6293	7616	7741
kroB100	**7**	NB	**15**	16	17	18	19	20
		Best cost	**7538**	8341	8180	8676	9268	9147
	9	NB	**12**	13	14	15	16	17
		Best cost	**6816**	6913	7597	7975	8426	8013
	11	NB	**10**	11	12	13	14	15
		Best cost	**6135**	6349	6463	7216	7597	7366
kroC100	**7**	NB	**15**	16	17	18	19	20
		Best cost	**7272**	7543	7864	8173	8644	8795
	9	NB	**12**	13	14	15	16	17
		Best cost	**6349**	6825	7317	7206	7612	7847
	11	NB	**10**	11	12	13	14	15
		Best cost	**5851**	6422	6515	6705	7484	7366
kroD100	**7**	NB	**15**	16	17	18	19	20
		Best cost	**6866**	8032	8030	8647	9084	9221
	9	NB	**12**	13	14	15	16	17
		Best cost	**6595**	6846	7778	7176	7538	7924
	11	NB	**10**	11	12	13	14	15
		Best cost	**5575**	6088	6026	6890	7517	7237

Table-2: The computational results of the proposed algorithm

Instance	NC	Best found tour cost						
kroE100	**7**	NB	**15**	16	17	18	19	20
		Best cost	**7407**	8208	7862	8429	8699	9339
	9	NB	**12**	13	14	15	16	17
		Best cost	**6709**	7022	7555	7558	8010	8126
	11	NB	**10**	11	12	13	14	15
		Best cost	**6287**	6822	6730	7219	7555	7734
kroA150	**7**	NB	**22**	23	24	25	26	27
		Best cost	**9220**	9880	10226	10264	10598	10951
	9	NB	**17**	23	24	25	26	27
		Best cost	**8106**	9339	10069	10264	10668	11029
	11	NB	**15**	18	19	20	21	22
		Best cost	**7408**	7967	8671	8980	9603	9290
kroB150	**7**	NB	**22**	23	24	25	26	27
		Best cost	**9143**	9568	9512	10044	9812	10262
	9	NB	**17**	23	24	25	26	27
		Best cost	**8123**	9153	9838	10044	9837	10356
	11	NB	**15**	18	19	20	21	22
		Best cost	**7733**	8353	8691	8589	9129	9242
kroA200	**7**	NB	**29**	30	33	34	35	36
		Best cost	**10783**	10883	11851	12130	12085	12419
	9	NB	**23**	25	26	27	28	29
		Best cost	**8944**	9933	10179	10586	10847	10861
	11	NB	**19**	26	27	28	29	30
		Best cost	**8432**	10278	10545	10533	10918	11184
alt532	3	NB	**178**	179	180	181	182	183
		Best cost	**34109**	34750	34448	35565	36559	43955
	5	NB	**107**	116	117	119	120	121
		Best cost	**32476**	38483	41617	40861	41049	41845
	7	NB	**76**	95	96	97	98	99
		Best cost	**35928**	38497	39515	39939	39755	39558
ali535	3	NB	**179**	182	183	184	185	186
		Best cost	**712**	743	743	731	747	738
	5	NB	107	**115**	116	121	122	130
		Best cost	1072	**792**	802	895	908	1103
	7	NB	**77**	91	92	94	95	105
		Best cost	**664**	698	716	738	779	1026
u574	3	NB	192	**193**	194	195	196	197
		Best cos	18311	**18253**	18787	18558	18749	18640
	5	NB	**115**	127	128	129	130	131
		Best cost	**17960**	18118	18332	18711	18840	18547
	7	NB	**99**	100	101	102	103	104
		Best cost	**15598**	15915	16062	16104	16296	17114
p654	3	NB	**220**	221	235	236	246	247
		Best cost	**19921**	19975	20145	20768	21173	21298
	5	NB	**141**	142	143	180	181	191
		Best cost	**20250**	20267	20302	20893	20872	21097
	7	NB	**113**	136	137	138	139	140
		Best cost	**19381**	20145	19883	20147	20221	20264

Table-3: Performance study of the proposed algorithm with respect to different algorithms for CSP

Instance	NC	Best found tour Cost								
		CPLEX [1]	LS2 [7]	ILP [24]	Hybrid ACO [25]	Hybrid GA [22]	Hybrid ABC [22]	PVNS [31]	Hybrid ABC[29]	Proposed Algorithm
berlin52	7	4019	3878	3878	3878	3887	3887	3887	3887	**1459**
	9	3430	3430	3430	3430	3430	3430	3430	3430	**1112**
	11	3742	3262	3262	3262	3262	3262	3262	3262	**1029**
st70	7	297	288	288	288	288	288	288	288	**176**
	9	271	259	259	259	259	259	259	259	**140**
	11	269	247	247	247	247	247	247	247	**108**
eil76	7	219	207	207	207	207	207	211	207	**118**
	9	198	186	185	186	185	186	185	186	**90**
	11	177	170	170	170	170	170	170	170	**61**
rat99	7	572	486	486	486	486	486	486	486	**382**
	9	462	455	455	455	455	455	455	455	**201**
	11	456	444	444	444	444	444	444	444	**269**
kroA100	7	10306	9674	9674	9674	9674	9674	9674	9674	**7365**
	9	9573	9159	9159	9159	9159	9159	9159	9159	**6641**
	11	9460	8901	8901	8901	8901	8901	8901	8901	**5651**
kroB100	7	11123	9537	9537	9537	9537	9537	9537	9537	**7538**
	9	9505	9240	9240	9240	9240	9240	9240	9240	**6816**
	11	9049	8842	8842	8842	8842	8842	8842	8842	**6135**
kroC100	7	10367	9723	9723	9723	9723	9723	9723	9724	**7272**
	9	9952	9171	9171	9171	9171	9171	9171	9171	**6349**
	11	9150	8632	8632	8632	8632	8632	8632	8632	**5821**
kroD100	7	11085	9626	9626	9262	9626	9626	9626	9626	**6866**
	9	10564	8885	8885	8885	8885	8885	8885	8885	**6595**
	11	9175	8725	8725	8725	8725	8725	8725	8725	**5575**
kroE100	7	9095	10150	10150	10150	10150	10150	10150	10150	**7407**
	9	9095	8991	8991	8991	8991	8992	8991	8992	**6709**
	11	8936	8450	8450	8450	8450	8450	8450	8450	**6287**
rd100	7	4105	3461	3461	3461	3461	3461	3461	3461	**2424**
	9	3414	3194	3194	3194	3194	3194	3194	3194	**2171**
	11	3453	2922	2922	2922	2922	2922	2922	2922	**1933**
kroA150	7	12367	11423	11800.00	11423	11423	11423	11423	11423	**9220**
	9	11955	10056	10056	10056	10056	10056	10056	10056	**8106**
	11	10564	9439	9439	9439	9439	9439	9439	9439	**7408**
kroA200	7	14667	13285	13285	13286	13285	13286	13285	13285	**10783**
	9	12683	11708	11708	11710	11708	11708	11708	11708	**8994**
	11	12736	10748	10748	10760	10748	10748	10748	10748	**8432**

Table-3 Continued.

Instance	NC	Best found tour Cost						
		LS2 [7]	ILP [24]	Hybrid ACO [25]	Hybrid ABC [22]	Hybrid GA [22]	PVNS [31]	Proposed Algorithm
att532	3	52399	52412	51616	-	-	51457	**50684**
	5	42634	42387	42212	-	-	42148	**40950**
	7	38186	38016	37741	37608	37506	37621	**35928**
ali535	3	1370	1368	1367	1370	1369	1365	**1262**
	5	1184	1206	1185	1187	1182	1183	**1034**
	7	1094	1086	1083	1084	1079	1079	**900**
p654	3	25158	25155	25166	25186	25119	25181	**23537**
	5	23226	23211	23242	23285	23205	23224	**20669**
	7	22121	22126	22125	22121	22118	22135	**20444**

Table-4: Performance study with respect to the average cost and the standard deviation of the results in 5 runs.

		LS2[7]		ILP[24]		Hybrid ACO[25]		Hybrid GA[22]		Hybrid ABC[22]		PVNS[31]		Proposed Algorithm	
berlin52	7	3887	0.0	3887	0.0	3887.4	0.01	3887	0.0	3887	0.0	3887	0.0	**1459**	0.0
	9	3430	0.0	3430	0.0	3430	0.0	3430	0.0	3430	0.0	3430	0.0	**1112**	0.0
	11	3262	0.0	3262	0.0	3262	0.0	3262	0.0	3262	0.0	3262	0.0	**1029**	0.0
st70	7	288	0.0	288	0.0	288	0.0	288	0.0	288	0.0	288	0.0	**176**	0.0
	9	259	0.0	259	0.0	259	0.0	259	0.0	259	0.0	259	0.0	**140**	0.0
	11	247	0.0	247	0.0	247	0.0	247	0.0	247	0.0	247	0.0	**108**	0.0
eil76	7	207	0.0	207	0.0	207	0.0	207	0.0	207	0.0	211	0.0	**118**	0.0
	9	186	0.54	185	0.0	186	0.54	185	0.0	186	0.54	185	0.0	**90**	0.0
	11	170	0.0	170	0.0	170	0.0	170	0.0	170	0.0	170	0.0	**61**	0.0
rat99	7	486	0.0	486	0.0	486	0.0	486	0.0	486	0.0	486	0.0	**382**	0.0
	9	455	0.0	455	0.0	455	0.0	455	0.0	455	0.0	455	0.0	**201**	0.0
	11	444	0.0	444	0.0	444	0.0	444	0.0	444	0.0	444	0.0	**269**	0.0
kroA100	7	9674	0.0	9674	0.0	9674	0.0	9674	0.0	9674	0.0	9674	0.0	**7363.2**	0.6
	9	9159	0.0	9159	0.0	9159	0.0	9159	0.0	9159	0.0	9159	0.0	**6236.2**	0.5
	11	8901	0.0	8901	0.0	8901	0.0	8901	0.0	8901	0.0	8901	0.0	**5651.2**	0.5
kroB100	7	9537	0.0	9537	0.0	9537	0.0	9537	0.0	9537	0.0	9537	0.0	**7538.2**	0.7
	9	9240	0.0	9240	0.0	9240	0.0	9240	0.0	9240	0.0	9240	0.0	**6693.2**	0.6
	11	8842	0.0	8842	0.0	8842	0.0	8842	0.0	8842	0.0	8842	0.0	**6135.3**	0.9
kroC100	7	9723	0.0	9723	0.0	9724	0.01	9723	0.0	9723	0.0	9723	0.0	**7272.2**	0.8
	9	9171	0.0	9171	0.0	9171	0.0	9171	0.0	9171	0.0	9171	0.0	**6349.2**	0.6
	11	8632	0.0	8632	0.0	8632	0.0	8632	0.0	8632	0.0	8632	0.0	**5821.3**	0.4
kroE100	7	10150	0.0	10150	0.0	10150	0.0	10150	0.0	10150	0.0	10150	0.0	**7407.2**	0.5
	9	8991	0.0	8991	0.0	8991	0.0	8991	0.0	8992	0.0	8991	0.0	**6709.3**	0.8
	11	8450	0.0	8450	0.0	8450	0.0	8450	0.0	8450	0.0	8450	0.0	**6287.2**	0.5
kroA150	7	11800	3.30	11423.00	0.0	11423	0.0	11423	0.0	11423	0.0	11423	0.0	**9220.3**	0.4
	9	10062.4	0.06	10057.6	0.02	10056	0.0	10056	0.0	10057	0.02	10056	0.0	**8106.2**	0.5
	11	9439	0.0	9439	0.0	9439	0.0	9439	0.0	9439	0.0	9439	0.0	**7408.5**	0.9
kroA200	7	13666.4	2.87	13327	0.32	13286	0.01	13285	0.0	13286	0.01	13286	0.01	**10783.3**	0.6
	9	11716.8	0.08	11731.6	0.20	11710	0.02	11708	0.0	11708	0.0	11710	0.02	**8994.3**	0.6
	11	10848.6	1.94	10865.6	1.09	10764.2	0.15	10748	0.0	10748	0.0	10761.2	0.12	**8432.2**	0.5
ali535	3	1387.0	1.46	1381.4	1.05	1370.0	0.22	1375.8	0.64	1384.0	1.24	1370	0.37	**712.75**	2.83
	5	1201.2	1.62	1210.2	2.39	1188.4	0.54	1190.6	0.73	1189.8	0.66	1188.4	0.46	**792.61**	2.17
	7	1103.6	2.09	1093.4	1.15	1084.2.2	0.30	1088.4	0.68	1082.8	0.17	1084.2	0.48	**664.46**	1.63
p654	3	25206.6	.32	25206.0	0.31	25182.8	0.22	25224.2	0.39	25133.4	0.03	25219.4	0.26	**19921.75**	2.54
	5	23258.4	0.23	23224.8	0.09	23289.4	0.36	23291	0.37	23215.6	0.05	23278.4	0.29	**20250.625**	2.24
	7	22233.4	0.52	22138.6	.09	22130.8	0.06	22125.2	0.3	22119.6	0.01	22141.2	0.07	**19381.75**	2.54

Table-5: The computational results due to different coverage range(r)

Instance	NC		Best tour cost found for different NB and coverage range r														
berlin52	7	NB	8			9			10			11			12		
		r	500	600	700	500	600	700	500	600	700	500	600	700	500	600	700
		result	2607	2198	1844	2782	2115	1913	3089	2867	2518	3166	2342	2102	3368	2698	2382
	9	NB	8			9			10			11			12		
		r	500	600	700	500	600	700	500	600	700	500	600	700	500	600	700
		result	2560	2040	2008	2692	2328	1985	3089	2867	2518	2977	2445	2129	3152	2636	2426
	11	NB	7			8			9			10			11		
		r	500	600	700	500	600	700	500	600	700	500	600	700	500	600	700
		result	2547	2094	1808	2597	2040	1993	2692	2316	1985	3089	2867	2518	2977	2474	2089
st70	7	NB	10			11			12			13			14		
		r	50	90	100	50	90	100	50	90	100	50	90	100	50	90	100
		result	205	185	182	211	181	198	216	204	206	230	224	223	238	235	235
	9	NB	8			9			10			11			12		
		r	50	90	100	50	90	100	50	90	100	50	90	100	50	90	100
		result	169	143	142	187	157	167	205	185	182	202	193	190	214	205	202
	11	NB	7			8			9			10			11		
		r	50	90	100	50	90	100	50	90	100	50	90	100	50	90	100
		result	162	131	123	153	143	150	178	161	157	202	185	182	202	193	190
eil76	7	NB	11			12			13			14			15		
		r	30	40	50	30	40	50	30	40	50	30	40	50	30	40	50
		result	156	130	123	132	129	132	147	143	155	142	142	142	156	157	157
	9	NB	9			10			11			12			13		
		r	30	40	50	30	40	50	30	40	50	30	40	50	30	40	50
		result	117	93	90	122	102	98	122	120	118	132	132	128	145	134	134
	11	NB	7			8			9			10			11		
		r	30	40	50	30	40	50	30	40	50	30	40	50	30	40	50
		result	117	88	68	112	83	78	107	97	85	118	102	93	126	115	111
kroA100	7	r	1000	1100	1200	1000	1100	1200	1000	1100	1200	1000	1100	1200	1000	1100	1200
		result	8054	8202	8056	8840	8766	8639	9408	8917	8917	9339	9189	8903	10594	10131	9532
	9	NB	12			13			14			15			16		
		r	1000	1100	1200	1000	1100	1200	1000	1100	1200	1000	1100	1200	1000	1100	1200
		result	8859	7903	7748	8289	8174	7857	8401	8165	8077	8130	8018	7966	8241	8349	8201
	11	NB	10			11			12			13			14		
		r	1000	1100	1200	1000	1100	1200	1000	1100	1200	1000	1100	1200	1000	1100	1200
		result	7827	7552	7552	7785	7646	7170	7814	7801	7631	8220	7753	7770	8401	8165	8077
kroC100	7	NB	16			17			18			19			20		
		r	1000	1100	1200	1000	1100	1200	1000	1100	1200	1000	1100	1200	1000	1100	1200
		result	8285	8090	8090	8647	8671	8344	8690	8690	8250	9079	8771	8771	9585	9175	9071
	9	NB	12			13			14			15			16		
		r	1000	1100	1200	1000	1100	1200	1000	1100	1200	1000	1100	1200	1000	1100	1200
		result	7415	7510	7415	7794	7747	7173	7954	7927	7801	7947	7553	7553	8360	8050	7846
	11	NB	10			11			12			13			14		
		r	1000	1100	1200	1000	1100	1200	1000	1100	1200	1000	1100	1200	1000	1100	1200
		result	7138	6937	6890	7110	7034	6917	7494	7319	7419	7822	7473	7126	7954	7927	7573
kroD100	7	NB	16			17			18			19			20		
		r	1000	1100	1200	1000	1100	1200	1000	1100	1200	1000	1100	1200	1000	1100	1200
		result	8422	8341	8169		9130	8431	9717	9083	8828	9426	8559	8959	9754	9346	9251
	9	NB	12			13			14			15			16		
		r	1000	1100	1200	1000	1100	1200	1000	1100	1200	1000	1100	1200	1000	1100	1200
		result	8263	7784	7093	7919	7911	7441	8018	8033	7783	8587	7917	7817	8202	8202	8202
	11	NB	10			11			12			13			14		
		r	1000	1100	1200	1000	1100	1200	1000	1100	1200	1000	1100	1200	1000	1100	1200
		result	7410	6906	6645	7346	6776	6776	8015	7779	7298	7437	7335	7335	8018	8018	7783
kroE100	7	r	1000	1100	1200	1000	1100	1200	1000	1100	1200	1000	1100	1200	1000	1100	1200
		result	8436	8330	7860	8875	8877	8525	9152	8985	8769	9157	8775	9589	9580	9422	9422
	9	NB	12			13			14			15			16		
		r	1000	1100	1200	1000	1100	1200	1000	1100	1200	1000	1100	1200	1000	1100	1200
		result	8263	7784	7093	7919	7911	7441	8018	8033	7783	8587	7917	7817	8202	8202	8202
	11	NB	10			11			12			13			14		
		r	1000	1100	1200	1000	1100	1200	1000	1100	1200	1000	1100	1200	1000	1100	1200
		result	7040	6951	6752	7409	7200	6911	7551	7513	7513	7920	7370	7220	8283	7651	7651

Table-5 Continued.

Instance	NC	Best tour cost found for different NB and coverage range r															
kroA150	7	NB	23			24			25			26			27		
		r	1000	1100	1200	1000	1100	1200	1000	1100	1200	1000	1100	1200	1000	1100	1200
		result	10215	10215	10178	10657	10438	10437	10893	10484	10484	10913	10592	10385	8871	8857	8755
	9	NB	18			19			20			21			22		
		r	1000	1100	1200	1000	1100	1200	1000	1100	1200	1000	1100	1200	1000	1100	1200
		result	9288	9013	8766	9850	9514	9163	9775	9312	9054	9943	9802	9643	10378	9282	9282
	11	NB	14			15			16			17			18		
		r	1000	1100	1200	1000	1100	1200	1000	1100	1200	1000	1100	1200	1000	1100	1200
		result	9284	8754	8592	8412	8207	7997	8840	8448	8396	8570	8570	8570	8871	8755	8755
kroB150	7	NB	22			23			24			25			26		
		r	1000	1100	1200	1000	1100	1200	1000	1100	1200	1000	1100	1200	1000	1100	1200
		result	9776	9631	9631	9661	9661	9661	9781	9710	9710	10381	10347	10318	10361	10195	9992
	9	NB	18			19			20			21			22		
		r	1000	1100	1200	1000	1100	1200	1000	1100	1200	1000	1100	1200	1000	1100	1200
		result	9019	8891	8203	9830	9218	9146	9259	9143	9143	9499	9387	9387	9528	9304	9132
	11	NB	15			16			17			18			19		
		r	1000	1100	1200	1000	1100	1200	1000	1100	1200	1000	1100	1200	1000	1100	1200
		result	8577	8335	8217	8414	8400	8345	8768	8547	8199	8975	8660	8628	9204	8760	8685
kroA200	7	NB	29			30			31			32			33		
		r	1000	1100	1200	1000	1100	1200	1000	1100	1200	1000	1100	1200	1000	1100	1200
		result	10335	10335	10223	11630	11314	11198	11341	11203	11133	11103	11103	11103	11868	11548	11480
	9	NB	23			24			25			26			27		
		r	1000	1100	1200	1000	1100	1200	1000	1100	1200	1000	1100	1200	1000	1100	1200
		result	10804	10407	10064	10104	9390	9390	10216	9655	10188	10387	9814	9773	10074	10074	10074
	11	NB	19			20			21			22			23		
		r	1000	1100	1200	1000	1100	1200	1000	1100	1200	1000	1100	1200	1000	1100	1200
		result	9253	8783	8507	9311	8988	8988	9413	9233	9039	9581	9581	9268	9396	9396	9288
ali535	7	NB	77			78			79			80			81		
		r	150	160	170	150	160	170	150	160	170	150	160	170	150	160	170
		result	907	743	743	817	806	806	875	805	780	781	781	735	843	814	814
	9	NB	60			61			62			63			64		
		r	150	160	170	150	160	170	150	160	170	150	160	170	150	160	170
		result	733	733	704	746	705	660	796	736	727	736	736	726	734	734	729
	11	NB	50			51			52			53			54		
		r	150	160	170	150	160	170	130	140	150	100	130	140	150	160	170
		result	648	615	615	663	652	652	681	681	681	739	720	710	635	617	575
u574	3	NB	192			193			194			195			196		
		r	1500	1600	1700	1500	1600	1700	1400	1500	1700	1400	1500	1600	1500	1600	1800
		result	22583	22514	22123	22250	22250	22250	22600	22600	22573	22711	21977	21977	22977	22568	22293
	5	NB	115			116			117			118			119		
		r	1400	1600	1800	1500	1700	1800	1500	1600	1700	1300	1400	1500	1300	1400	1500
		result	18077	17750	17631	17237	17237	17081	16655	16655	16655	17933	17304	17304	18101	17182	17182
	7	NB	82			83			84			85			86		
		r	1400	1500	1600	1500	1600	1700	1400	1500	1600	1400	1500	1600	1200	1400	1600
		result	14899	14899	14773	14385	13340	14340	14938	14712	14712	14840	14629	14629	15329	14792	14774

Table-6: The computational results due to different coverage range(r)

Instance	NC		Best tour cost found for different NB														
berlin52	7	NB	8 ($r=982$)					9 ($r=1186$)					10 ($r=1124$)				
		size	50%	60%	70%	80%	100%	50%	60%	70%	80%	100%	50%	60%	70%	80%	100%
		result	2581	2400	1829	1734	1459	2196	1930	1930	1575	1540	2895	2552	2513	2244	2157
		NB	11 ($r=1125$)					12 ($r=1192$)					13 ($r=1315$)				
		size	50%	60%	70%	80%	100%	50%	60%	70%	80%	100%	50%	60%	70%	80%	100%
		result	2351	2087	2087	2050	1639	2983	2983	2663	2609	2361	2983	2983	2663	2609	2607
	9	NB	6 ($r=960$)					7 ($r=1082$)					8 ($r=982$)				
		size	50%	60%	70%	80%	100%	50%	60%	70%	80%	100%	50%	60%	70%	80%	100%
		result	-	-	1742	1649	1112	-	2039	1866	1516	1180	2576	2088	1996	1943	1455
		NB	9 ($r=1125$)					10 ($r=1124$)					11 ($r=1125$)				
		size	50%	60%	70%	80%	100%	50%	60%	70%	80%	100%	50%	60%	70%	80%	100%
		result	2345	2031	1933	1922	1528	2895	552	2313	2244	2157	2328	2115	2074	2074	1647
	11	NB	5 ($r=780$)					6 ($r=955$)					7 ($r=1082$)				
		size	50%	60%	70%	80%	100%	50%	60%	70%	80%	100%	50%	60%	70%	80%	100%
		result	-	-	-	1929	1029	-	1703	1644	1443	953	2127	2037	1813	1487	1122
		NB	8 ($r=982$)					9 ($r=955$)					10 ($r=1124$)				
		size	50%	60%	70%	80%	100%	50%	60%	70%	80%	100%	50%	60%	70%	80%	100%
		result	2576	2088	1996	1949	1455	2675	2296	2017	1910	1547	2895	2552	2313	2244	2157
st70	7	NB	10 ($r=80$)					11 ($r=70$)					12 ($r=52$)				
		size	50%	60%	70%	80%	100%	50%	60%	70%	80%	100%	50%	60%	70%	80%	100%
		result	231	203	201	201	172	231	214	200	200	185	-	271	249	233	205
		NB	13 ($r=56$)					14 ($r=54$)					15 ($r=62$)				
		size	50%	60%	70%	80%	100%	50%	60%	70%	80%	100%	50%	60%	70%	80%	100%
		result	267	248	236	231	212	-	278	265	257	235	271	266	248	237	237
	9	NB	8 ($r=72$)					9 ($r=60$)					10 ($r=82$)				
		size	50%	60%	70%	80%	100%	50%	60%	70%	80%	100%	50%	60%	70%	80%	100%
		result	221	176	161	147	140	-	222	213	162	158	231	203	201	201	176
		NB	11 ($r=72$)					12 ($r=50$)					13 ($r=56$)				
		size	50%	60%	70%	80%	100%	50%	60%	70%	80%	100%	50%	60%	70%	80%	100%
		result	233	198	200	200	181	-	265	233	222	204	267	246	236	229	222
	11	NB	7 ($r=84$)					8 ($r=51$)					9 ($r=52$)				
		size	50%	60%	70%	80%	100%	50%	60%	70%	80%	100%	50%	60%	70%	80%	100%
		result	174	158	135	135	108	-	236	218	202	131	-	227	206	184	164
		NB	10 ($r=82$)					11 ($r=72$)					12 ($r=52$)				
		size	50%	60%	70%	80%	100%	50%	60%	70%	80%	100%	50%	60%	70%	80%	100%
		result	-	231	203	201	176	233	198	204	201	181	-	247	225	225	201
eil76	7	NB	11 ($r=60$)					12 ($r=33$)					13 ($r=35$)				
		size	50%	60%	70%	80%	100%	50%	60%	70%	80%	100%	50%	60%	70%	80%	100%
		result	158	150	132	115118	-	175	161	138	130	-	-	161	155	139	139
		NB	14 ($r=45$)					15 ($r=35$)					16 ($r=27$)				
		size	50%	60%	70%	80%	100%	50%	60%	70%	80%	100%	50%	60%	70%	80%	100%
		result	179	154	148	147	148	-	205	182	167	152	-	-	220	164	155
	9	NB	9 ($r=35$)					10 ($r=45$)					11 ($r=45$)				
		size	50%	60%	70%	80%	100%	50%	60%	70%	80%	100%	50%	60%	70%	80%	100%
		result	-	-	135	124	90	-	-	153	133	93	152	134	128	120	115
		NB	12 ($r=50$)					13 ($r=60$)					14 ($r=35$)				
		size	50%	60%	70%	80%	100%	50%	60%	70%	80%	100%	50%	60%	70%	80%	100%
		result	139	132	131	121	125	149	137	137	131	131	-	178	170	156	152
	11	NB	7 ($r=45$)					8 ($r=40$)					9 ($r=45$)				
		size	50%	60%	70%	80%	100%	50%	60%	70%	80%	100%	50%	60%	70%	80%	100%
		result	-	-	111	104	62	-	134	119	110	61	-	128	100	93	89
		NB	10 ($r=42$)					11 ($r=46$)					12 ($r=50$)				
		size	50%	60%	70%	80%	100%	50%	60%	70%	80%	100%	50%	60%	70%	80%	100%
		result	150	130	122	103	103	142	127	124	119	109	139	132	132	126	125
kroA100	7	NB	15 ($r=3408$)					16 ($r=947$)					17 ($r=2852$)				
		size	50%	60%	70%	80%	100%	50%	60%	70%	80%	100%	50%	60%	70%	80%	100%
		result	-	8328	8081	8081	7365	-	-	10129	9735	7857	8805	8277	8059	7743	7743
		NB	18 ($r=2160$)					19 ($r=2352$)					20 ($r=1520$)				
		size	50%	60%	70%	80%	100%	50%	60%	70%	80%	100%	50%	60%	70%	80%	100%
		result	9309	8982	8734	8660	8347	9262	8788	8788	8806	8774	-	-	10072	9851	9349
	9	NB	12 ($r=1976$)					13 ($r=2569$)					14 ($r=2560$)				
		size	50%	60%	70%	80%	100%	50%	60%	70%	80%	100%	50%	60%	70%	80%	100%
		result	8804	7618	7161	6987	6641	7716	7229	6382	6236	6236	8226	8195	7744	7698	7616
		NB	15 ($r=1449$)					16 ($r=2426$)					17 ($r=2267$)				
		size	50%	60%	70%	80%	100%	50%	60%	70%	80%	100%	50%	60%	70%	80%	100%
		result	-	8859	8183	7835	7804	-	-	9935	9387	8150	8682	8231	8231	8223	7772
	11	NB	10 ($r=2995$)					11 ($r=2445$)					12 ($r=2052$)				
		size	50%	60%	70%	80%	100%	50%	60%	70%	80%	100%	50%	60%	70%	80%	100%
		result	6508	6156	6156	6244	5651	7361	7193	6931	6931	6485	7736	7630	7128	7128	6460
		NB	13 ($r=2641$)					14 ($r=2560$)					15 ($r=1835$)				
		size	50%	60%	70%	80%	100%	50%	60%	70%	80%	100%	50%	60%	70%	80%	100%
		result	7759	7431	7182	7182	6293	8226	8195	7744	7698	7616	8760	7842	7842	7675	7675

Table-6 continued

Instance	NC	Best tour cost found for different NB														
kroB100	7	NB	15 ($r = 2233$)					16 ($r = 2423$)					17 ($r = 2532$)				
		size	50%	60%	70%	80%	100%	50%	60%	70%	80%	100%	50%	60%	70%	80%	100%
		result	8664	8343	8343	8051	7538	8534	8534	8585	8365	8341	9158	8227	8177	7822	8180
		NB	18 ($r = 2216$)					19 ($r = 1244$)					20 ($r = 1519$)				
		size	50%	60%	70%	80%	100%	50%	60%	70%	80%	100%	50%	60%	70%	80%	100%
		result	9003	8822	8773	8659	8659	-	11177	9898	9227	9227	10846	10014	9684	9544	9147
	9	NB	12 ($r = 1191$)					13 ($r = 1468$)					14 ($r = 1842$)				
		size	50%	60%	70%	80%	100%	50%	60%	70%	80%	100%	50%	60%	70%	80%	100%
		result	7004	7009	6839	6729	6729	-	9968	8203	7423	6913	8832	7927	8066	7614	7597
		NB	15 ($r = 2440$)					16 ($r = 1961$)					17 ($r = 2457$)				
		size	50%	60%	70%	80%	100%	50%	60%	70%	80%	100%	50%	60%	70%	80%	100%
		result	8308	8106	8073	7966	7966	9035	8621	8459	8284	8284	8572	8168	7807	7807	7366
	11	NB	10 ($r = 2212$)					11 ($r = 1747$)					12 ($r = 2281$)				
		size	50%	60%	70%	80%	100%	50%	60%	70%	80%	100%	50%	60%	70%	80%	100%
		result	6642	6194	6251	6247	6135	8108	7015	6972	6872	6349	6795	6849	6361	6361	6361
		NB	13 ($r = 2556$)					14 ($r = 1842$)					15 ($r = 2192$)				
		size	50%	60%	70%	80%	100%	50%	60%	70%	80%	100%	50%	60%	70%	80%	100%
		result	7322	7215	7175	7167	7167	8832	7927	8066	7614	7597	8214	7847	7847	7797	7366
kroC100	7	NB	15 ($r = 2835$)					16 ($r = 2003$)					17 ($r = 2374$)				
		size	50%	60%	70%	80%	100%	50%	60%	70%	80%	100%	50%	60%	70%	80%	100%
		result	8137	7704	7428	7382	7272	8399	7854	8060	7952	7543	8674	8513	8384	8279	7864
		NB	18 ($r = 2359$)					19 ($r = 1635$)					20 ($r = 1757$)				
		size	50%	60%	70%	80%	100%	50%	60%	70%	80%	100%	50%	60%	70%	80%	100%
		result	8484	8443	8443	8242	8137	9912	9332	8708	8708	8664	10351	9891	9486	9342	8795
	9	NB	12 ($r = 2535$)					13 ($r = 1478$)					14 ($r = 1902$)				
		size	50%	60%	70%	80%	100%	50%	60%	70%	80%	100%	50%	60%	70%	80%	100%
		result	7545	7205	7155	6599	6349	-	8195	7921	7384	6825	8219	7819	7146	7325	7317
		NB	15 ($r = 1802$)					16 ($r = 1316$)					17 ($r = 3027$)				
		size	50%	60%	70%	80%	100%	50%	60%	70%	80%	100%	50%	60%	70%	80%	100%
		result	8372	7952	7617	7501	7206	-	9343	8607	8342	7612	8058	7981	7708	7708	7708
	11	NB	10 ($r = 1907$)					11 ($r = 2417$)					12 ($r = 2662$)				
		size	50%	60%	70%	80%	100%	50%	60%	70%	80%	100%	50%	60%	70%	80%	100%
		result	9095	8267	7934	7219	5851	-	-	8621	8316	6422	-	9027	8969	8626	6515
		NB	13 ($r = 1246$)					14 ($r = 2998$)					15 ($r = 3440$)				
		size	50%	60%	70%	80%	100%	50%	60%	70%	80%	100%	50%	60%	70%	80%	100%
		result	-	-	-	10377	6705	-	-	9867	9867	7884	7399	7392	7392	7392	7366
kroD100	7	NB	15 ($r = 2508$)					16 ($r = 1414$)					17 ($r = 1700$)				
		size	50%	60%	70%	80%	100%	50%	60%	70%	80%	100%	50%	60%	70%	80%	100%
		result	9095	8267	7934	7219	6866	-	-	8621	8316	8032	-	9027	8969	8626	8030
		NB	18 ($r = 1200$)					19 ($r = 1900$)					20 ($r = 1600$)				
		size	50%	60%	70%	80%	100%	50%	60%	70%	80%	100%	50%	60%	70%	80%	100%
		result	-	-	-	10377	8647	-	-	9867	9867	9084	10675	9671	9355	9355	9221
	9	NB	12 ($r = 1665$)					13 ($r = 2140$)					14 ($r = 1590$)				
		size	50%	60%	70%	80%	100%	50%	60%	70%	80%	100%	50%	60%	70%	80%	100%
		result	-	8415	7376	6474	6474	7805	7386	7070	7011	6846	9585	8211	7876	7876	7778
		NB	15 ($r = 1600$)					16 ($r = 1282$)					17 ($r = 1601$)				
		size	50%	60%	70%	80%	100%	50%	60%	70%	80%	100%	50%	60%	70%	80%	100%
		result	-	9104	7756	7593	7176-	9712	9004	8594	7538	-	9587	8506	8506	7929	7924
	11	NB	10 ($r = 1960$)					11 ($r = 1950$)					12 ($r = 2154$)				
		size	50%	60%	70%	80%	100%	50%	60%	70%	80%	100%	50%	60%	70%	80%	100%
		result	7478	6761	6145	5686	5525	7245	6754	6561	6076	6076	7681	7004	6857	6465	6026
		NB	13 ($r = 1592$)					14 ($r = 1592$)					15 ($r = 1653$)				
		size	50%	60%	70%	80%	100%	50%	60%	70%	80%	100%	50%	60%	70%	80%	100%
		result	-	7629	7404	7199	6890	9585	8211	7876	7876	7517	954	8701	8165	7279	7237
kroA150	7	NB	22 ($r = 3400$)					23 ($r = 3500$)					24 ($r = 1900$)				
		size	50%	60%	70%	80%	100%	50%	60%	70%	80%	100%	50%	60%	70%	80%	100%
		result	10181	9762	9762	9546	9220	9931	10282	9567	9542	542	10673	10788	10470	10611	10226
		NB	25 ($r = 3660$)					26 ($r = 2890$)					27 ($r = 1186$)				
		size	50%	60%	70%	80%	100%	50%	60%	70%	80%	100%	50%	60%	70%	80%	100%
		result	10427	10427	10395	10395	10264	10900	10953	10386	10314	10314	13393	12257	12190	11424	10951
	9	NB	17 ($r = 3200$)					23 ($r = 2300$)					24 ($r = 2770$)				
		size	50%	60%	70%	80%	100%	50%	60%	70%	80%	100%	50%	60%	70%	80%	100%
		result	8487	8487	8487	8033	8106	10303	10214	9850	9935	9339	10501	10655	10092	10172	10069
		NB	25 ($r = 3660$)					26 ($r = 2710$)					27 ($r = 1870$)				
		size	50%	60%	70%	80%	100%	50%	60%	70%	80%	100%	50%	60%	70%	80%	100%
		result	10427	10800	10395	10366	10264	10737	10783	10249	10249	10249	10737	10737	10249	10249	10249
	11	NB	15 ($r = 2090$)					18 ($r = 1620$)					19 ($r = 1240$)				
		size	50%	60%	70%	80%	100%	50%	60%	70%	80%	100%	50%	60%	70%	80%	100%
		result	8366	7562	7562	7805	7408	10181	9149	8822	8624	7967	-	11735	9918	9535	8671
		NB	20 ($r = 2050$)					21 ($r = 2870$)					22 ($r = 1820$)				
		size	50%	60%	70%	80%	100%	50%	60%	70%	80%	100%	50%	60%	70%	80%	100%
		result	9609	8955	9282	9245	8980	967	9395	9454	9547	9547	9670	9571	8845	8845	8845

Table-7: Results under the restriction of group coverage cost NC(r/3)

Instance	NC	Result						
berlin52	**7**	NB	**8**($r = 982$)	9($r = 1186$)	10($r = 1124$)	11($r = 1125$)	12($r = 1192$)	13($r = 1315$)
		Best cost	**1790**	1919	2161	2112	2371	2529
	9	NB	6($r = 960$)	7($r = 1082$)	**8**($r = 982$)	9($r = 1125$)	10($r = 1124$)	11($r = 1125$)
		Best cost	1546	1487	**1455**	1929	2181	2070
	11	NB	5($r = 780$)	6($r = 955$)	7($r = 1082$)	**8**($r = 982$)	9($r = 955$)	10($r = 1124$)
		Best cost	1923	1559	1487	**1463**	1839	2168
st70	**7**	NB	**11**($r = 82$)	12($r = 72$)	13($r = 52$)	14($r = 56$)	15($r = 54$)	16($r = 62$)
		Best cost	**124**	155	155	149	172	198
	9	NB	9($r = 72$)	**10**($r = 57$)	11($r = 82$)	12($r = 72$)	13($r = 49$)	14($r = 56$)
		Best cost	141	**118**	123	133	124	154
	11	NB	7($r = 84$)	8($r = 51$)	**9**($r = 52$)	10($r = 82$)	11($r = 72$)	12($r = 52$)
		Best cost	111	105	**95**	107	121	122
eil76	**7**	NB	10($r = 60$)	**11**($r = 33$)	12($r = 34$)	13($r = 43$)	14($r = 33$)	15($r = 27$)
		Best cost	208	**202**	227	235	240	252
	9	NB	**8**($r = 34$)	9($r = 45$)	10($r = 43$)	11($r = 50$)	12($r = 58$)	13($r = 32$)
		Best cost	**162**	203	191	188	226	235
	11	NB	**7**($r = 43$)	8($r = 39$)	9($r = 43$)	10($r = 42$)	11($r = 46$)	12($r = 50$)
		Best cost	**136**	178	197	191	186	224
kroA100	**7**	NB	**15**($r = 1875$)	16($r = 947$)	17($r = 2852$)	18($r = 2160$)	19($r = 2352$)	20($r = 1520$)
		Best cost	**8191**	8191	8261	8664	8880	9448
	9	NB	**12**($r = 1976$)	13($r = 2569$)	14($r = 2560$)	15($r = 1449$)	16($r = 2426$)	17($r = 2267$)
		Best cost	**7170**	7333	7759	8565	8032	8046
	11	NB	**10**($r = 2995$)	11($r = 2445$)	12($r = 2052$)	13($r = 2641$)	14($r = 2560$)	15($r = 1835$)
		Best cost	**6276**	6966	7360	7306	7375	7821
kroB100	**7**	NB	**15**($r = 2233$)	16($r = 2423$)	17($r = 2532$)	18($r = 2216$)	19($r = 1244$)	20($r = 1519$)
		Best cost	**8074**	8421	8159	8943	9511	9384
	9	NB	12($r = 1191$)	13($r = 1468$)	**14**($r = 1842$)	15($r = 2440$)	16($r = 1961$)	17($r = 2457$)
		Best cost	7919	8104	**7903**	8031	8413	8141
	11	NB	**10**($r = 2212$)	11($r = 1747$)	12($r = 2281$)	13($r = 2556$)	14($r = 1842$)	15($r = 2192$)
		Best cost	**6298**	6811	6745	7347	7494	7915
kroC100	**7**	NB	**15**($r = 2835$)	16($r = 2003$)	17($r = 2374$)	18($r = 2359$)	19($r = 1635$)	20($r = 1757$)
		Best cost	**7678**	8209	8212	8365	8886	8958
	9	NB	**12**($r = 2535$)	13($r = 1478$)	14($r = 1902$)	15($r = 1802$)	16($r = 1316$)	17($r = 3027$)
		Best cost	**6764**	7813	7270	7684	8012	7997
	11	NB	10($r = 1907$)	**11**($r = 2417$)	12($r = 2662$)	13($r = 1246$)	14($r = 2998$)	15($r = 3440$)
		Best cost	6771	**6489**	6786	7797	7325	7371
kroD100	**7**	NB	**15**($r = 2508$)	16($r = 1414$)	17($r = 1699$)	18($r = 1197$)	19($r = 1908$)	20($r = 1590$)
		Best cost	**7447**	8952	8988	9181	9181	9203
	9	NB	12($r = 1665$)	**13**($r = 2140$)	14($r = 1592$)	15($r = 1601$)	16($r = 1282$)	17($r = 1601$)
		Best cost	7751	**7140**	8028	7930	8579	8053
	11	NB	**10**($r = 1599$)	11($r = 1950$)	12($r = 2154$)	13($r = 1592$)	14($r = 1592$)	15($r = 1653$)
		Best cost	**6032**	6489	6683	6927	7803	7210
kroA150	**7**	NB	22($r = 3408$)	**23**($r = 3511$)	24($r = 1915$)	25($r = 3664$)	26($r = 2912$)	27($r = 2892$)
		Best cost	7645	**7252**		8051	9090	8880
	9	NB	**17**($r = 3202$)	23($r = 2302$)	24($r = 2773$)	25($r = 3664$)	26($r = 2719$)	27($r = 1876$)
		Best cost	**6577**	7774	6713	7548	8648	10926
	11	NB	15($r = 2091$)	18($r = 1627$)	19($r = 1240$)	20($r = 2056$)	21($r = 2871$)	22($r = 1826$)
		Best cost				8220	6438	8614

Table-8: Results of the CSPs with fuzzy cost matrices using two proposed approaches (same result)

Instance	NC	Best tour cost found for different NB					
berlin52	**7**	NB=**8**	9	10	11	12	13
		[1446.8,1449, 1450.4]	[1359.7,1542, 1543.5]	[215.4,2157, 2158.7]	[1636.6,1639, 1641.3]	[2276.3,2279, 2282.0]	[2603.8,2607, 2610.5]
	9	NB=**6**	7	8	9	10	11
		[958.5,960, 961.3]	[1178.8,1180, 1181.6]	[1480.9,1483, 1484.7]	[1534.8,1537, 1538.2]	[2154.4,2157, 2158.7]	[1642.4,1645, 1648.5]
	11	NB =**5**	6	7	8	9	10
		[742.1,743, 743.7]	[951.8,953, 954.9]	1105.5,1107, 1108.5]	[1480.9,1483, 1484.7]	[1540.8,1543, 1544.9]	[2154.4,2157, 2158.7]
st70	**7**	NB=**11**	12	13	14	15	16
		[181.2,186, 188.8]	[199.7,203, 205.2]	[215.0,219, 222.8]	[231.3,235, 239.3]	[237.3,241, 244.3]	[251.8,256, 260.4]
	9	NB=**8**	9	10	11	12	13
		[138.6,140, 141.0]	[152.3,155, 156.3]	[161.6,164, 165.8]	[173.4,176, 178.1]	[178.5,181, 184.2]	[189.3,193, 196.4]
	11	NB =**7**	8	9	10	11	12
		[105.4,108, 109.5]	[140.2,143, 144.9]	[161.6,164, 165.8]	[178.2,181, 184.2]	[108.7,111, 112.7]	[122.0,125, 127.2]
eil76	**7**	NB= **11**	12	13	14	15	16
		[113.2,116, 119.1]	[123.9,126, 128.5]	[135.0,139, 141.8]	[139.9,143, 146.5]	[148.3,152, 156.3]	[153.7,158, 161.5]
	9	NB=**9**	10	11	12	13	14
		[87.6,90, 92.1]	[99.7,102, 104.4]	[110.1,113, 115.3]	[122.0,125, 127.2]	[136.4,140, 142.7]	[147.4,151, 153.8]
	11	NB =7	**8**	9	10	11	12
		[60.6,62, 63.8]	**[59.2,61, 62.5]**	[90.1,92, 93.5]	[105.9,108, 109.7]	[108.7,111, 112.7]	[122.0,125, 127.2]
kroA100	**7**	NB= **15**	16	17	18	19	20
		[7054.3,7058, 7062.8]	[8034.8,8038, 8041.5]	[8019.2,8023, 8027.5]	[8121.5,8126, 8129.7]	[8923.4,8928, 8932.6]	[9350.3,9356, 9360.8]
	9	NB=**12**	13	14	15	16	17
		[6327.6,6331, 6333.7]	[6733.3,6736, 6738.3]	[7611.5,7615, 7618.9]	[7716.7,7721, 7724.5]	[8253.1,8257, 8260.8]	[7869.0,7873, 7877.1]
	11	NB =**10**	11	12	13	14	15
		[5662.9,5665, 5667.5]	[6346.9,6350, 6352.0]	[6468.7,6472, 6475.1]	[6885.9,6889, 6812.1]	[7611.5,7615, 7618.9]	[7835.2,7839, 7842.0]
kroB100	**7**	NB= **15**	16	17	18	19	20
		[7210.2,7214, 7217.0]	[8148.1,8152, 8156.3]	[8049.2,8055, 8059.6]	[8725.6,8730, 8735.2]	[8681.5,8686, 8690.6]	[9311.4,9315, 9318.5]
	9	NB=**12**	13	14	15	16	17
		[6523.9,6527, 6530.2]	[7136.1,7139, 7142.1]	[7411.1,7415, 7418.4]	[7669.8,7674, 7678.4]	[8266.5,8271, 8275.3]	[8007.7,8012, 8015.2]
	11	NB =**10**	11	12	13	14	15
		[5808.8,5812, 5814.7]	[6397.7,6400, 6402.4]	[6537.5,6541, 6543.4]	[6891.2,6895, 6899.2]	[7411.1,7415, 7418.4]	[7688.0,7691, 7695.4]
kroC100	**7**	NB= **15**	16	17	18	19	20
		[7335.8,7340, 7344.5]	[7811.9,7818, 7820.4]	[7961.8,7965, 7974.9]	[8160.1,8165, 8170.0]	[8804.9,8810, 8815.3]	[8850.7,8856, 8860.7]
	9	NB=**12**	13	14	15	16	17
		[6558.7,6562, 6564.2]	[6659.3,6663, 6666.4]	[7051.8,7055, 7058.6]	[6915.4,6919, 6922.5]	[7797.2,7801, 7805.0]	[7948.1,7952, 7956.7]
	11	NB =**10**	11	12	13	14	15
		[5659.6,5662, 5665.3]	[6379.7,6382, 6384.8]	[6492.3,6496, 6498.6]	[6819.6,6823, 6827.5]	[7051.8,7055, 7058.6]	[7129.5,7134, 7138.6]

Table-8:continued

Instance	NC	Best tour cost found for different NB					
kroD100	**7**	NB= 15	**16**	17	18	19	20
		[7397.9,7401, 7404.9]	**[7381.9,7386, 7390.5]**	[7678.2,7682, 7686.7]	[8110.5,8115, 8120.0]	[8935.8,8940, 8945.0]	[9406.0,9410, 9415.1]
	9	NB=**12**	13	14	15	16	17
		[6339.7,6343, 6346.6]	[6947.2,6950, 6953.8]	[7406.6,7410, 7413.1]	[6962.4,6965, 6968.7]	[7815.3,7819, 7822.5]	[7934.9,7939, 7941.9]
	11	NB =**10**	11	12	13	14	15
		[5612.3,5615, 5617.7]	[5792.4,5795, 5797.4]	[6301.7,6305, 6307.8]	[6811.8,6815, 6819.2]	[7267.0,7271, 7274.0]	[7399.3,7403, 7406.1]
kroE100	**7**	NB=**15**	16	17	18	19	20
		[7370.8,7375, 7378.4]	[8195.7,8199, 8202.5]	[8068.6,8072, 8075.8]	[8720.5,8726, 8730.4]	[8495.4,8501, 8506.2]	[9218.9,9224, 9229.2]
	9	NB=**12**	13	14	15	16	17
		[6673.1,6676, 6679.9]	[7233.7,7238, 7240.9]	[7551.2,7555, 7557.9]	[7605.7,7610, 7614.0]	[8121.8,8126, 8130.7]	[8094.9,8100, 8105.0]
	11	NB =**10**	11	12	13	14	15
		[6285.1,6287, 6289.4]	[6605.2,6608, 6611.0]	[6673.7,6677, 6680.7]	[6949.8,6953, 6955.9]	[7551.2,7555, 7557.9]	[7605.7,7610, 7614.0]
rd100	**7**	NB= **15**	16	17	18	19	20
		[2397.0,2401, 2403.1]	[2590.5,2593, 2597.8]	[2579.6,2583, 2586.9]	[2745.6,2751, 2755.3]	[2843.4,2848, 2852.4]	[2996.7,3003, 3008.0]
	9	NB=12	**13**	14	15	16	17
		[2222.3,2225, 2228.5]	**[2187.6,2190, 2193.0]**	[2384.1,2388, 2392.4]	[2442.6,2446, 2450.2]	[2581.1,2586, 2590.3]	[2379.0,2385, 2388.8]
	11	NB =**10**	11	12	13	14	15
		[1930.4,1933, 1935.8]	[1979.3,1982, 1984.4]	[2124.1,2127, 2130.0]	[2297.8,2302, 2305.5]	[2384.1,2388, 2392.4]	[2445.1,2448, 2452.6]
kroA150	**7**	NB=**22**	23	24	25	26	27
		[9178.8,9184, 9190.1]	[9719.5,9725, 9730.6]	[10212.8,10219, 10224.7]	[10497.3,10504, 10510.3]	[10292.4,10299, 10305.6]	[10974.4,10981, 10988.8]
	9	NB=**17**	18	19	20	21	22
		[7965.6,7971, 7975.0]	[8439.4,8444, 8447.9]	[8725.4,8730, 8735.4]	[8743.9,8749, 8755.0]	[9361.3,9367, 9372.7]	[9465.7,9471, 9476.8]
	11	NB =**14**	15	16	17	18	19
		[7313.9,7318, 7320.9]	[7729.0,7732, 7735.6]	[7520.4,7525, 7529.3]	[8084.0,8088, 8092.2]	[8614.7,8620, 8625.0]	[8691.4,8696, 8701.1]
kroA200	**7**	NB=**29**	30	31	32	33	34
		[10678.6,10686, 10692.6]	[11095.3,11102, 11109.9]	[11240.2,11247, 11254.5]	[11400.6,11408, 11414.8]	[11608.1,11616, 11624.3]	[11755.1,11765, 11774.4]
	9	NB=**23**	24	25	26	27	28
		[9245.4,9252, 9258.2]	[9775.0,9720, 9726.5]	[9810.7,9818, 9824]	[10340.3,10346, 10352.3]	[10249,10256, 10262.3]	[10505.4,10512, 10520.1]
	11	NB =**19**	20	21	22	23	24
		[8509.9,8515, 8519.2]	[8669.7,8676, 8681.3]	[8928.5,8934, 8937.2]	[9381.8,9388, 9392.6]	[9535.5,9542, 9547]	[9545.5,9552, 9557]

Table-9: Results of the CSPs with rough cost matrices using two proposed approaches (same result)

Instance	NC	Best tour cost found for different NB					
berlin52	**7**	NB= **8**	9	10	11	12	13
		[1446,1462.5, 1422,1482.4]	[1547,1559.8, 1516.3,1585.5]	[2116,2141.6, 2090,2164.6]	[1639,1661.6, 1604.4,1694.4]	[2344,2370.4, 2316.1,2400.1]	[2709,2742.9, 2671.9,2778.2]
	9	NB=**6**	7	8	9	10	11
		[909,923.6, 889.3,934.4]	[1181,1196.8, 1165.2,1210.2]	[1483,1505.5, 1456.8,1524.5]	[1537,1560.3, 1509.7,1586.3]	[2116,2141.6, 2090.2164.6]	[1645,1665.6, 1615.9,1697]
	11	NB =**5**	6	7	8	9	10
		[743,751.8, 729.9,764.8]	[915,927.5, 901.5,938.8]	1194,1207.4, 1178.9,1224]	[1483,1505.5, 1456.8,1524]	[1549,1561.3, 1531.9,1579.7]	[2116,2141.6, 2090.2,2164.6]
st70	**7**	NB= **10**	11	12	13	14	15
		[162,178.4, 143.6,201.7]	[197,221, 173.5,253.4]	[202,216.4, 178.6,247.5]	[221,250.1, 192.7,286.3]	[234,263.6, 191.8,287]	[244,276, 200.5,306.9]
	9	NB=**8**	9	10	11	12	13
		[138,157.1 ,125.9,174.5]	[168,189, 144.1,208.2]	[162,178.4, 143.6,201.7]	[200,222.2 ,173.8,248.3]	[206,224.9, 175.9,246.8]	[222,252.6, 192.4,284.5]
	11	NB =**7**	8	9	10	11	12
		[126,139.9, 119.1,152.2]	[144,159.4, 129.3,174.2]	161,183.1, 140.9,206.3]	[162,178.4, 143.6,201.7]	[200,222.2, 173.8,248.3]	[210,234.4, 197.5,263.5]
eil76	**7**	NB=**11**	12	13	14	15	16
		[119,145.6, 82.4,172.6]]	[125,149.8, 96,173.4]	[142,174, 111.3,204.4]	[133,170.4, 92,210.3]	[152,188.5, 114.2,221.2]	[162,195.5, 120.6,237.5]
	9	NB=**9**	10	11	12	13	14
		[87,97.7, 69.3,130.1]	[98,119.4, 71.3,139.6]	[120,135.5, 92.9,165.5]	[125,144.4, 97,172.6]	[136,163.3, 104.1,199.5]	[140,170.5, 100,200.4]
	11	NB =**7**	8	9	10	11	12
		[67,81.3, 48.5,103.9]	[78,94, 58.3,119.6]	96,115.5, 68.9,136.7]	[100,127.6, 76.2,157.3]	[117,139.4, 86,171.1]	[125,144.4, 97,172.6]
rat99	**7**	NB= **15**	16	17	18	19	20
		[385,406.7, 354.5,445.3]	[403,431.2, 364.1,459.7]	[406,452.5, 367.1,484.8]	[426,467, 376.5,518]	[441,479.2, 387.1,529.9]	[450,504.6, 384.5,564.8]
	9	NB=**11**	12	13	14	15	16
		[304,327.8, 279.3,351.1]	[325,352.7, 294.5,382.5]	[349,380.5, 320.8,402.1]	[372,403.2, 347.5,431.2]	[370,404.9, 333.9,441.3]	[385,425.3, 346.9,465.2]
	11	NB =**9**	10	11	12	13	14
		[260,276.9, 234.7,290.8]	[269,293.7, 250.1,315]	[304,327.8, 279.3,351.1]	[330,352.5, 306.1,378.2]	[346,379, 316.4,409.8]	[372,403.0, 347.3,431.2]
rd100	**7**	NB= **15**	16	17	18	19	20
		[2401,2434.2, 2364.3,2473.9]	[2599,2636.5, 2563.6,2686.1]	[2534,2565.3, 2495.7,2604.2]	[2712,2761.6, 2656.8,2809.5]	[2848,2896.6, 2796.2,2940.5]	[2927,3032.8, 2913.4,3071.1]
	9	NB=**12**	13	14	15	16	17
		[2161,2189.9, 2128.1,2226.2]	[2190,2215.4, 2157.4,2237.8]	[2428,2459.9, 2376.7,2496.7]	[2512,2546.7, 2471.2,2582.4]	[2633,2665.2, 2607.8,2710.7]	[2640,2686.4, 2607.8,2728.9]
	11	NB =**10**	11	12	13	14	15
		[1952,1975.7, 1921.3,2005.8]	[2027,2054.8, 1999.1,2084.2]	[2127,2154.8, 1999.1,2084.2]	[2262,2288.6, 2226.8,2323.9]	[2428,2459.9, 2376.7,2496.7]	[2527,2564.1, 2490.6,2606.1]
kroA100	**7**	NB= **15**	16	17	18	19	20
		[7291,7327.8, 7241.6,7374]	[8038,8077.5, 8006.8,8120.9]	[8023,8060.6, 7985.4,8109.2]	[8199,8236.3, 8171.8,8284.8]	[8996,9035.3, 8958.2,9069.3]	[9356,9387.4, 9298.5,9439.3]
	9	NB=**12**	13	14	15	16	17
		[6331,6357.4, 6235,6388.9]	[6736,6756.8, 6697.8,6794.2]	[7684,7722, 7644.7,7759.2]	[7721,7754.2, 7687.1,7797.6]	[8257,8287.5, 8220.9,8327.5]	[7916,7951, 7886.5,7995.6]
	11	NB =**10**	11	12	13	14	15
		[5665,5683.9, 5636.6,5708.7]	[6350,6375.7, 6321.1,6401]	[6472,6506.8, 6441.5,6534.3]	[6869,6905.3, 6837.1,6942.6]	[7684,7722, 7644.7,7755.2]	[7614,7645.9, 7570.3,7697.2]

Table-9:continued

Instance	NC	Best tour cost found for different NB					
kroB100	**7**	NB= **15**	16	17	18	19	20
		[7214,7235.2, 7182.9,7291.4]	[8152,8185.1, 8109.2,8222.7]	[8056,8085.4, 8014.7,8125.1]	[8692,8734.1, 8649.2,8773.7]	[8686,8730.8, 8636.5,8782.5]	[9135,9176.4, 9090.7,9229.4]
	9	NB=**12**	13	14	15	16	17
		[6527,6558.8, 6504.3,6589.8]	[7139,7165.4, 7098.3,7200.6]	[7415,7447.7, 7386.2,7483.5]	[7674,7708, 7643,7750.8]	[8271,8300.8, 8222.5,8341.7]	[8135,8173.6,] 8089,8225.4
	11	NB =**10**	11	12	13	14	15
		[5779,5800.3, 5744.5,5817.9]	[6400,6420.1, 6366.7,6440.9]	[6514,6540.5, 6481.4,6578.8]	[7247,7276.6, 7201.9,7310.3]	[7415,7447.7, 7386.2,7483.5]	[7691,7720, 7638,7767]
kroD100	**7**	NB= 15	**16**	17	18	19	20
		[7401,7435.6, 7631.1,7471.8]	**[7386,7424.3, 7344.8,7468.9]**	[7682,7719, 7639.5,7763.8]	[8115,8159.9, 8063.9,8211.3]	[8920,8956.6, 8875.9,9006.9]	[9284,9321.6, 9236.5,9377.6]
	9	NB=**12**	13	14	15	16	17
		[6343,6364.2, 6315.7,6398.4]	[6950,6977.8, [6950,6977.8,	[7271,7303.4, 7238.7,7339.1]	[6965,7001.7, 6927.7,7037.2]	[7926,7968.4, 7894.7,7995.8]	[7939,7978.9, 7890.8,8018.4]
	11	NB =**10**	11	12	13	14	15
		[5615,5633.3, 5587.7,5655.5]	[5795,5820.6, 5768.7,5844.6]	[6305,6324.2, 6272.4,6353.7]	[6768,6793.8, 6738.9,6823.1]	[7271,7303.1, 7238.7,7339.1]	[7359,7388.2, 73247424.8]
kroA150	**7**	NB= **15**	16	17	18	19	20
		[2401,2434.2, 2364.3,2473.9]	[2599,2636.5, 2563.6,2686.1]	[2534,2565.3, 2495.7,2604.2]	[2712,2761.6, 2656.8,2809.5]	[2848,2896.6, 2796.2,2940.5]	[2927,3032.8, 2913.4,3071.1]
	7	NB=**22**	23	24	25	26	27
		[9184, 9228.3, 9132.6, 9294.5]	[9823, 9872.4, 9771.2, 9931.2]	[10344, 10407.1, 10286.7, 10460.5]	[10583, 10643.5, 10528.8, 10702.9]	[10299, 10351, 10235.9, 10410.3]	[10917, 10981.1, 10836.3, 11047.5]
	9	NB=**17**	18	19	20	21	22
		[7806, 7841, 7764.8, 7879.9]	[8560, 8607.7, 8511.6, 8645]	[8695, 8735.1, 8649.9, 8774.7]	[8749, 8798.6, 8702.5, 8849.9]	[9163, 9211.1, 9093.1, 9266.3]	[9471, 9531.8, 9416.5, 9577.7]
	11	NB=**14**	15	16	17	18	19
		[7523, 7559.4, 7481.8, 7590.9]	[7732, 7757.2, 7704.7, 7794.7]	[8058, 8100.7, 8018.4, 8134.7]	[8088, 8121.8, 8043.9, 8157.9]	[8561, 8602.2, 8523.8, 8644.5]	[8808, 8854.1, 8766.1, 8897.9]

References

[1] Current John R., Schilling David A.: The Covering Salesman Problem,The Ohio University,Columbus,Ohio 43210.

[2] Campbell A. M., Vandenbussche D., Hermann W.: Routing for Relief Efforts,transporation system Vol. 42, No. 2, May (2008).

[3] Dorigo, M., Gambardella, L.M.: Ant colonies for the traveling salesman problem. Biosystems 43, 73–81 (1997).

[4] Dorigo, M., Di Caro, G.: The ant colony optimization meta-heuristics. In: Corne, D., Dorigo, M., Glover, F. (eds.) New Ideas in Optimization, pp. 11–32. McGraw-Hill, London (1999).

[5] Gerhard R.: TSPLIB-A traveling salesman problem library, ORSA journal of computing 3(4) 376-384(1999).

[6] Gendreau M, laporte G, Semet F.: The covering tour problem.Computers & operation research;4010 86-94(1992).

[7] Golden, B.L,Nazi-Azimi, Z., Raghavan,S., Salari,M., Toth,P.: The Generalized Covering Salesman Problem,INFORMS journals on computing ,24(4),534-553(2012).

[8] Hachicha M, Hodgson M., Laporte G, Semet F.: Heuristics for the multi-vehicle covering tour problem.Computers & Operation Research :27:29-42(2000).

[9] Kergosien Y., Lenté C., Billaut J.: Home health care problem An extended multiple Traveling Salesman Problem Multidisciplinary International Conference on Scheduling : Theory and Applications (MISTA 2009), Dublin, Ireland 10-12 August 2009.

[10] Khanra A., Maiti M. K., Maiti M.: Profit Maximization of TSP with Uncertain Parameters Through a Hybrid Algorithm, Proceedings of the 4th International Conference on Frontiers in Intelligent Computing: Theory and Applications (FICTA), Advances in Intelligent Systems and Computing Science, DOI10.1007/s40595-017-0099-z, Springer(2016).
[11] Khan I., Maiti M. K.: A novel hybrid algorithm for generalized traveling salesman problems in different environments, Vietnam Journal of Computer Science, DOI: 10.1007/s40595-017-0099-z, Springer(2018).
[12] Khan I., Pal S., Maiti M. K.: A modified particle swarm optimization algorithm for solving traveling salesman problem with imprecise cost matrix, 4th International Conference on Recent Advances in Information Technology (RAIT-2018), DOI: 10.1109/RAIT.2018.8389060.
[13] Khan I., Pal S., Maiti M. K.:A Hybrid PSO-GA Algorithm for Traveling Salesman Problems in Different Environments, International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems, 27(5), 693-717 (2019).
[14] Kotenko I.,Saenko I.: Hierarchical fuzzy situational networks for online decision-making: Application to telecommunication systems Computers & operation research ,39, 2594-2602(2019).
[15] Kumar R.: Blending Roulette Wheel Selection and Rank Selection in Genetic Algorithms,International Journal of Machine Learning and Computing, Vol. 2, No. 4, August(2012).
[16] Lin S, Kernighan B.: An effective heuristics algorithm for travelling salesman problem .Operation Research ;21:498-516(1973).
[17] Labbe M., Laporte G., Martin I., Salzar G.: The ring star problem:polyhedral analysis and exact algorithm.Networks ;43(3):177-89(2004).
[18] Labbe M.,Laporte G.,Martin I.,Salzar G.: Locating median cycles in networks.European Journal of Operations Research ;160:457-70(2005).
[19] Liu B.: Theory and Practice of Uncertain Programming, Physica-Verlag, Heidelberg, (2002).
[20] Liu B., Iwamura K.: A note on chance constrained programming with fuzzy coeffcients,, Fuzzy Sets and Systems 100, 229-233(1998).
[21] Maurya A. K., Kumar N.: Localization Problem In Disaster Management Smartphone Application,International Journal of Advanced Research in Computer Science September-October (2017).
[22] Pandiri V., Singh A., Rossi A.: Two hybrid metaheuristic approaches for the covering salesman problem,Neural Computing and Applications,Springer-Verlag London Ltd., part of Springer Nature (2020).
[23] Pramanik P., Maiti M.K., Maiti M.: Three level partial trade credit with promotional cost sharing, .
[24] Salari ,M., & Naji-Azimi,Z.: An integer programming -based local search for the covering salesman problem.Computers & operation research ,39,2594-2602 (2012).
[25] Salari M., Reihaneh M., Sabbagh M.S.: Combining ant colony optimization algorithm and dynamic programming technique for solving the covering salesman problem.Computer & Industrial Engineering ;83:244-251(2015).
[26] Sierksma, G.: Hamiltonicity and the 3-OPT procedure for the travelling salesman problem. Appl. Math. 22(2), 351-358 (2014).
[27] Tripathy S., Tulshyan A., Kar S., Pal T.: A metameric genetic algorithm with new operator for covering salesman problem with full coverage. Int J Control Theory Appl10(7):245–252 (2017).
[28] Ungureanu V.: Traveling Salesman Problem with Transportation,Computer Science Journal of Moldova, vol.14, no.2(41), (2006).
[29] Venkatesh P., Srivastava G., Singh A.: A Multi-start Iterated Local Search Algorithm with Variable Degree of Perturbation for the Covering Salesman Problem,Harmony Search and Nature Inspired Optimization Algorithms,Advances in Intelligent Systems and Computing 741, (2019).
[30] Zadeh L.: Fuzzy sets. Information and Control, 8 338-356 (1965) .
[31] Zang X., Jiang Li., Ratli M., Ding1 B.: A parallel variable neighborhood search for solving covering salesman problem,Springer-Verlag GmbH Germany, part of Springer Nature (2020).

Supplementary Files

This is a list of supplementary files associated with this preprint. Click to download.

- Highlight.docx
- declarationofcompetinginterests.docx