Operational Research in Engineering Sciences: Theory and Applications
Vol. 5, Issue 2, 2022, pp. 28-60
ISSN: 2620-1607
eISSN: 2620-1747
DOI: https://doi.org/10.31181/oresta300622015b

ORESTA

A NEW HEURISTIC ALGORITHM FOR MULTI VEHICLE ROUTING PROBLEM WITH AND/OR-TYPE PRECEDENCE CONSTRAINTS AND HARD TIME WINDOWS

Unes Bahalke [1], Nima Hamta [2*], Amir Reza Shojaeifard [3], Maryam Alimoradi [4], Samira Rabiee [5]

[1] Department of Industrial Engineering, Iran University of Science and Technology, Tehran, Iran
[2] Department of Mechanical Engineering, Arak University of Technology, Arak, Iran
[3] Department of Statistical Sciences, School of Economics and Management, Università di Bologna, Italy
[4] Baumann Bildung und Qualifizierung GMBH, Daten-analyse und Prozess-analyse, Berlin, Germany
[5] Department of Industrial Engineering, University of Eyvanekey, Eyvanekey, Iran

Received: 18 February 2022
Accepted: 14 March 2022
First online: 30 June 2022

Research paper

Abstract: *In this paper the classic known Multi Vehicle Routing Problem (VRP) is studied where classically several vehicles are set in a central depot, depending on the allocation strategy, each vehicle starts traveling to visit a set of nodes one after another to provide a service of gathering or delivering commodities with the aim of minimizing total traveling distances and costs. In the current paper, this classic problem is extended by new approach of AND/OR precedence constraints (PC) which have not been considered so far in the related literature. Traditionally, PC have been considered in VRPs as the 'AND' type, where the immediate successor of a node cannot be visited until its predecessor nodes have reached their end of services. However by additional OR-type precedence constraints, there are some interconnected nodes through the concept of OR, therein no mandatory need to visit all predecessors of a successor node is acquired before it can be met, and only finishing a part of them can let to that particular node to get visited. This fact is widely observed in pickup and routing and distribution real cases where requisites for some specific products can be fulfilled via various potential suppliers. Implementation of this type of PC graph in VRP is considered as the first introduction and application in the category of this problem. So, initially, the problem is mathematically formulated, then, due to problem's NP-hard complexity and allocation-routing characteristics, a decomposition-based technique is utilized to solve the*

* Corresponding author.
unesbahalke@gmail.com (U. Bahalke), n.hamta@arakut.ac.ir (N. Hamta), amirreza.shojaeifard@studio.unibo.it (A.R. Shojaeifard), m.alimoradi@autlook.de (M. Alimoradi), kavosh2007ra@yahoo.com (S. Rabiee)

problem. The procedure is based on recently enhanced Benders' decomposition approach named as Branch and Search Algorithm, partitioning the original main problem into allocation and routing parts. Unlike the previous version of Benders algorithm, logic based, this newly promoted method acts in a way that at the allocation level, it generates partition of nodes with feasible solutions in lower routing level. The routing part itself has been enhanced by heuristics to cover AND/OR PC graphs. Furthermore, the efficiency of the proposed solution procedure is tested and verified by running on several generated problems in different sizes and in the larger scale it is implemented on the real case of a distribution company in Iran.

Key words: *Multi Vehicle Routing Problem, AND/OR precedence constraints, Hard Time Windows, Branch and Search Algorithm*

1. Introduction

The Vehicle Routing Problem (VRP) is one of the most studied optimization problems and is considered with the optimal routes to be designed by a fleet of vehicles to serve a set of customers (final users) (Golden, 2008). Since many papers have been devoted to the development of VRP, many variants of this problem have been presented by now. For example, the Capacitated VRP (CVRP), in which there is a homogeneous fleet of vehicles where the only constraint is the vehicle capacity, or the VRP with Time Windows (VRPTW), where customers are served in a specified time interval and the schedule of the vehicle trips should be determined. In this paper the new approach of AND/OR precedence relation type has implemented in the body of the classic routing problem. This new implementation will have effects on the traveling schedules and sequences which will lead to changes in processing times and consequently will affect on total time and cost of the whole task assignment and scheduling of travels. The rest of the paper will provide the literature of the previous studies in VRP that will clarify the changes and the effects of this new contribution of AND/OR precedence relationships on the classic routing problems which is the first time introduced in VRPs.

Recently, much attention has been paid to more complicated variants of VRP, which are closer to the practical distribution problems in the real world. Particularly, these variants are characterized by multiple vehicle types, multiple trips, multiple depots or other operational concerns such as loading constraints (Toth & Vigo, 2002). For typical applications of this problem can mention to solid waste collection, school bus routing, dial-a-ride systems, street cleaning, transportation of handicapped persons/workers, routing of sellers/maintenance units. Among the various surveys on the VRP in the book by Toth and Vigo (Toth & Vigo, 2002).

In VRP, the obtained routes must meet operational constraints based on the nature of the transported goods, the quality of service level, and the characteristics of customers and vehicles. Some typical operational constraints are the following: the load of each vehicle cannot exceed the vehicle capacity along each route; the customers can serve as delivery or collection of goods in a route, or both possibilities can exist; and customers can be served only in their time windows and the working times of associated vehicles' drivers (Coelho, 2016).

Precedence constraints determines the order in which the customers should be served in a route. One type of precedence constraint needs that a given customer be served in the same route serving a given subset of other customers and that the customer must be visited before (or after) the customers belonging to the associated subset. This case is called *pickup and delivery problems,* wherein the routes can perform both the collection and delivery of goods, and the goods collected from the pickup customers must be carried to the corresponding delivery customers by the same vehicle. Another type of precedence constraint imposes that if different types of customers are served in the same route, the order in which the customers are visited is fixed. This situation is called *VRP with Backhauls,* wherein again, the routes can perform both the collection and delivery of goods (Zhang, 2017).

This paper considers an important variant of the VRP, in which a fleet of vehicles with different capacities and costs distributes the goods between depots and customers. The problem is known as the Mixed Fleet VRP or as the Heterogeneous Fleet VRP, firstly considered in a structured way in Golden et al. (Golden, 1984).

VRP is one of the major problems in transportation systems that arose from traveling salesman problem (TSP). The goal of TSP is to find the shortest tour among a given set of cities which salesman visited them based on the traveling costs. The solution of feasible assembly sequence of TSP based on the AND/OR precedence can be found by restricting the next city visited by the salesman. This condition was called the constrained TSP (Chen, 1990). Then, all feasible assembly sequences can be generated based on this concept of the constrained TSP (Chen, 1990, 1989, 1990).

De Fazio et al. considered a simplified model and generated the precedence knowledge based on a series of AND/OR rules (De Fazio & Whitney, 1987). MÖhring et al. presented efficient algorithms to solve the general model of AND/OR precedence constraints (MÖhring, 2004). Donald et al. applied AND/OR precedence constraints to assign scheduling tasks according to minimal length schedules (Gillies, 1995). AND/OR precedence constraints have been utilized for scheduling jobs by Donald et al. then two heuristic algorithms were applied to schedule AND/OR task systems and minimize completion time. Finally, the worst-case performance of these algorithms was analyzed by them (Gillies, & Liu, 1990). Möhring applied a linear-time algorithm to deduce additional AND/OR precedence constraints (MÖhring, 2004). Sanghyup et al. considered a flexible job-shop scheduling problems with AND/OR precedence constraints and developed genetic and Tabu search algorithms to solve it (Lee, 2012).

One type of the most important problems in variant of the VRP is multi vehicle routing problem with time windows (MVRPTW) that has been noticed by many researchers and distribution company managers, due to its wide application in urban transportation. In this area, Dong et al. consider a multi-objective VRP with time windows and used a tissue P system based on evolutionary algorithm (Dong, 2018). Anggodo et al. used a genetic algorithm to optimize a multi-trip VRP with time windows on the problems of the tourist routes in Banyuwangi (Anggodo, 2017). Ghoseiri et al. presented a new model and solution for multi-objective VRP with time windows. They used goal programming and genetic algorithm to solve it (Ghoseiri & Ghannadpour, 2010). Chungyu et al. used a hybrid heuristic algorithm to solve multi-vehicle and multi-depot vehicle routing problem with time windows (Chunyu & Xiaobo, 2010). Ariyani et al. used a hybrid method called GA-SA to solve a multi-trip vehicle routing problem with time windows (Ariyani, 2018).

The literature of exact method shows that the usage of exact method has been scarce. MVRPTW with exact solution approaches can be found in the related literature rarely. For example, the branch and price algorithm was one of exact methods which Hernandeza et al. used to solve multi-trip VRP with time windows. The presented method was the first exact solution approaches for this important problem (Hernandeza, 2016). Goel et al. considered a multi-objective VRP and solved it by improved multi ant colony algorithm (Goel & Maini, 2021). This method isn't an exact solution approach. Cömert et al. used a hierarchical approach consisting of two stages as cluster-first route-second to solve a vehicle routing problem with soft time window (Cömert ,2017). Cetin et al. surveyed a VRP with hard time windows wherein pickup and delivery is simultaneous (Cetin & Gencer, 2010).

The branch and price algorithm was presented for multi-trip VRPTW by Hernandez et al. (Hernandez, 2016). Also, an exact hybrid method as combining a branch-price-and-cut (BPC) algorithm was used to solve the VRPTW by Alvarez et al. wherein deliverymen had been considered multiple (Alvarez & Munari, 2017). Parragh et al. considered the truck and trailer VRPTW and used a branch and price and adaptive large neighborhood search. A vehicle routing problem with a heterogeneous fleet and time windows was introduced by Jiang et al. They used the Tabu search algorithm to solve it. Presented VRPTW by Miranda et al. considered stochastic travel and service time (Jiang, 2014). A meta-heuristic method was applied to solve the vehicle routing problem with time windows by Bouthillier et al. (Le Bouthillier & Crainic,2005). Nazif et al. applied a genetic algorithm to solve vehicle routing problem with time windows (Nazif & Lee, 2010). A multi objective vehicle routing problem with time windows has been considered by Chiang & et al. they used an evolutionary algorithm to solve it (Chiang & Hsu, 2014). Bae et al. surveyed a multi-depot vehicle routing problem with time windows wherein delivery and installation vehicles was considered (Bae & Moon, 2016). Wang considered a hybrid swarm optimization genetic algorithm to solve vehicle routing problem (Wang, 2015). Kumar et al. considered a time-dependent VRP with time windows and solved it using a genetic algorithm as one of meta-heuristic methods (Kumar & Panneerselvam, 2015). Pierre et al. solved a VRP with time windows using a genetic algorithm (Pierre & Zakaria, 2016). Koc et al. developed heterogeneous fleet vehicle routing problems with time windows then utilized a hybrid evolutionary algorithm to solve it (Koc, 2015). Dabia applied a Branch and price as an exact method to solve a vehicle routing problem with time windows (Dabia, 2013). Azi et al. applied an exact algorithm and solved the VRPTM wherein multiple use of vehicles had been considered (Azi, 2010). Mingozzi et al. presented an exact method to solve the multi-trip vehicle routing problem. Their computational results indicated that the proposed exact algorithm can solve MTVRP (Mingozzi, 2013). Hernandez et al. solved the multi-trip vehicle routing problem with time windows by presentation a branch and price algorithm (Hernandez, 2016). Hernandez et al. presented an exact two-phase algorithm to solve the multi-trip vehicle routing problem with time windows and limited duration, wherein first phase considered possible ordered lists of clients based on the maximum trip duration criterion. And the second phase utilize a Branch and Price algorithm to generate and choose a best set of trips so that all customers are visited (Hernandez, 2014).

The previous researches are summarized in Table 1. By examining the frequency of methods presented in previous researches and the tendency to achieve an accurate and optimal solution, conducting research using exact methods will be felt. Since recently several studies have focused on exact method, in this paper will apply a branch and search algorithm to solve MVRPTW.

Table 1. A classification of VRP in the recent literature

Title of articles	Single-trip VRP	Multi-trip VRP	Solution method	
			Exact method	Meta/heuristic method
A new exact algorithm to solve the multi-trip vehicle routing problem with time windows and limited duration (Hernandez, 2014)		✓	✓	
Branch-and-price algorithms for the solution of the multi-trip vehicle routing problem with time windows (Hernandez, 2016)		✓	✓	
An exact algorithm for the multi-trip vehicle routing problem (Mingozzi, 2013)		✓	✓	
An exact algorithm for a vehicle routing problem with time windows and multiple use of vehicles (Azi, 2010)		✓	✓	
Branch and price for the time-dependent vehicle routing problem with time windows (Dabia, 2013)	✓		✓	
A hybrid evolutionary algorithm for heterogeneous fleet vehicle routing problems with time windows (Koc, 2015)	✓			✓
A knowledge-based evolutionary algorithm for the multi objective vehicle routing problem with time windows (Chiang & Hsu, 2014)		✓		✓
Optimized crossover genetic algorithm for vehicle routing problem with time windows (Nazif & Lee, 2010)	✓			✓
A cooperative parallel meta-heuristic for the vehicle routing problem with time windows (Le Bouthillier & Crainic, 2005)	✓			✓
The vehicle routing problem with hard time windows and stochastic travel and service time (Miranda & Conceição, 2016)	✓			✓
An exact hybrid method for the vehicle routing problem with time windows and multiple deliverymen (Alvarez & Munari, 2017)	✓		✓	✓
Vehicle routing problem with a heterogeneous fleet and time windows (Jiang, 2014)	✓			✓
Branch-and-price and adaptive large neighborhood search for the truck and trailer routing problem with time windows (2017)	✓		✓	
Branch-and-price algorithms for the solution of the multi-trip vehicle routing problem with time Windows (Hernandez, 2016)		✓	✓	
A new approach for solution of vehicle routing problem with hard time window (Cömert ,2017)	✓			✓

Title of articles	Single-trip VRP	Multi-trip VRP	Solution method	
			Exact method	Meta/heuristic method
Improved Multi-Ant-colony algorithm for solving Multi-Objective Vehicle Routing Problems (Goel & Maini, 2021)		✓		✓
Branch-and-price algorithms for the solution of the multi-trip vehicle routing problem with time windows (Hernandeza, 2016)		✓	✓	
Hybrid Genetic Algorithms and Simulated Annealing for Multi-trip Vehicle Routing Problem with Time windows (Ariyani, 2018)		✓		✓
Research on multi-vehicle and multi-depot vehicle routing problem with time windows electronic commerce (Chunyu & Xiaobo, 2010)		✓		✓
Multi-objective vehicle routing problem with time windows using goal programming and genetic algorithm (Ghoseiri & Ghannadpour, 2010)		✓		✓
Optimization of multi-trip vehicle routing problem with time windows using genetic algorithm (Anggodo, 2017)		✓		✓
A tissue P system based evolutionary algorithm for multi-objective VRP with Time Windows (Dong, 2018)		✓		✓
A time-dependent vehicle routing problem with time windows for e-commerce supplier site pickups using genetic algorithm (Kumar & Panneerselvam, 2015)	✓			✓

As we can see and study in the literature of VRP, we could not find any paper referring to implementation of AND/OR precedence constraints in the problem structure. However, we can see this type of relationships in the real situations and industries in which ignoring this fact will result in not optimized assignments and sequences that will lead to more costs to industry owners. In this study, this approach is introduced and it is tried to clarify these costs. Due to the huge complexity of the problem, a hybrid general algorithm is designed and proposed to handle the problem.

This paper is structured as follows. Multi-Trip Vehicle Routing Problem with hard Time Windows (MVRPTW) problem with AND/OR-Type precedence constraints is explained in section 2. In section 3, developed algorithms to solve the considered problem is described in detail. Section 4 presents the computational experiments in which the results obtained by proposed algorithm. Finally, Section 5 is devoted to conclusions and recommendations for future research.

2. Problem Description and Mathematical Formulation

2.1. AND/OR-MTVRPTW

In this section, the AND/OR Multi-Trip Vehicle Routing Problem with hard Time Windows (AND/OR-MTVRPTW) problem is described and formulated in a mixed-

integer linear programming model. The problem is generally based on the well-known Vehicle Routing Problem with Hard time Windows with more similarity to very familiar Pickup and Delivery type which has got many attentions in last recent years and there are huge numbers of accomplished and running competing studies, focusing on optimum node visit scheduling and permutations in the routes.

One of the contributions of this paper is to applying a new form of PC arc into VRPs, as an important factor in scheduling, which has resulted in more complex problems but with the better cost-oriented outcome. To the best of authors' knowledge, this issue has not been considered so far in the history of VRPs. This type of synchronization has existed in the real VRP problems, but in most cases, it's denied or simplified by forcing it to change into an AND-type PC arc. So, in order to make the benefit of considering AND/OR PC arcs, the main and the first point is to identify the synchronizations with OR-type characteristics and making them group together with linked OR-arcs. Finally, implementing the developed mathematical model and algorithms would lead to time and cost benefits for companies to deal with this kind of problem. When the details of the problem are clarified as following, more details will be illustrated that how it will end up with better results.

As mentioned before, the problem deals with traveling of a limited number of vehicles along with the geographically scattered nodes/customers in order to deliver products or serve a service. According to Li & Lim benchmark web page (Li & Lim, 2008), the first priority goal is to minimize the number of used vehicles and the second priority is to minimize the total traveled distances of vehicles. The position of each node is declared as a point on the XY $-$ $plane$ and the distances between nodes are calculated by using Cartesian coordinates $A(x_1, y_1)$ and $B(x_2, y_2)$. All vehicles start their travels at the time zero on the first trip (p_1) from the depot (n_0) and are allowed to have several trips. Each trip starts from the depot and ends to the depot. All vehicles are the same and are limited by maximum capacity restrictions which must not be exceeded in every trip of the particular vehicle. The program of traveling should be planned in a way that at the end of the travels, all nodes/customers must be visited once, not more, not less. Each customer has its own time window for taking service, declared as $[\text{EarlyTime}_i, \text{LateTime}_i]$ which means that it would not accept service/delivery before or after its determined window. Thus, if a vehicle reaches a node at a time before EarlyTime, it must wait until that time window opens. Also, it is clear that received after $LateTime$ is prohibited.

The above descriptions are the basic conditions of a VRPTW problem, except for the multi-trip part. The maximum allowed trip number for every vehicle had been fixed at one. But in this paper, in order to get more harmony with our considered problem, vehicles are allowed to do multi trips in which maximum of trip numbers are bounded. In the following, the considered AND/OR synchronization constraints are clarified.

By application of AND/OR synchronization constraints, permutations would lead to change, consequently, every cost function which is derived by or dependent on sequences will be affected.

There are two types of PC arcs that are assumed as AND-type and OR-type. The first type refers to classic precedence relationships defined as linked arc from i to j $(i \rightarrow j)$, in which i is the precedence of j that must be completely processed prior to j.

In some new versions, the restriction on completely has been changed and limitations on finishing precedence task have been reduced. In general, in the literature of the VRPs, only the first type has been considered. As it was mentioned, in some cases, various types of network graphs with different levels of restrictions have extended and formed. The most well-known type of PC is found in pickup and delivery VRPs where nodes are linked as a pair by an AND-type arc. The pickup node has the precedence role and the delivery node has the successor role in which the pickup one must be completely processed before reach the delivery one. In this paper, this type of synchronization is considered as an AND-type PC arc where there is no restriction to put an AND-type pair into a single same vehicle anymore.

To better illustrate this difference, according to our considered problems, there are no pickup and delivery services, but still, there are a kind of limitations in which some specific nodes in the same route, should have been visited before their pairs. So, there is no obligation to put an AND-type pair into a single same vehicle and they are free to be serviced individually by any vehicle. But if both paired nodes determined to be visited by the same vehicle, PC must be implemented and it would make them met the precedence relationship constraint.

This approach would lead to higher complexity of the problem. In fact, when the problem consists of N nodes and K vehicles, where all nodes are paired, it deals with the allocation of $N / 2$ nodes into K vehicles. However, by eliminating the same vehicle restriction of paired nodes, by allocating N nodes into K vehicles, the complexity of the problem will be doubled.

Furthermore, the second OR-type PC arc has also considered in this study where it is counted as the first implementation of OR-type arc in VRPs. This PC arc is implemented when starting a process is dependent to other processes. It is defined for tasks with multiple predecessors where finishing of only one predecessor would let the successor could start its process. In many real industrial cases, because of the simplicity or due to Lack of knowledge and awareness of the subject, they were mistakenly considered as AND-type arcs. Forcing and restricting a set of processes into limited options which were led to detrimental consequences, losing potential better scheduling alternatives. For an example, consider a subset of nodes A, B and C in a routing problem with N nodes and node C as a successor linked to nodes A and B by AND-type arcs. Assume that in a practical condition, visitation of node C could start its process by either completion of service to A or B. Besides, assume that the distance between B and C is relatively large compared to the distance between A and C. So, according to the classic formation of AND arcs, both nodes of A and B must be visited prior to C even if its final permutation lead to worse traveling costs. But if both of them were initially defined as OR-type, the order of serving node B could be moved to any further sequence after visiting node C. Thus, by these newly opened alternatives they could have resulted in at least better outcomes in terms of traveling costs.

With all the above interpretations, it is now easier to understand that how would consider the OR-type pc arcs in a vehicle routing problem could bring benefit to a company. It can widen the alternatives by providing more open space with possible better options for the decision makers. This can lead to more efficient plans with better outcomes. In the following, the introduced problem is mathematically modeled used notations are declared.

2.2. Mixed-Integer Mathematical model

In this section the problem is mathematically formulated and the used notations and variables are declared. Further assumptions of the problem are assumed as following:

- There are N customer nodes plus the node depot as Node 0.
- Each customer has a demand that must be supplied by the unique vehicle.
- Each customer must be visited only once.
- Either AND or OR-type arcs are implemented when linked pairs assigned to be visited by the same vehicle.
- Each node has a fixed value of service time.
- No stochastic parameter or input data is considered.
- Preemption or interruption is not allowed
- Vehicles' speeds are equal to one unit.
- Every route is traveled with single vehicle and includes multi trips each starting from the depot and ending to the depot.

Following introduces notations and their corresponding definitions:

i, j, t	Indexes for all nodes
k, s, r	Index of vehicle
K	Maximum number of available vehicles
p, n	Index of a trip in a route
N	Number of all nodes
$Dis_{i\,j}$	Distance between node i and j
D_i	Demand of node i
$Cost$	Cost of using a vehicle
$Capacity$	Maximum capacity of a vehicle
P	Maximum number of Allowed trips
Service_i	Service time on node i
EarlyTime_i	Lower bound for time window of node i.
LateTime_i	Upper bound for time window of node i.
V_k	A binary variable indicating use of vehicle k.
$x_{i\,j\,n\,k}$	A binary decision variable indicates visit of node j immediately after node i at trip n by vehicle k.
$z_{i\,n\,k}$	A binary variable indicating assignment of node i on trip n by vehicle k.
$St_{i\,k}$	Start time of servicing to node i by vehicle k.
$Co_{n\,k}$	Reach time to node depot after finishing trip n by vehicle k.
$y_{i\,j\,k}$	A binary variable that indicates if visiting of node i occurred earlier than node j by vehicle k.

The mathematical model is developed as follows:

$$\text{minimize } Obj = \sum_{i\,j\,n\,k} x_{i\,j\,n\,k} \cdot dis_{i\,j} + \sum_{k=1}^{K} V_k \cdot cost \tag{1}$$

$$V_k \geq z_{0\,1\,k} \tag{2}$$

$$V_k \geq V_{k+1} \tag{3}$$

$$z_{i\,n\,k} = \sum_{j} x_{i\,j\,n\,k} = 1 \qquad for\ all\ i\ , n, k \tag{4}$$

$$z_{j\,n\,k} = \sum_{i} x_{i\,j\,n\,k} = 1 \qquad for\ all\ j, n, k \tag{5}$$

$$y_{i\,j\,k} + y_{j\,i\,k} \geq \left[\left(\sum_{n} z_{i\,n\,k} + \sum_{n} z_{j\,n\,k}\right) - 2\right] \cdot \text{BigM} + 1 \qquad i \neq j \neq 0 \tag{6}$$

$$y_{i\,j\,k} + y_{j\,i\,k} \leq 1 \qquad for\ all\ i \neq 0, j \neq 0, k \tag{7}$$

$$\sum_{r} y_{i\,j\,r} + y_{j\,i\,r} \leq \left[2 - \left(\sum_{n} z_{i\,n\,k} + \sum_{n} z_{j\,n\,s}\right)\right] \cdot \text{BigM} \qquad i \neq j \neq 0 \tag{8}$$

$$St_{j\,k} \geq St_{i\,k} + \text{servic}_i + (y_{i\,j\,k} - 1) \cdot \text{BigM} \tag{9}$$

$$\sum_{i \in \overrightarrow{or} j} y_{i\,j\,k} \geq [\sum_{n} (z_{t\,n\,k} + z_{j\,n\,k})] - 1 \qquad for\ all\ t\ \overrightarrow{OR}\ j, \tag{10}$$

$$\sum_{i} D_i \cdot z_{i\,n\,k} \leq Capacity \tag{11}$$

$$St_{j\,k} \geq St_{i\,k} + \text{servic}_i + Dis_{i\,j} + \ (x_{i\,j\,n\,k} - 1) \cdot \text{BigM} \tag{12}$$

$$\sum_{j} x_{i\,j\,n\,k} - \sum_{j} x_{j\,i\,n\,k} = 0 \qquad for\ i = \{0,1,2,..,N\} \tag{13}$$

$$St_{j\,k} \geq St_{i\,k} + \text{Service}_i \qquad for\ i\ \in i\ \overrightarrow{AND}\ j \tag{14}$$

$$Co_{n\,k} \geq St_{j\,k} + \text{Service}_j + dis_{j\,0} + (x_{j\,0\,n\,k} - 1) \cdot \text{BigM} \tag{15}$$

$$St_{j\,k} \geq Co_{n\,k} + dis_{0\,j} + \ (x_{0\,j\,n+1\,k} - 1) \cdot \text{BigM} \tag{16}$$

$$St_{i\,k} \geq \text{EarlyTime}_i \tag{17}$$

$$St_{i\,k} \leq \text{LateTime}_i \tag{18}$$

Equation (1) presents the objective function of minimizing travel cost which includes traveling distance related costs and vehicles using costs. By giving a very big positive value to $cost$, model gives a first priority to minimize the total number of used vehicles. Constraints (2) and (3) show the usage of vehicle k. Equations (4) and (5) indicate assignment of nodes i and j to route k. Constraints (6), (7) and (8) indicate that if two nodes are in the same route of a vehicle, variable y will take value and one node will be serviced sooner and one another, later. In other words, if two individual nodes are assigned to two different routes, no sequence relationship is defined between those two. Constraint (9) synchs starting times to visiting orders. Constraint (10) is developed in order to declare OR-type precedence relationship. It defines that if node j is linked by a set of OR-type predecessors, servicing of only one member of them, those who are in the same route with node j, is adequate to let the successor j to be visited. Constraint (11) limits the vehicle's capacity. Constraint (12) assigns starting times for nodes in a route which are visited consecutively. Equation (13) declares that every node in a route has one input and one output. Constraint (14) shows AND-type relationships. It is clear that every node i which is paired with a successor node j in a route k must be visited prior to j. Constraints (15) and (16) make a link between completion time of a trip in a route and start time of the first customer in next trip. At the end, constraints (17) and (18) implement the hard time window restrictions.

3. Developed Algorithms

3.1. General Scheme

In this section, our developed algorithm is introduced. The applied algorithm's general scheme is completely new and it is based on the integration of rules, random search and a new intelligent Large Neighborhood Searching technique which is entirely fit to VRP problems. Also, in the inner layers of the algorithm, the latest heuristic algorithms' key implementations, well-known meta-heuristics like Simulated Annealing and Tabu search are also considered. Moreover, in order to make use of Genetic Algorithms' wide diversity characteristics, some solution pools are created to avoid losing elite solutions.

The challenging part in the VRP problems with hard time windows is the feasibility check. In many studies of the literature, the complexity of problem makes researchers to use mathematical models to check the feasibility of every generated solution. Since it consumes the most part of computational time and it will become more complex by adding AND/OR precedence arcs, the performance of any presented algorithm will be affected. Therefore to overcome this challenge, a new solution has been proposed in this paper, and the use of mathematical models has been greatly limited, and rule-based techniques have been used instead. Entire process of the developed algorithm is presented as follows.

The general procedure of the algorithm starts by a heuristic rule OR_ENH_HEU which generates a feasible solution, considering two main factors of using maximum

capacity and minimum traveling costs. Then, the obtained solution goes through two different techniques called Parent and Child in order to eliminate vehicles those are recognized as bad allocations. Next, the remained bad allocations' routes and their belonging nodes are eliminated from the solution and are transferred to stack.

Afterwards, some techniques are applied along the algorithm in order to add stacked nodes as rule-based adding, randomly adding and randomly substitutions with the aim of reconstructing a complete solution with minimum cost. Hence, according to the strategy beyond the minimizing vehicles, it is called drop and add.

Furthermore, the algorithm deals with two searching techniques, the local search and the intelligent Large Neighborhood Search called intllLNS from now on. The local search itself includes two different perturbation methods MOV_PERTURB and REP_PERTURB where tabu movements and Simulated Annealing approaches are implemented there.

The intllLNS procedure is completely novel and acts based on the characteristics of the nodes in the same route. It evaluates their geographical positions and also creates time zones according to same routed nodes' early and late times. Then the large numbers of intelligent movements or replacements are done based on the initial evaluations.

One of the useful advantages of the developed algorithm is its solution pool which is active any time a solution is generated. It determines whether a new feasible solution be admitted to the pool or not. This option helps to avoid losing diversity and also keep holding elite solutions.

Another advantage of this algorithm is that vehicles can always be removed along the local search and intllLNS. This is possible due to the elimination approach that is embedded in all stages of the algorithm. By means of this approach while a feasible solution with fewer vehicles is produced, then there is no chance to get back to former worse allocations. This obligation is implemented due to the high priority of minimizing number of vehicles. Additionally, at the end of every iteration, the Child procedure is performed to be certain about the minimum possible number of vehicles. it is clear that Child would not guarantee optimum number of vehicles but it does some attempts to minimize it as much as possible based on the rules, allocations and sequences provided by the whole process. The proposed algorithm is named as Hybrid AND/OR Intelligent Large Neighborhood Search algorithm called as HAOR_intLNSA. Its' general scheme is interpreted in the following flowchart in Figure 1.

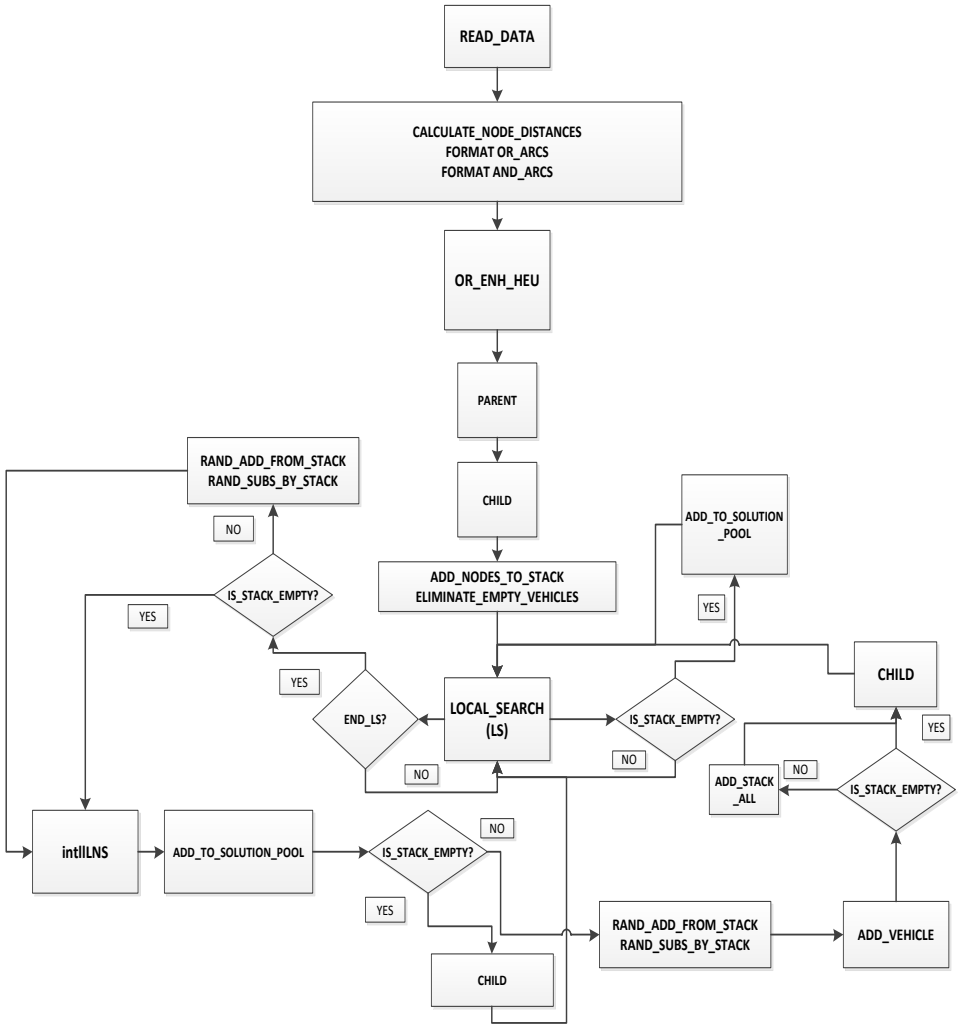

Figure 1. HAOR_intLNSA flowchart

The pseudocode of the main function of the HAOR_intLNSA is presented in Figure 2.

```
main () {
        READ_DATA();
        Calculate_Distances();
        Format_Arcs();

        OR_ENH_HEU();

        PARENT();

        CHILD();
        ADD_TO_STACK();

NEW_CYCLE:

        LOCAL_SEARCH(){
        int LS_iteration = 0;
        do {
        int rand_number = rand ();
        if (rand_number < perturb_orient_percentage) {
        TYPE_1_PERTURB();   }
        else {
        TYPE_2_PERTURB();
                                        }

        if (IS_STACK_EMPTY) {
                sort_pool();
                consider_new_solution();
                        }
        else{
                rand_add_from_stack();
                rand_subs_by_stack();
                }
                LS_iteration++;
                } while (LS_iteration < MAX_LS_iter);


                        } //local search end;
        intllLNS () {

        int LNS_iteration = 0;
        do () {
        int lns_rand_number = rand ();
        if (lns_rand_number < intllLNS_orient_percentage) {
        TYPE_1_intllLNS();   }
        else {
        TYPE_2_intllLNS();
                                        }

        if (IS_STACK_EMPTY) {
                sort_pool();
                consider_new_solution();
                        }
        else {
                rand_add_from_stack();
                rand_subs_by_stack();
                }
         LNS_iteration++;
                }while (LNS_iteration < MAX_LS_iter);

} // intllLNS end;

        if (IS_STACK_EMPTY) {
                if (is_terminal_condition_met){
                        sort_pool();
                        CHOOSE_BEST_POOL_MEMBER();


                        } //algorithm finishes here
                else {
                        goto  NEW_CYCLE;
                        }

        }
        else {
                add_vehicle();
                 //try to add all stacks into all available routes

                add_stack_to_all() {
                        one_by_one_add();
                        if (succeed_adding) {
                                                        sort_pool();
                                                        consider_new_solution();
                                                        goto NEW_CYCLE;
                                                }

                }

        } //end of else;

} //main end;
```

Figure 2. HAOR_intLNSA pseudocode

3.2. OR_ENH_HEU

HAOR_intLNSA starts by a heuristic rule which at the end of its process, it generates one feasible solution. This solution is constructed with focus on late times as first priority and then traveling distance as second priority. **OR_ENH_HEU** starts assignments of nodes to routes through a step-by-step procedure. At each step a set of candidates ($CandidateSet$) is setup, and then every individual node from this set is tested by adding to current solution which its feasibility is evaluated through a rule-based feasibility checking process. Candidates who successfully pass the evaluation stage will be included in the final set of candidates called as $NomineeSet$. At final step, the nearest node to the last positioned node in the current solution is admitted as new assignment. Finally, the $CandideSet$ are updated and all members in $NomineeSet$ are eliminated. This procedure continues until all nodes have been assigned to routes.

At the beginning of the rule, number of input OR-Type arcs of each node i (if any) is held in variable $ORpre[i]$. Then the $CandidateSet$ is reconstructed at the beginning of assignment for every new route, considering $ORpre[i]$ in which all nodes with no input OR arc ($ORpre[i] == 0$) will be included in the CandidateSet. The reason beyond this action is to give more option and chance for OR-predecessor nodes to assign. Because fixing an OR-Type successor node before any of its predecessors in a route would not let any of its predecessors to assign to that route.

Step 1: Declare variables for all i $ORpre[i] = 0$;
Declare variable $holder = 0$;
Declare $VehicleNumber = 0$;
for all $i \neq 0$ Declare $isVisited[i] = 0$;
Step 2: for all i, if ($ORpre[i] == 0$ &&$isVisited[i] == 0$) $Then$ add i to CandidateSet ;
Step 3: **if** ($FeasibilityCheck\ (CandidateSet(i)\ ) ==$ **true**)$then$ add i to NomineeSet ;
if ($CandidateSet == \emptyset$) $then$ **goto** Step 6;
Step 4: **if** ($CandidateSet \neq \emptyset$ &&$NomineeSet == \emptyset$) $then$ {$VehicleNumber++$;
$holder = 0$; $CandidateSet = \emptyset$; **goto** Step 2; }

Step 5: **if** (Min distance == $dis[holder][NomineeSet(i)]$)
$then$ {$holder = i$; $isVisited[i] = 1$; }
for all i **if** ($pc\ (holder \rightarrow i) == ORType$) $then$ { $ORpre[i] - -$;
if ($isVisited[i] == 0$)add i to $CandidateSet$;
}
for all i **if** ($pc\ (i \rightarrow holder\ ) == ANDType$)
$then$ {remove i from $NomineeSet$; **goto** Step 2; }
Step 6: End;

Figure 3. Pseudocode of OR_ENH_HEU

After each successful assignment, $CandidateSet$ should be updated where newly assigned node will be eliminated. Then it will be checked that whether the assigned node is an AND-Type successor or not. If it is AND-Type successor, then its paired predecessor will be removed from $CandidateSet$. Also, if assigned node is an OR-Type predecessor of node i, then $ORpre[i] = ORpre[i] - 1$. Also, node i will be added to

$CandidateSet$ for the current route since one of its predecessors has been finished already in that route. Finally, the general procedure of OR_ENH_HEU rule is interpreted as pseudocode in Figure 3.

One of the advantages of **HAOR_intLNSA** is the application of a rule-based feasibility check since it takes much less computational time than the used application of mathematical models. The procedure of this rule is declared in following section.

3.3. Rule-based feasibility check

According to the described procedure of generating feasible solutions, the rule is doing core job of creating complete route. Whenever the feasibility of a node is checked, the rule determines the visiting order of all allocated nodes in the route. Then, it checks feasibility in the aspects of AND/OR precedence relationships, number of trips and time windows. Next, it returns constructed route along with a true or false value in the case of feasibility or infeasibility.

The core logic of this rule is based on the giving priority to nodes with earlier late times. This simple but efficient rule is implemented repeatedly in classic VRP and single machine scheduling problems dealing with due dates (Hu, 2018),(Gordon, 1997) and its efficiency is proved already. Due to the extended aspects of the in-handed problem and its high complexity, simple sorting of nodes would not be a cost-efficient work, because it might be losing a high number of feasible solutions due to its weak created order of visits in the route. So, all assigned nodes in a specific route are determined in a permutation of visiting order one by one where in each step, noted constraints are tested.

Since the rule checks all aspects of feasibility, it will not lead to return a true feasible feedback in contrast with mathematical model, but it is still possible to lose some mathematically feasible solutions. Regarding this issue, two conditions are defined that if rule falls the route into the following two categories, it will not return directly a decision on feasibility. The first one is a condition in which all nodes are visited in their time windows and all precedence constraints are met but the number of trips exceeds the predetermined upper bound. In this case, the considered assignment is sent to check by mathematical model and the model's feedback is referred as feasibility status. The second one refers to a condition in which the number of trips is in bound and precedence relations are met but time window restrictions are violated. In this situation, first, feasibility of time window restrictions is checked under a completely hypothetical condition without capacity constraints with one trip. If an assignment successfully passes feasibility check, then a shifting method is applied in order to check possible feasibility of initial formed route. It acts in a way that if forward trip consists of lower capacity usage than former trip, shifting replaces last ordered node from former trip into forward trip and then it checks if this action lead to feasible route. Shifting iterates until find a feasible route but if all possible shifting replacements are done and no feasible route is found, then rule returns final false decision of feasibility.

The initial application of the simple rule-based feasibility check was led to weak results where it was losing lots of feasible solutions. But by evaluations of differences between mathematical and rule's created routes, it's found that embedding of above introduced novel revising methods could bring a significant improvement. So, one of

the most important weaknesses in the application of rule-based feasibility check has been largely eliminated. Since computational time of an algorithm is one of the most prominent factors on deciding an algorithm, using rule-based methods will considerably reduce computational time.

3.4. Removing inefficient routes

After creating a feasible solution, **HAOR_intLNSA** sets up the routes in order to drop those vehicles with low consisting nodes. By dropping the vehicles, their belonging nodes are transferred to stack and they remain there until find a good assignment to re-enter them into the solution.

Before this elimination occurs, the algorithm does some attempts to move the nodes which are belonged to the targeted routes, into the rest. It should be noted that the targeted routes are diagnosed by means of an input parameter, $meanThreshold$, indicating a level for the number of nodes in a route. Thus, all routes under the $meanThreshold$ are included in the target set.

The algorithm continues with two steps before that elimination happens. The first step is called $Parent$ in which for all nodes in targeted routes, their chances of assignment in the rest routes are tested. This checking for possible movement starts from the nearest route and ends to the farthest one and in the meantime anywhere it leads to a feasible solution, checking for that node ends and the movement is done. After ending $Parent$, there might still some nodes remain in the targeted routes, so, the algorithm continues with a uniting approach in Step 2 called $Child$. It tries to unite sporadic nodes. For this aim, the $Child$ iteratively moves nodes into other feasible routes among the targeted routes. These movements repeat until no better movements are possible in the aspect of the number of empty routes.

After the above steps, the targeted routes will be updated since it might some changes in the number of nodes in routes have occurred during $Parent$ and $Child$. Then, newly diagnosed targeted routes are eliminated and their belonging nodes are transferred to stack in a process called $AddToStack$.

By ending the process of elimination and $AddToStack$, the algorithm enters into the cycle part, starting with a local search approach to find neighbors with possible better solutions. In this paper, two types of perturbation strategies are applied which are defined and described in the following section.

3.5. Local Search

In this paper, two types of perturbation approaches are implemented as local search. The first type is a well-known perturbation method which has used consecutively in the literature. In this type, two randomly selected nodes from the randomly selected route are marked in order to be replaced by two randomly selected nodes of the randomly destined route. If this replacement ends up with an acceptable result, it will be done but if not, then the action is considered tabu.

This tabu consideration might use memory, however it avoids wasting computational time by blocking repetition. Furthermore, In order to widen the search space, the same strategy as Simulated Annealing is implemented. The sum of travel cost of the route of origin and the destination route, before replacement, is compared

to the sum of costs for both after replacement. By setting up an acceptable range, the resulted replacements might be done even though the latter costs are worse than the initial ones. This range has determined by an input parameter called $SaRate$, by a value bigger than 1.

The second type of perturbation is a novel movement approach that moves nodes instead of replacement. The logic of this perturbation is to consider the elimination of vehicles as much as possible. When a movement perturbation occurs, that selected node might be the last member of that route, so by transferring it, the origin vehicle will be useless and automatically will be removed from the solution. It should be noted that all movements during the algorithm proceeding have arranged in a way that only occurs between routes that are not empty.

In order to move a single node to another active route, the feasibility of this movement and also the resulted cost difference between the former and after movement, will be considered. Although the route of origin is selected randomly, the destination route is not. The movement of that particular selected node goes through a complete test in all other active routes. At last, the elite route which has led to the highest improvement in cost will be selected to transfer. The same tabu and $SaRate$ related strategies, which are used in the first type, are considered in this type of perturbation, too.

Every time a perturbation occurs and a complete feasible solution is produced, it is checked for entering into the provided solution pool. The pool is embedded with the aim of storing elite solutions. It is limited in the size which is determined by a $poolSize$ parameter. A solution should meet two conditions to get into the pool. First, its cost should be different from those solutions which are already stored in the pool. Besides, its cost should be less than the last member of the pool.

Next, if this admission process is successful, then the whole pool will be re-sorted based on solutions' cost values. This whole procedure is called $ConsiderNewAllocation$.

Finally, because there are probabilities of resulting with some empty vehicles, the final solution of this step is reconstructed and all routes are rearranged, with a new label of numbers, in order to completely remove those empty routes which are remained between active ones.

3.6. Add from stack

There are three kinds of methods, which are developed in order to pick nodes from the stack and add them into solutions, or to replace them. These picking efforts are embedded at different parts of the algorithm, during, and after perturbing, during, before, and after intelligent LNS and after adding a new vehicle.

The first type iteratively selects a random node from the stack and tries to embed it into a randomly selected route. This type is called $addFromStack$. The second type which is called $SubStack$, attempts to substitute a random node of a random route with a random node of the stack. It does this replacement if that substitution leads to a better cost function. At last, the type $StackToAll$ tries to vacate the stack pool, by testing every individual member of the stack into all routes. These three methods are partially embedded inside the local search as well as intllLNS.

In search-based optimization algorithms, it is logical to design some techniques to diversify the search space due to avoiding trapping in local optima. In this paper, a novel procedure is developed which its details are clarified in the next section.

3.7. Intelligent Large Neighborhood Search

In many of large neighborhood search algorithms, it is usual to see big changes at once since a large group of individuals are decided randomly to alteration. In the AND/OR MTVRPTW there are restrictions in which group random alterations would lead to high number of infeasibilities due to two types of precedence relationships, decisive time windows and capacity of the vehicles in trips. Although this type of LNS has seen in the literature of PDPTW, decision to do randomly large perturbations would not be an adroit action.

For this reason, in this paper, two approaches are developed to perform LNS. Both of them are designed based on the problem's main characteristics. They will remove a bunch of infeasible or inefficient moves. Instead, they will act in an intelligent way by setting up an elite group. This group consists of nodes that have a high probability to fit in destined routes, which might lead to better solutions.

The approach focuses on the geographical position of nodes in a route on XY-Plane. At the first step, it gathers information of each route's traveling costs and defines an index as $costIndex = \frac{travelCost_k}{NumberOfNodes_k}$ for all routes of a solution. Lower values of this index, indicates that the corresponding vehicle of the route travels reasonable distance compared to the number of customers it must visit. Consequently, the higher of this index, shows inappropriate allocation, in which the vehicle might be able to carry out more customer visitations along that traveled distance. According to this defined index, all routes of a solution are sorted from high values to low. Next, random nodes from a set of candidates are selected to transfer to those routes which are at the top of the sorted list with high values of index. For setting a group of candidates, the proposed LNS acts in an intelligent way by finding a set of nodes which are seem to be appropriate choices to transfer to specific routes. Due to this action, many of the random choices and vain attempts with low probability of success perturbation will be eliminated. In order to clarify this action, see Figure 4. It is a sample of a route for a vehicle which should visit 6 customers in two trips. In this Figure, it is clear that nodes A and B would not be appropriate choices for entering to route k concerning the aspect of traveling costs. On the other side, nodes C and D seems to be more fit. In order to discern and distinguishing these set of appropriate and inappropriate nodes, intllLNS performs a rule. Based on this rule, for each route a center point in XY-plane is calculated as $X_{center_k} = \frac{\sum_{i \in k} X_i}{NumberOfNodes_k}$, $Y_{center_k} = \frac{\sum_{i \in k} Y_i}{NumberOfNodes_k}$, then by means of this center point and the radius equal to $dis[center][farthest\ node\ in\ k]$, a circular zone is determined. Every node inside this zone is included in the candidate set of that particular route, as seen in Figure 4.

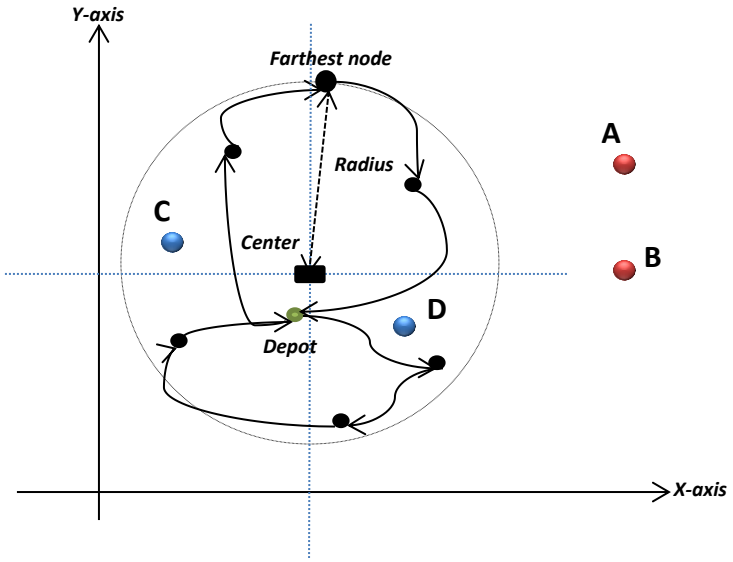

Figure 4. intllLNS circular zoning

As aforementioned, first approach of intllLNS considers cost indexes and it tries to add nodes from other routes or stack into targeted route. As explained, this set of targeted routes are marked from the top part of the list which this part is determined by a ratio between [0, 1] where it is fixed as an input parameter called $intllRate$.

The second approach of intllLNS considers idle times in a route. Idle time is defined as the time between actual reaching time of a vehicle to a node and the early time of that node. In fact, when a vehicle reaches a node at a time before the time window, it should wait until that time window opens. This time gap is considered as idle time. In this approach, all gaps of a route are cumulated and considered as idle time index and the maximum occurred gap along with its starting and ending points are considered as a label on that route. Then, the routes in a solution are sorted according to this index from top to down. All routes in the top part of the list have more idle times.

Same as the first approach, a set of candidate nodes is created. In this case, those nodes in which their time windows are inside the duration of the maximum gap of the route are included in the candidate set of that route. Finally, as same as first approach, intllLNS iteratively runs and repeats above operations until the maximum number of allowed iterations are met. According to above explanations, the pseudo code of the intllLNS is proposed as following in Figure 5.

Finally, after the intllLNS and adding from stack operations, the algorithm checks if the stack is empty or not. If stack is still not empty, then a new vehicle is added to the solution. A full procedure of OR_ENH_HEU rule for a single vehicle is implemented, in order to construct a complete feasible route, considering nodes in the stack. After this final step of the cycle, another attempt is done called $AddToAll$, which as mentioned before, tries to vacate the stack pool by testing all individual nodes in all active routes.

```
Step 1: For all routes calculate CostIndex(k);
                    CIOrder = SortRoutes (CostIndex);
Step 2: For all routes calculate IdleTimeIndex(k);
                    ITOrder = SortRoutes (IdleTime);
Step 3: If (rand
< PintllOrient) { For all routes in top of CIOrder SetCandid(CircleZone(k));
                                                    SelectRandNode(SetCand
                                                    checkMove(i, k);
                    intIter + +; }
Step 4: If (rand
≥ PintllOrient) { For all routes in top of ITOrder SetCandid(MaxGap(k));
                                                    SelectRandNode(SetCand
                                                    checkMove(i, k);
                    intIter + +; }
Step 5: If (intIter < MaxLNSIteration)
                        { If(IsStackEmpty){ AddfromStack (j, k); SubsStack(j, (i, k
                          goto Step 1; }
        else    end;
```

Figure 5. intllLNS Pseudocode

4. Computational Experiments

In this chapter of the paper, the performance of both developed mathematical model and HAOR_intLNSA heuristic algorithm is tested through experiments on well-known benchmark instances of Li and Lim(Li & Lim, 2008). Since the exact implementation of these instances would not fit to our introduced problem, some modifications have been made to them. Besides, minimum size of the instances found in the literature begins with the size of 100 nodes. Due to the resulted high complexity of AND/OR-MTVRPTW, Proposed MILP model would not be able to optimally solve these instance sizes in a reasonable computational time. Our experiments using MILP model show no optimal solution even after 3 days for an instance with 100 sizes in which at the best case, it could reach to 39% gap from the lower bound. Therefor these limitations make us to do some modifications on standard set of benchmarks of Li and Lim in the aspect of problem characteristics as well as problem sizes (Li & Lim, 2008). The details are clarified in following sub-section.

4.1. Data Generation

As it was mentioned, the minimum size found in standard PDPTW begins with 100 nodes, and we should note that in the PDPTW nodes are paired in a way that they cannot be served by separate vehicles. This fact itself reduces the complexity of the problem where it practically solves allocation of $N/2$ sets into k vehicles however in AND/OR-MTVRPTW the allocation of N sets into k vehicles are considered. Besides, dealing with multiple trips adds to the complexity of the base PDPTW problem. So, in order to evaluate the proposed model's efficiency, set of small sized instances are derived, using standard instances with 100 nodes from all categories of **LC_type_I**, **LC_type_II**, **LR_type_I**, **LR_type_II**, **LRC_type_I** and **LRC_type_II**.

At the first step of this operation, all pickup nodes which have same position with their delivery pairs are eliminated. For better understanding of this action, if we pay more attention to the instances of Li and Lim (Li & Lim, 2008), we will find that in the existing examples, there are pairs of nodes that practically both pickup and delivery have occurred at the same node. So, the purpose of elimination here is related to this kind of pairs and it is implemented on the pickup parts. As a result of this operation, some nodes will be left alone, without any precedence connection. Then, a set of random nodes with their belonging pairs will be selected until the number of nodes in the set reaches to the targeted size.

Afterwards, in order to build feasible OR arcs, for every AND designated arc from i to j $(i \rightarrow j)$ we give a 50 percent chance to change into OR arc. Then, all resulted OR candidates (if any) are partitioned into sets in which the maximum number of included pairs in a set is determined by an input parameter called $MaxPart$. Finally, from all delivery part of the paired nodes in a set, a single one is randomly selected as OR arc successor which all pickup nodes will be connected to that selected successor as its OR predecessors. And the remained other delivery nodes got relief with none precedence connection.

Consequently, by means of above described four step procedure, small sized instances are generated. Also, in order to generate larger sized instances, the first, third and the last steps of the proposed procedure are carried out. The entire trend of the used procedure is summarized in four steps as in Figure 6 and all generated instances are available in an attachment to this paper.

Step 1: for all pickup i and delivery j, if $\big(X(i) == X(j) \&\& Y(i) == Y(j)\big)\{$
eliminate(i); $\}$
Step 2: size $= 0$; PickRand(i, j); size $=$ size $+ 2$;
if $(size < SIZE)$ goto Step 2;
Step 3: for all $i \overrightarrow{AND} j$, if (rand(t) $< 50\%$) $ORsetCandid = (i \rightarrow j)$;
Step 4: for t from 1 to T, { Partition$(ORsetCandid, MaxPart)$;
DestinOR(t) $=$ rand $\big(delivery(j)\big)$;
for 1 to MaxPart if (is_AND_arc_exist i to j) creatOR $\Big(i \overrightarrow{OR}$ DestinOR(t)$\Big)$; }

Figure 6. Instance generating rule

In the following section, all results of the experiments are provided that begins with a small sample example with complete explanation for the problem and the optimal result.

4.2. Results and Discussions

The introduced problem of AND/OR-MTVRPTW is new, so, any exact coincide could be found neither in the aspect of problem structure nor in the aspect of solution algorithms. Therefore, the performance of the **HAOR_intLNSA** is evaluated via comparisons against optimal solution of the proposed mathematical model for the small sized instances. Six categories of problems are considered in which in total, 24 instances are derived by means of the rule explained in previous section. Furthermore, since there is not any gauge for testing the developed algorithm, it is decided to test

our algorithm by making some modification on Li and Lim (Li & Lim, 2008) well-known set of instances. The instances and the proposed algorithm's input parameters and conditions are arranged in a way that the final results could be compared in some aspects under the determined conditions. For this aim, at the first step, the demand requests are all set to a same positive unit number of 1. Because, there is no pickup and delivery services are considered in the introduced problem. Moreover, because the positive and negative values of the nodes' demands in a vehicle neutralize each other, practically, the capacities of vehicles are ineffective. Thus, in the generated large instances, the maximum capacity is set to a value equal to the maximum number of nodes allocated in a vehicle according to the best solution found in the PDPTW benchmark. Since the demands are all set to 1 that's why the number of nodes value is considered as the maximum capacity. The rest of information regarding the input parameters and conditions are provided along the tables of results.

In the following it is tried to more clarify the introduced problem's details and features by solving a small-sized sample instance. Then, the efficacy of the HAOR_intLNSA in small scales is testified through a set of comparisons against mathematical model's optimal results. Furthermore, the performance of algorithm is checked by setting a set of comparisons including instances with 100, 200 and 400 nodes in size. Then the obtained results are interpreted and the efficiency of the proposed methodology is discussed.

Suppose the following scattered nodes in $XY-plane$ with 10 customer nodes and one depot in Figure 7 and with input data presented in Table 2.

Table 2. Sample example data

Node	X	Y	Demand	EarlyTime	LateTime	ServiceTime
Depot	5	5	0	0	1000	0
A	5	6	1	1	5	1
B	6	6	1	3	7	1
C	4	4	2	1	3	1
D	8	8	2	8	13	1
E	3	3	3	10	20	1
F	12	12	1	10	20	1
G	4	3	1	3	7	1
H	7	6	1	5	8	1
I	8	9	1	13	20	1
J	10	10	2	7	15	1

Positions of nodes and corresponding precedence constraints arcs are depicted in Figure 7. For this sample instance maximum number of allowed trips is 3 and the capacity of all vehicles is assumed 3.

According to optimum solution achieved by mathematical model, the example is solved in total 5 trips. Three vehicles are used and all start their travels at the time zero. The first trip of first vehicle begins from depot, then the vehicle visits node A, B and H, next it returns to depot and then starts a further trip by visiting D, I and ends in depot. The second vehicle starts its first trip by visiting node C, then G and back to depot and continues its service by traveling to node J and F and finally ends travel in depot. The third vehicle carries out a single visitation of node E.

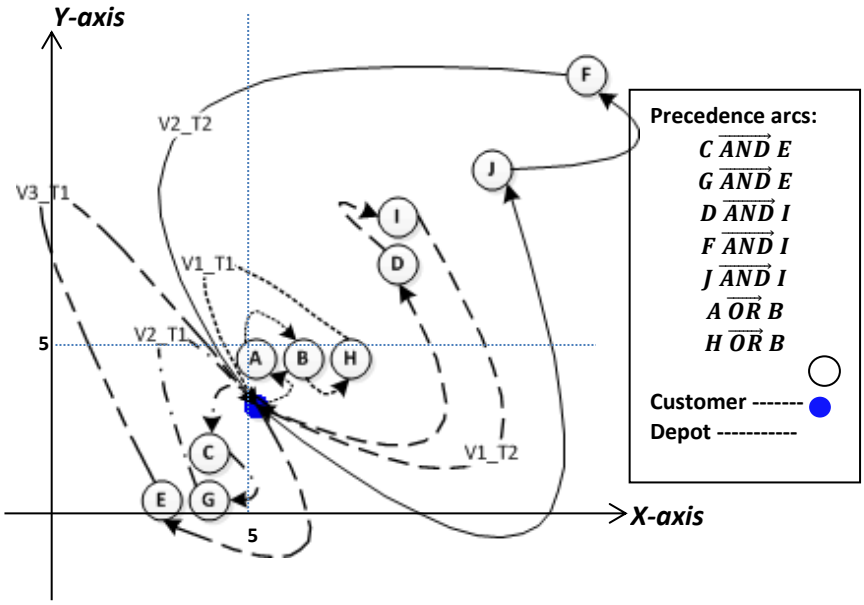

Figure 7. Solution of Sample Example

In this sample the concept of OR precedence relationship is clearly seen in V1_T1. First vehicle starts its trip by visiting A, then B, then node H, even though both nodes of A and H are the OR-type predecessors of node B. But since in the concept of OR-pc, the completion of only one of the predecessors are adequate to start the successor, this fact is implemented and node B is allowed to be visited after giving service to node A.

In V1_T2 we see that the node D is visited prior to node I as designated by an AND-type pc arc. The rest of precedence constraints have not met, because pairs have not assigned in the same vehicles.

In the following the generated small sized instances are solved by mathematical model and the HAOR_intLNSA. All six categories of the bench-mark are considered and in total, 24 instances are generated. Because of the high complexity of the problem, the most cases of these small instances could not be solved optimally by math model in reasonable time. So, beside of these instances a set of so smaller feasible instances are also created by authors that their optimality are solved and proved by the solver. Table 3 presents the detailed results of these experiments. It should be noted that the Mixed Integer Linear Model is written in C++ using Cplex 12.8 iLog Concert Technology. Experiments have done on a core i7 pc with 3.34 GHz CPU speed and 8 Mb of RAM.

4.2.1. Parameters Setting

According to the descriptions given about the procedure of the developed algorithm, there is a set of parameters that their values should be determined before the execution of the program. Due to this determination action, a full factorial design is implemented to extract the most effective and efficient parameter values. So, for each parameter a set of values are predetermined, evoked through initial test experiences. It is clear that we should take an instance(s) in order to apply the all

design of experiments to decide of the parameters' values. For this aim, we took two sample instances, one for small-size and the other as the representative for big-size instances. The size of the instances should be selected in such a way that their solving times are taken into account due to the high number of tests ($3 \times 2 \times 1 \times 3 \times 3 \times 2 \times 1 \times 1 \times 1 \times 2 = 216$). So, considering above mentioned conditions two instances with 16 and 100 nodes are selected for small and big size instances, respectively. Table 3 has given the set of parameters and their relevant values. The last two columns of Table 3 indicate decided value for each parameter for each group of instances.

Table 3. Parameters setting of the developed algorithm

Parameter	Description	Alternatives	Small-Size	Big-Size
Pop_size	Population size	{1,20,50}	20	1
Total_iteration	Total iteration of the algorithm	{100,500}	100	500
LNS_Iteration	Total iteration of LNS procedure	{100}	100	100
IntllRate	A portion of the targeted vehicles	{0.3, 0.5, 0.8}	0.3	0.3
Perturb_orient	A percentage to go to move or replace perturb	{30%, 50%, 80%}	50%	50%
Perturb_iter	Total iteration of perturbation	{100,500}	100	500
LNS_orient	A percentage to go to or replace perturb	{50%}	50%	50%
sub_stack	Number of add from stack with substitute target	{100}	100	100
R_Add_stack	Number of iterations to randomly add from stack	{100}	100	100
Acc_rate	Simulated annealing rate in which solutions with this difference from earlier solution would be accepted	{1.2,1.5}	1.2	1.2

Making use of above arrangements the algorithm and the mathematical models are executed and obtained results are concluded in Tables 4 to 8.

Table 4. Small-size instances results

Size	Max_vehicl	Max_trip	Capacity	No.OR-arcs	No.AND-arcs	Obj_cplex	Time(s)	Vehicle	Trips	Obj_HAOR_intLNSA	No.Vehicle	No.trip	Time(s)	Gap
7	6	3	4	2	2	28.2843	1	2	2	28.2843	2	2	10	0%
8	6	3	4	3	2	36.0328	1	2	2	36.0328	2	2	7	0%
9	6	3	4	2	3	53.8195	1	2	3	53.8195	2	3	15	0%
10	6	3	4	4	2	48.3182	2	2	3	48.3182	2	3	12	0%
11	6	3	4	2	5	39.7498	6	3	5	39.7498	3	5	10	0%
12	6	3	4	2	5	82.0936	5	3	5	82.0936	3	5	11	0%
13	6	3	4	2	5	127.327	55	4	6	127.327	4	6	20	0%
14	6	3	4	4	5	193.604	239	4	7	193.604	4	7	18	0%
15	6	3	4	5	5	196.379	212	4	6	196.379	4	6	20	0%
16	6	3	4	6	5	346.967	225	5	8	346.967	5	8	24	0%

Instances in Table 4 are generated in a complete random manner with aim of producing feasible problems by authors. As shown in Table 4, mathematical model could reach to optimum solutions for all of these 10 instances. As seen the computational times have started to grow from the instance with size of 13. The efficiency of the proposed algorithm is clearly proved due to its performance on capability to optimum results in a small amount of computational time. Once the size of the problems grows to 21 nodes, it has come out that the mathematical model solver got in trouble and could not reach to optimality for a considerable portion of the instances, however, the proposed algorithm has resulted in reliable results.

Table 5. Results for instances with medium size

Instance_size	Max_vehicle	Max_trip	Capacity	No.OR-arcs	No.AND-	Best_found_Obj	Opt_Gap	Time(s)	Vehicle	Trips	Obj_HAOR_intLNSA	No.Vehicle	No.trip	Time(s)	Gap
LC101_21	6	3	4	6	4	375.17	0%	41	3	6	375.17	3	6	64	0
LC102_27	6	3	4	7	5	603.50	40%	14430	5	-	483.23	4	8	70	-19.9%
LC103_23	6	3	4	3	7	622.83	24.99%	68262	4	-	602.4	4	6	56	-3.2%
LC104_21	6	3	4	3	6	524.30	33.34%	4078	3	-	408.68	3	5	57	-22.1%
LC201_21	6	3	4	5	5	501.40	0%	21	2	6	503.04	2	6	69	0.3%
LC202_21	6	3	4	5	5	436.35	33.29%	3836	3	-	477.54	2	6	54	9.4%
LC203_21	6	3	4	7	3	450.54	0.5%	10493	2	-	458.21	2	6	68	1.7%
LC204_21	6	3	4	6	4	457.74	0%	22324	2	5	457.74	2	5	40	0%
LR101_21	6	3	4	3	7	459.84	0%	7	6	6	459.84	6	6	58	0%
LR102_21	6	3	4	5	4	474.95	0%	892	4	6	474.95	4	6	24	0%
LR103_21	6	3	4	3	7	-	-	4309	-	-	532.77	3	7	65	**
LR104_21	6	3	4	3	7	-	-	4440	-	-	520.21	3	8	37	**
LR201_21	6	3	4	3	7	417.43	0%	447	2	5	417.43	2	5	70	0%
LR202_21	6	3	4	4	6	510	0.06%	4986	2	-	516.78	2	6	26	1.3%
LR203_21	6	3	4	4	6	557.4	0.1%	3662	2	-	558.82	2	6	34	0.2%
LR204_21	6	3	4	5	5	488.91	0.07%	4637	2	-	486.41	2	6	69	-0.5%
LRC101_21	6	3	4	5	5	472.23	0%	259	5	5	472.23	5	5	43	0%
LRC102_21	6	3	4	6	4	-	-	3673	-	-	468.88	5	5	48	**
LRC103_21	6	3	4	6	4	677.50	33.36%	3646	3	-	492.14	3	7	61	-27.3%
LRC104_21	6	3	4	5	5	440.87	59.96%	57041	5	-	467.32	3	8	38	5.9%
LRC201_21	6	3	4	3	7	507.53	0%	1895	2	5	507.53	2	5	53	0
LRC202_21	6	3	4	2	8	600.05	0.1%	3666	2	-	600.05	2	6	116	0
LRC203_21	6	3	4	5	5	672.23	33.36%	4807	3	-	605.56	2	6	32	-9.9
LRC204_21	6	3	4	4	6	586.91	0.12%	8840	2	-	589.73	2	6	59	0.4

Results in Table 5 show a complete superiority of the developed algorithm in comparison with exact mathematical model since it obtained better results for majority part of the instances. Three rows of Table 5 are in grey color. It should be noted that the mathematical results show difference from lower bound considering a

cost value for every vehicle used. For an example in case LC_202_21 it seems that the best found traveling cost function of mathematical model is better than the result of algorithm, however, we should note that the solution found by the algorithm resulted in lower number of vehicles. So to better evaluate the outcome, the numbers of the resulted used vehicles are also should be taking into account while evaluating the efficiency of the algorithm.

Table 6. HAOR_intLNSA results for instances with size 100

Instance_Size	NO. Arcs	Max_vehicle	Max_trip	Capacity	Obj_HAOR_intLNSA	No.Vehicle	Time(s)
LC101_100	47	10	3	14	881.65	6	148
LC102_100	47	10	3	14	893.19	7	93
LC103_100	48	9	3	14	1022.56	4	119
LC104_100	47	9	3	14	780.77	4	80
LC201_100	49	3	2	36	525.84	2	76
LC202_100	49	3	2	36	568.47	2	104
LC203_100	49	3	2	36	608.74	2	118
LC204_100	49	3	2	36	545.37	2	97
LR101_100	47	19	3	8	1576.68	13	140
LR102_100	45	17	3	8	1203.11	11	78
LR103_100	48	13	3	8	1375.58	6	92
LR104_100	48	9	3	8	890.75	4	121
LR201_100	49	4	2	28	1430.57	3	126
LR202_100	50	3	2	38	1078	2	122
LR203_100	49	3	2	47	899.46	2	80
LR204_100	50	2	2	51	906.82	2	66

As described before three instance groups are generated in order to evaluate the performance of proposed algorithm. The results of all three are gathered in Tables 6-8. As described before, the arcs are produced and implemented in the problems. The total number of all AND and OR arcs are noted in the tables in front of each instance case. The maximum vehicle number for each instance is determined considering the best minimum found so far in the literature for the classic form of the problem. and the total maximum allowed trips are considered randomly from the set of {2,3} giving more chance to value 3 (80%) for instances with max_vehicle more than 10 and giving more chance for value 2 for instances with max_vehicle number lower than 10.

For each instance, best traveling cost objective value with obtained minimum number of vehicles are noted. Also, the computational time for each instance is also calculated in seconds and depicted in a separate column.

Table 7. HAOR_intLNSA results for instances with size 200

Instance_size	No. Arcs	Max_vehicle	Max_trip	Capacity	Obj_HAOR_intLNSA	No.Vehicle	Time(s)
LC101_200	94	20	3	14	2456.98	14	296
LC102_200	95	19	3	14	2555.21	14	240
LC103_200	97	17	3	14	2870.66	12	223
LC201_200	98	6	2	36	1796.59	5	299
LC202_200	98	6	2	36	1840.47	5	266
LC203_200	99	6	2	36	1867.14	5	226
LR101_200	95	20	3	12	4400.31	16	232
LR102_200	95	17	3	16	4733.81	13	214
LR103_200	96	14	3	18	4179.17	11	264
LR201_200	99	5	2	30	4101.25	5	223
LR202_200	99	4	2	54	3711.68	4	238
LR203_200	99	4	2	54	2846.28	4	238
LRC1_301_200	94	19	3	12	3369.57	16	248
LRC1_302_200	97	15	3	20	3559.85	12	284
LRC1_303_200	95	13	3	24	3325.64	11	219

Since the problem in its current form is accounted as a novel class of problem, so making comparisons with other authors proposed algorithms seems to be not suite for this case. Even re-coding of the algorithms developed in the history of VRP problems would not be a good choice to show the capability of the proposed algorithm. In this research authors did their best to make use of the methodologies which are derived from the most recent achievements in literature of the problem. Looking the body of the algorithm shows this fact that in the perturb process or simple LNS or simple feasibility check process we first applied the most used and claimed functions, then we did improvements and made considerable enhancements in each part. Also by introducing the new procedure for LNS we could obtain better results by eliminating time consuming recent procedures. Considering all of these there is one way to show the capability and efficiency of our proposed algorithm. By using predetermined values for capacity and maximum number of vehicles considering the bench mark and the data given in the Li and Lim (Li & Lim, 2008). The resulted traveling costs show a much about the performance of the algorithm. For example we took a look for the traveling costs of the instances in their classic form of the third group of instances with size 400. As it is seen, obtained traveling costs by HAOR_intLNSA for the major part of instances are better than what were resulted before. Actually this is not a fair comparison since the form of precedence relations are changed and also trips options are added in these new instances which all lead to more alternatives. But it can at least show that HAOR_intLNSA is reliable algorithm for the introduced class of problem since it could reach to results as they are anticipated by widening the feasible space.

This problem is new and very attractive with challengeable several aspects from developing exact algorithms to constructing search based fast solution techniques. And also it has resulted in a set of problems with very high complexity. Thus, authors

invite researchers and authors to do studies on this problem since it has lot to grow and progress.

Table 8. HAOR_intLNSA results for instances with size 400

Instance_size	No. Arcs	Max_vehicle	Max_trip	Capacity	Obj_HAOR_intLNSA	No.Vehicle	Time(s)	Travel_cost(PDPTW)
LC101_400	189	40	3	14	6805.55	29	370	7152.06
LC102_400	189	38	3	14	8148.32	26	310	8007.79
LC103_400	190	32	3	14	8264.86	25	345	8678.23
LC201_400	197	12	3	38	4213.38	8	253	4116.33
LC202_400	198	12	3	38	4071.44	8	340	4144.29
LC203_400	197	12	3	38	4024.43	8	239	4401.08
LR101_400	192	40	3	14	11032.72	32	396	10639.75
LR102_400	191	30	3	17	10800.66	24	251	11009.51
LR103_400	192	22	3	28	8763.18	17	245	9251.13
LR201_400	199	8	3	50	8601.78	8	298	9726.88
LR202_400	199	7	3	50	9233.86	7	246	9405.40
LR203_400	198	5	3	50	9687.67	5	275	10282.01
LRC1_301_400	192	36	3	15	8902.64	28	359	9124.52
LRC1_302_400	191	31	3	15	8222.36	29	299	8346.06
LRC1_303_400	194	24	3	18	7736.29	23	327	7805.16

5. Conclusions and future research

In this paper, AND/OR precedence constraints have introduced in the field of vehicle routing problems with the aim of minimizing total traveling distances and costs. This type of precedence relations has not been considered so far in the literature of VRPs. Although the nature of this type of precedence relations exists in the body of problem but in the real cases, it has ignored by the researchers. Therefore, this paper introduces this extension to that former problem for the first time as well as proposing a practical general hybrid optimization algorithm which is able to solve the medium and large size problems. By this additional OR-type precedence constraints to the classical 'AND' type PC, it is not needed to visit all predecessors of a successor node before it can be met, and finishing one of them can let to that particular successor to get started. This paper implemented this type of PC graph in VRP for the first time in the related literature. The problem was mathematically formulated. Since VRPs are known as NP-hard, even in simple versions, our more complicated problem was also NP-hard. Therefore, a decomposition based heuristic method was employed to solve the problem. Indeed, the routing part was enhanced by heuristics to cover AND/OR PC graphs. The computational experiments on several problem instances with different sizes demonstrated the efficiency of proposed solution method in terms of solution quality.

Because of the novel nature of the problem, it is recommended two future research guidelines: on one hand, this work can be proceeded with other types of VRPs such as

VRP with Backhauls and open VRP; on the other hand, the application of other heuristic/meta-heuristic algorithms may lead the problem to the better solutions.

References

Alvarez, A., & Munari, P. (2017). An exact hybrid method for the vehicle routing problem with time windows and multiple deliverymen. *Computers & Operations Research*, *83*, 1-12. https://doi.org/10.1016/j.cor.2017.02.001

Anggodo, Y. P., Ariyani, A. K., Ardi, M. K., & Mahmudy, W. F. (2017). Optimization of multi-trip vehicle routing problem with time windows using genetic algorithm. *Journal of Environmental Engineering and Sustainable Technology*, *3*(2), 92-97. https://doi.org/10.21776/ub.jeest.2017.003.02.4

Ariyani, A. K., Mahmudy, W. F., & Anggodo, Y. P. (2018). Hybrid Genetic Algorithms and Simulated Annealing for Multi-trip Vehicle Routing Problem with Time Windows. *International Journal of Electrical & Computer Engineering (2088-8708)*, *8*(6). http://doi.org/10.11591/ijece.v8i6.pp4713-4723

Azi, N., Gendreau, M., & Potvin, J. Y. (2010). An exact algorithm for a vehicle routing problem with time windows and multiple use of vehicles. *European Journal of Operational Research*, *202*(3), 756-763. https://doi.org/10.1016/j.ejor.2009.06.034

Bae, H., & Moon, I. (2016). Multi-depot vehicle routing problem with time windows considering delivery and installation vehicles. *Applied Mathematical Modelling*, *40*(13-14), 6536-6549. https://doi.org/10.1016/j.apm.2016.01.059

Cetin S., Gencer C. (2010). Vehicle routing problems with hard time windows and simultaneous pickup and delivery: a mathematical model. J. Fac. Eng. Arch. Gazi Univ. 25(3): 579–585]. https://doi.org/10.4236/jss.2015.33008

Chen, C. P. (1989, November). Precedence knowledge acquisition for generating robot assembly sequences. In Conference Proceedings., IEEE International Conference on Systems, Man and Cybernetics (pp.71-76).IEEE. https://doi.org/10.1109/icsmc.1989.71255

Chen, C. P. (1990, January). Planning optimal precedence-constraint robot assembly sequences problem with neural computation. In *Applications of Artificial Intelligence VIII* (Vol. 1293, pp. 320-331). International Society for Optics and Photonics. https://doi.org/10.1117/12.21080

Chen, C. P. (1990, June). Neural computation for planning AND/OR precedence-constraint robot assembly sequences. In *1990 IJCNN International Joint Conference on Neural Networks* (pp. 127-142). IEEE. https://doi.org/10.1109/ijcnn.1990.137557

Chiang, T. C., & Hsu, W. H. (2014). A knowledge-based evolutionary algorithm for the multi objective vehicle routing problem with time windows. *Computers & Operations Research*, *45*, 25-37. https://doi.org/10.1016/j.cor.2013.11.014

Chunyu, R., & Xiaobo, W. (2010, October). Research on multi-vehicle and multi-depot vehicle routing problem with time windows for electronic commerce. In *2010 international conference on artificial intelligence and computational intelligence* (Vol. 1, pp. 552-555). IEEE. https://doi.org/10.1109/aici.2010.121

Coelho, V. N., Grasas, A., Ramalhinho, H., Coelho, I. M., Souza, M. J., & Cruz, R. C. (2016). An ILS-based algorithm to solve a large-scale real heterogeneous fleet VRP with multi-

trips and docking constraints. *European Journal of Operational Research*, *250*(2), 367-376. https://doi.org/10.1016/j.ejor.2015.09.047

Cömert, S. E., YAZGAN, H. R., Sertvuran, I., & ŞENGÜL, H. (2017). A new approach for solution of vehicle routing problem with hard time window: an application in a supermarket chain. *Sādhanā*, *42*(12), 2067-2080. https://doi.org/10.1007/s12046-017-0754-1

Dabia, S., Ropke, S., Van Woensel, T., & De Kok, T. (2013). Branch and price for the time-dependent vehicle routing problem with time windows. *Transportation science*, *47*(3), 380-396. https://doi.org/10.1287/trsc.1120.0445

De Fazio, T., & Whitney, D. (1987). Simplified generation of all mechanical assembly sequences. *IEEE Journal on Robotics and Automation*, *3*(6), 640-658. https://doi.org/10.1109/jra.1987.1087132

Dong, W., Zhou, K., Qi, H., He, C., & Zhang, J. (2018). A tissue P system based evolutionary algorithm for multi-objective VRPTW. *Swarm and evolutionary computation*, *39*, 310-322. https://doi.org/10.1016/j.swevo.2017.11.001

Errico, F., Desaulniers, G., Gendreau, M., Rei, W., & Rousseau, L. M. (2018). The vehicle routing problem with hard time windows and stochastic service times. *EURO Journal on Transportation and Logistics*, *7*(3), 223-251. https://doi.org/10.1007/s13676-016-0101-4

Ghoseiri, K., & Ghannadpour, S. F. (2010). Multi-objective vehicle routing problem with time windows using goal programming and genetic algorithm. *Applied Soft Computing*, *10*(4), 1096-1107. https://doi.org/10.1016/j.asoc.2010.04.001

Gillies, D. W., & Liu, J. S. (1990, December). Scheduling tasks with and/or precedence constraints. In *Proceedings of the Second IEEE Symposium on Parallel and Distributed Processing 1990* (pp. 394-401). IEEE. https://doi.org/10.1109/spdp.1990.143572

Gillies, D. W., & Liu, J. W. S. (1995). Scheduling tasks with AND/OR precedence constraints. *SIAM Journal on Computing*, *24*(4), 797-810. https://doi.org/10.1137/s0097539791218664

Goel, R., & Maini, R. (2021). Improved multi-ant-colony algorithm for solving multi-objective vehicle routing problems. *Scientia Iranica*, *28*(6), 3412-3428. https://doi.org/10.24200/sci.2019.51899.2414

Golden, B., Assad, A., Levy, L., & Gheysens, F. (1984). The fleet size and mix vehicle routing problem. *Computers & Operations Research*, *11*(1), 49-66. https://doi.org/10.1016/0305-0548(84)90007-8

Golden, Bruce L., Subramanian Raghavan, Edward A. Wasil, eds (2008). *The vehicle routing problem: latest advances and new challenges*. Vol. 43. Springer Science & Business Media. https://doi.org/10.1007/978-0-387-77778-8

Gordon, V., Potapneva, E., & Werner, F. (1997). Single machine scheduling with deadlines, release and due dates. *Optimization*, *42*(3), 219-244. https://doi.org/10.1080/02331939708844360

Hernandez, F., Feillet, D., Giroudeau, R., & Naud, O. (2014). A new exact algorithm to solve the multi-trip vehicle routing problem with time windows and limited duration. *4or*, *12*(3), 235-259. https://doi.org/10.1007/s10288-013-0238-z

Hernandez, F., Feillet, D., Giroudeau, R., & Naud, O. (2016). Branch-and-price algorithms for the solution of the multi-trip vehicle routing problem with time

windows. *European Journal of Operational Research*, *249*(2), 551-559. https://doi.org/10.1016/j.ejor.2015.08.040

Hu, C., Lu, J., Liu, X., & Zhang, G. (2018). Robust vehicle routing problem with hard time windows under demand and travel time uncertainty. *Computers & Operations Research*, *94*, 139-153. https://doi.org/10.1016/j.cor.2018.02.006

Jiang, J., Ng, K. M., Poh, K. L., & Teo, K. M. (2014). Vehicle routing problem with a heterogeneous fleet and time windows. *Expert Systems with Applications*, *41*(8), 3748-3760. https://doi.org/10.1016/j.eswa.2013.11.029

Koç, Ç., Bektaş, T., Jabali, O., & Laporte, G. (2015). A hybrid evolutionary algorithm for heterogeneous fleet vehicle routing problems with time windows. *Computers & Operations Research*, *64*, 11-27. https://doi.org/10.1016/j.cor.2015.05.004

Kumar, S. N. & Panneerselvam, R. (2015). A time-dependent vehicle routing problem with time windows for e-commerce supplier site pickups using genetic algorithm. Intelligent Information Management, 7, 181-194. https://doi.org/10.4236/iim.2015.74015

Li & Lim (2008) benchmark web page www.sintef.no/projectweb/top/pdptw/li-lim-benchmark.

Le Bouthillier, A., & Crainic, T. G. (2005). A cooperative parallel meta-heuristic for the vehicle routing problem with time windows. *Computers & Operations Research*, *32*(7), 1685-1708. https://doi.org/10.1016/j.cor.2003.11.023

Lee, S., Moon, I., Bae, H., & Kim, J. (2012). Flexible job-shop scheduling problems with 'AND'/'OR'precedence constraints. *International Journal of Production Research*, *50*(7), 1979-2001. https://doi.org/10.1080/00207543.2011.561375

Mingozzi, A., Roberti, R., & Toth, P. (2013). An exact algorithm for the multitrip vehicle routing problem. *INFORMS Journal on Computing*, *25*(2), 193-207. https://doi.org/10.1287/ijoc.1110.0495

Miranda, D. M., & Conceição, S. V. (2016). The vehicle routing problem with hard time windows and stochastic travel and service time. *Expert Systems with Applications*, *64*, 104-116. https://doi.org/10.1016/j.eswa.2016.07.022

Möhring R.H., Skutella M., Stork F. (2004). Forcing relations for AND/OR precedence constraints. SIAM Journal on computing. http://dx.doi.org/10.14279/depositonce-14708

Möhring, R. H., Skutella, M., & Stork, F. (2004). Scheduling with AND/OR precedence constraints. *SIAM Journal on Computing*, *33*(2), 393-415. https://doi.org/10.1137/s009753970037727x

Nazif H. and Lee L. S. (2010). Optimized crossover genetic algorithm for vehicle routing problem with time windows. American Journal of Applied Sciences, 7(1), 95-101. https://doi.org/10.3844/ajassp.2010.95.101

Parragh, S. N., & Cordeau, J. F. (2017). Branch-and-price and adaptive large neighborhood search for the truck and trailer routing problem with time windows. *Computers & Operations Research*, *83*, 28-44. https://doi.org/10.1016/j.cor.2017.01.020

Pierre, D. M., & Zakaria, N. (2017). Stochastic partially optimized cyclic shift crossover for multi-objective genetic algorithms for the vehicle routing problem with time-

windows. *Applied Soft Computing*, *52*, 863-876. https://doi.org/10.1016/j.asoc.2016.09.039

Qi, X., Yu, G., & Bard, J. F. (2002). Single machine scheduling with assignable due dates. *Discrete Applied Mathematics*, *122*(1-3), 211-233. https://doi.org/10.1016/s0166-218x(01)00316-x

Toth, P., & Vigo, D. (2002). Models, relaxations and exact approaches for the capacitated vehicle routing problem,Discrete Applied Mathematics, 123(1–3), 487-512 . https://doi.org/10.1016/S0166-218X(01)00351-1

Wang, Y., Ma, X., Xu, M., Liu, Y., & Wang, Y. (2015). Two-echelon logistics distribution region partitioning problem based on a hybrid particle swarm optimization–genetic algorithm. *Expert Systems with Applications*, *42*(12), 5019-5031. https://doi.org/10.1016/j.eswa.2015.02.058

Zhang, D., Cai, S., Ye, F., Si, Y. W., & Nguyen, T. T. (2017). A hybrid algorithm for a vehicle routing problem with realistic constraints. *Information Sciences*, *394*, 167-182. https://doi.org/10.1016/j.ins.2017.02.028