Single depot heterogeneous capacitated vehicle routing problem with simultaneous delivery and pickup for disaster management systems

Santanu Banerjee [1a], Soumen Atta [2b], Goutam Sen[3a]

[a] *Department of Industrial and Systems Engineering, Indian Institute of Technology Kharagpur, , West Bengal, 721302, India*
[b] *Laboratoire des Sciences du Numérique de Nantes (LS2N), UMR 6004, UFR Sciences et Techniques, Université de Nantes, IMT Atlantique, 4 Rue Alfred Kastler, Nantes, 44307, France*

Abstract

During pre-disaster situations when there is a need for evacuation and supply dispatch for rationing to areas with high warnings of potential danger, tasks need to be performed simultaneously and both resources of time and money need to be managed. We consider the problems where cost management of pre-disaster emergency funds needs to be done with a considerable warning time available for performing a complete emergency operation. Mathematical formulations are developed for the single depot multiple heterogeneous capacitated vehicle routing problem with simultaneous delivery and pickup for disaster management systems. Here, the road networks are allowed to be specific to each vehicle type (based on the vehicle's width or other factors). The number of vehicles of each type is considered to be known in advance. Heuristic flavors for obtaining good solutions to the problem are developed and compared with an exact commercial solver; the results of which indicate that it is able to provide good feasible solutions for very large problems, whereas for smaller instances, the commercial solver provides much better results after a considerable amount of time. Four exact formulations are further compared we introduce the idea of redundant constraints for the first time which is shown to provide faster solutions irregularly. The algorithms

[1]E-mail: santanu@kgpian.iitkgp.ac.in
[2]Corresponding Author E-mail: soumen.atta@univ-nantes.fr, soumen.atta@ls2n.fr
[3]E-mail: gsen@iem.iitkgp.ac.in

are applicable to problems in which each customer may have either a delivery to be performed or a pick-up demand as well as when they may require both kinds of operation which would be done sequentially during the stoppage. Previously existing data as well as novel generated data-sets were used to compare four exact formulations with seven heuristic flavors obtaining better results than reported previously.

Keywords: OR in Disaster Relief, Multiple Vehicle Heterogeneous Fleet, Single Depot Vehicle Routing Problem, Simultaneous Delivery and PickUp, Machine Tuning with Jelly Fishing

1. Introduction

Delay in intelligent integration of infrastructure or pre-planning in developing nations have led to calamities being made of natural disasters enhanced by climate change. These seems to be tackled generally after they strike. Prediction and pre-disaster evacuation is essential for reducing the potential damages and much research is being done in this regard to upgrade and install new resilient infrastructure. It is essential that swift evacuation (pre or post naturally occurring disasters) is carried out alongwith delivery of relief materials and supplies. For this specific task, National Disaster Response Force[4] has been set up in India which is the backbone of providing humanitarian aid. A major problem during such emergency evacuation-cum-relief operation is efficient management of time and other available resources especially that of the disaster relief funds. It is especially important to cautiously gauge the existing financial constraints and strategically plan for pre-disaster evacuation. The decision of sending a particular type of vehicle to a specific relief point with adequate resources may not be intuitive and needs to be calculated based on whether we wish to minimize the time, cost or other direct or indirect factors. For this specific case, we model a real-world scenario assuming that the resources and vehicles are together present at the same depot which is also the final destination for bringing back evacuees. Efficient VRP algorithms would be combined within key decision making processes while conducting emergency operations and therefore fast heuristics are necessary to provide very good solutions. Such algorithms would be integrated within key Decision Support Systems enabling authorities to save time while

[4]https://ndrf.gov.in/

making decisions in the emergency operation theatres, which has the potential of saving multiple lives. The resulting problem is that of delivering goods which would consist of essential relief materials, water, clothes, food, medicines and other supplies, and then pickup humans (sick or affected victims) in need of urgent assistance within multiple vehicles which may be of different types. Each vehicle type may have any number of identical vehicles with limited capacities which may visit any node atmost once and would be returning to the same depot with the evacuees. We don't take into account the ergonomic challenges faced by the evacuees during the evacuation in on and leave the compatibility issues of vehicle types with human evacuees to be a future research area. Here, we consider the the objective as minimizing the fixed and variable costs enabling intelligent management of disaster specific funds during a pre-disaster phase considering adequate time being available for conducting the entire operation. Additional regulations to minimize the delivery time of each relief material as well as minimize the pick-up to drop-off time of each evacuee must also be considered during practical implementation of such algorithms.

We develop a compact exact formulation for the vehicle routing problem with simultaneous pickup and delivery with heterogeneous vehicles having compatibility with road networks. We find redundant constraints in previously developed formulation (Avci and Topaloglu, 2016) and try to understand their uility in reducing the solution-time (which may be researched further especially in the case of open-sourced solvers). We compare our formulations with Avci and Topaloglu (2016) and Keçeci et al. (2021). We develop the ATSAlgorithmHeuristic which use multiple logics to choose best routes. This choice is then made to progressively improve in ATSA-MachineTuning using multiple parameters tuned by the algorithm based on the toughness of the solution found. This is further improved by taking inspiration from the locomotion strategy of immortal jellyfishes which use their umbrella-shaped hoods or bells which pulsate and propel water. We identify that this process of squeezing their hood/bell margin may be introduced to relax the tightening standard deviation in the ATSA-MT-JellyFishing.

This paper is structured as follows: Section 2 provides a literature review highlighting existing taxonomy for classifying VRPs and foreseeing the need for development of a formula-based taxonomy (similar to what is used to represent formulas of chemical compounds); Section 3 presents a mathematical formulation of the described problem and highlights the role of redundant constraints to potentially decrease the exact solution's time; Section 4

explains the developed ATSAHeuristic and the 3 flavors of the ATSA-MT Heuristic and differentiates them with existing heuristics developed by Avci and Topaloglu (2016); Section 5 provides a comparative analysis based on computational results and observations are drawn from the same. A new improved version of the Heuristic inspired by the natural locomotion (Squeeze and Relax phases) of a JellyFish is also introduced. Finally, we conclude in Section 6 highlighting the gaps for future research avenues.The full forms of all acronyms used in this paper may be found in Table 12.

2. Related works

Gendreau et al. (1999) considered the TSP with pickup and delivery constraining the number of visits of the single vehicle at each customer to one, and minimized the total length of the route using Tabu Search and Local Search improvements. Du et al. (2021) considers a Simultaneous PickUp and Delivery TSP minimizing the total cost consisting of back order, lost sale and travel time. Since they consider the multi-trip TSP, this may be deemed identical to the homogeneous VRPSDP in case both minimize the total distance travelled or a total cost.

After the VRP was formally brought into existence in the present era by Dantzig and Ramser (1959), Min (1989) for the first time considered the problem of simultaneous delivery and pickup minimizing the total travel time of a vehicle's route considering traffic congestion factors. Salhi and Nagy (1999) develop a cluster insertion based heuristic considering both the simultaneous and mixed cases of VRPPD/VRPB adapting it to the case of multiple depots. They further consider single as well as multiple depot VRPs with pickups and deliveries (VRPPD) and develop composite heuristics for both the simultaneous (VRPSDP) as well as mixed (VRPPD) cases in (Nagy and Salhi, 2005). An insertion based heuristic with four different insertion criterias for the capacitated VRPSDP was also considered by (Dethloff, 2001) for homogeneous vehicles. Montané and Galvão (2002) proposed an exact formulation of the homogeneous VRPSDP for aerial (helicopter) transportation of individuals to oil exploration production platforms located in the Campos Basin in the state of Rio de Janeiro by extending classical VRP algorithms of tour partitioning and grouping, while also highlighting the TSP with pick-up and delivery mentioned in (Mosheiov, 1994) which strengthens a cheapest feasible insertion heuristic. Vural (2003) used Genetics Algorithms to solve the VRPSDP for the first time using meta heuristics.

Achuthan et al. (2003) proposed an improved branch-and-cut algorithm for a single-depot capacitated VRP minimizing the total distance travelled by all vehicles. A VRPSDP problem allowing multiple visits of homogeneous vehicles to individual nodes was developed by (Mitra, 2005). Montané and Galvão (2006) discuss about formulations with and without maximum distance constraints and use a Tabu Search to minimize the total travelling distance, at each step selecting from among multiple types of movements to obtain inter and intra-route adjustments. A two-index commodity-flow model studying the application of branch-and-price techniques was experimented on benchmark instances by (Dell'Amico et al., 2006). Chen and Wu (2006) sequentially construct homogeneous vehicular routes using insertion based procedure to develop initial solutions and propose hybrid schemes to improve routes using tabu lists and record-to-record travel, minimizing the total distance to service all customers. Bianchessi and Righini (2007) consider a range of heuristics for the VRPSDP combining local search and tabu search with complex and variable neighbourhood schemes. Zhang et al. (2007) uses ant-colony optimization to solve a single depot VRPSDP with homogeneous capacity constrained vehicles while also limiting maximum distance. A variable neighbourhood descent was proposed by (Subramanian et al., 2007) where they used a greedy randomised adaptive search. Ai and Kachitvichyanukul (2009) develop a particle swarm optimization to minimize fixed and variable costs of routes made by homogeneous vehicles. Gajpal and Abad (2009) propose an ant colony optimization with construction rules and multi-route local search schemes to solve a homogeneous capacitated VRPSDP and go on to develop a saving-based heuristic in (Gajpal and Abad, 2010) by merging existing routes using cumulative net-pickup to find this performing better than insertion-based heuristics. Zachariadis and Kiranoudis (2011) uses local-search to explore the solution space with varying aspiration criteria while examining rich neighbourhood structures for the VRPSDP consisting of vehicles with the same capacity. Alabas-Uslu and Dengiz (2011) develop a self-adaptive local search to solve a VRP consisting of capacity constrained homogeneous vehicles while considering only the delivery problem. Fard and Akbari (2013) considered a hybrid tabu search algorithm while constraining each tour's time considering capacity constrained homogeneous vehicles for the VRPSDP. Goksal et al. (2013) proposed a hybrid algorithm combining discrete particle swarm optimization with variable neighbourhood search, preserving the swarm diversity using an annealing-like strategy and implement novel giant tours to represent VRPSDP solutions. They further

highlight the first usage of giant tours by (Prins, 2004) in his hybrid genetic algorithm to minimize total cost of the trips in a capacitated VRP performed by identical vehicles. Such giant tour representations were further applied in heterogeneous VRP solutions. Subramanian et al. (2013) use dynamic programming and develop a branch-cut-and-price approach to minimize total travel costs, improving lower bounds of benchmark instances that consider a fleet of identical vehicles. Johal (2014) develop a new three-phase method to solve the multi-objective VRPSDP by construcing a set of diverse partial solutions, determining assignment possibilities for each sub-problem; and solving the sub-problems using parallel genetic algorithms having genetic operator switching mechanisms, an accuracy analysis tool and fitness evaluation mechanisms. A summary of the VRPSDP solution techniques with algorithmic features and strategies used may be found in (Kalayci and Kaya, 2016) in which they hybridize an ant-colony system for long-term memory structure empowered with a variable-neighborhood search for intense local search.

Koch et al. (2018) for the first time considered VRPSDP with three dimensional loading constraints and time windows by developing a hybrid heuristic. Barma et al. (2019) developed an AntLion Heuristic to minimize the total routing distance in a multi-depot VRP with homogeneous capacity constrained vehicles only considering the delivery problem. A VRPSDP is considered in (Hornstra et al., 2020) consisting of capacity constrained homogeneous vehicles, with handling costs and use adaptive large neighbourhood search, testing new policies for handling decisions. Christopher et al. (2021) studied the variable neighbourhood descent and tabu search algorithm to minimize the total distance utilizing an insertion heuristic and improving the customer's position using the neighbourhood operator. Ning et al. (2021) study a VRPSDP to reduce carbon emissions and improve de-carbonization and use a multi-phase quantum particle swarm optimization combining the advantages of parallelism and superposition consisting of homogeneous vehicles. Utama et al. (2022) solve the fuel consumption capacity vehicle routing problem (FCCVRP) minimizing fuel consumption costs using a hybrid jellyfish (HJF) algorithm to generate optimal fuel consumption through population and iteration parameters while also highlighting their study's limitation of not accounting for the pick-up load at each node. Agarwal and Venkateshan (2022) consider a symmetric VRPSDP improving a no-good cut method within the branch-and-bound tree. Öztaş and Tuş (2022) hybridises iterated local search, variable neighbourhood descent and threshold acceptance and

uses a roulette wheel to select routes during a perturbation phase based on route information. They compare their distance minimization objective with existing test problems from multiple frequently used sources.

A record-to-record travel algorithm for the HVRP is developed by Li et al. (2007) while referring to Taillard (1999)'s Column Generation technique of combining Linear Programming with Tabu Search. Liu et al. (2009) considered heterogeneous vehicles with unlimited fleet size to solve the vehicle fleet mix problem (FSMVRP) using genetic algorithms, while (Prins, 2009) considered VRPs with predetermined number of heterogeneous vehicles (HFVRP) as well as when the fleet size is unlimited (VFMP) using memetic algorithms; both minimizes the total trip cost. A hybrid algorithm for the Heterogeneous Fleet VRP utilising iterated local search and set partitioning was developed in (Subramanian et al., 2012), while a hybrid genetic algorithm based population is found in (Liu, 2013) considering variants of the fixed and variable costs. Pessoa et al. (2018)'s branch-price-and-cut algorithm for heterogeneous vehicle routing is also able to solve multiple depot and site-dependant vehicle routings, allowing vehicles to differ in capacity, cost, depot allocation or customer-subsets; but only consider the delivery problem. A tabu search algorithm solves the mix fleet size problem in (Meliani et al., 2019) boosted by an adaptive memory algorithm known as probabilistic diversification and intensification. A reinforcement learning based hyper-heuristic was used in (Qin et al., 2021) to minimize the maximum routing time of heterogeneous vehicles. Máximo et al. (2022) develop an adaptive iterated local search heuristic for heterogeneous fleet of vehicles minimizing the total travelling cost for the delivery problem.

A taxonomic review of the Pick-Up and Delivery Problem (PDP) alongwith its variants may be found in (Gutiérrez-Sánchez and Rocha-Medina, 2022) having unexplored perspectives. A classification table is presented in (Tan and Yeh, 2021) dividing models based on customer, vehicle or depot considerations, and algorithms into exact, heuristics and meta-heuristics. In their taxonomic review, (Elshaer and Awad, 2020) classifies PDPs according to the study by (Parragh et al., 2008); into backhauls (VRPB), clustered backhauls, mixed backhauls, divisible deliveries and pickups and VRP with simultaneous delivery and pickup (VRPSDP). Çağrı Koç et al. (2020) survey VRPSDP comparing heuristic performances describing several problem variants and applications. Çağrı Koç et al. (2016) classify the heterogeneous vehicle routing problems of delivery, highlighting the difference in nomenclatures of the HVRP which uses an unlimited number of heterogeneous

vehicles versus the HFVRP where the maximum fleet size is predetermined. They further talk about existing population, tabu search and other heuristics and computationally compare the results reported in literature. A MultiObjective Transportation Network Design is put forth in (Eksioglu et al., 2009) providing an all encompassing VRP taxonomy, pointing out the need for taxonomy evolution as "no taxonomy should be considered fixed for all time". Berbeglia et al. (2007) survey and classify static pickup and delivery problems mentioning VRPSDP as a multi-vehicle one-to-many-to-one PDP with combined requests.

We observe that the considered problem of vehicle routing (VRP) with heterogeneous vehicles for performing simultaneous delivery and pickup can descent from the HFVRP/HVRP (Heterogeneous Fleet/Heterogeneous), the VRPSDC/VRPSPD/VRPSDP (simultaneous delivery and collection / simultaneous pick-up and delivery / simultaneous delivery and pickup), as well as the TSP (Travelling Salesman Problem). Although the VRP has progressed a lot with multiple heuristics as well as exact formulations for various intricate cases, the Heterogeneous Fleet Vehicle Routing Problem with Simultaneous Delivery and PickUp (HFVRPSDP) does not yet seem to have the best compact formulation for fastest exact solutions. We therefore try to bridge this gap with a new compact formulation having capacity constrained heterogeneous fleet of vehicles with network compatibility being taken into consideration. We introduce the concept of redundant constraints which are real-life logics represented as constraints, and for this specific problem has been proven to have a redundancy as it can also be obtained from two other constraints considered already. Counter-intuitively, it seems that such redundant constraints help the formulation get to optimality faster for certain problem instances and further research should be done in this regard to develop the best exact formulation for the mHFVRPSDP or any class or problem for that matter. We compare both the types of formulation developed by us with the existing exact formulation in Avci and Topaloglu (2016) and Keçeci et al. (2021). We further generate new test cases highlighting the network differences for each vehicle type and compare the developed heuristics on these novel instances, by altering the exact formulations in (Avci and Topaloglu, 2016) and (Keçeci et al., 2021) slightly so as to allow for this new feature to be incorporated into their considered problem.

Table 1: Extended Vehicle Type specifications from Avci and Topaloglu (2016)

Sample of available Vehicle details at the single Depot						
Vehicle Type [T]	Number of this type of Vehicles [H]	Capacity [Q]	Variable Cost [V]	Fixed Cost [F]	Assuming differentiating attributes for each Layer w.r.t. OpenStreetMaps (Road Vehicle Type Compatibility for generating appropriate layers from the original Network)	p (for L^pNorm calculations)
V1	25	350	1	50	highway = Motorway, Trunk, Primary, Secondary, Tertiary, Unclassified, Residential	1.8
V2	15	450	1.04	80	highway = Motorway, Trunk, Primary, Secondary, Tertiary	1.6
V3	15	600	1.08	120	highway = Motorway, Trunk, Primary	1.4
V4	10	800	1.14	150	highway = Motorway, Trunk	1.2

3. Problem description and mathematical formulation

We consider a single depot with multiple types of vehicles which shall cater to demands at the relief points (nodes) as well as pickup evacuees in case of emergency and disasters. Each distinct Vehicle Type consists of homogeneous vehicles which shall be loaded with the required relief materials to be delivered to the Relief Points it is going to visit. We allow different Vehicle Types to differ by their Vehicular Capacity, Fixed routing Costs, Variable routing costs, as well as in their shortest paths between the two same nodes, (generated by changing p values[5] for each Vehicle Type as shown in Table 1). Since this does not consider multiple trips and as split delivery is disallowed, in all cases it must be ensured that no individual pickup or delivery value at any node exceeds the capacity of the largest vehicle used during the operation. We consider a linear formulation allowing only a single vehicle to visit each node, with the Delivery and PickUp activities performed simultaneously and in that order (therefore we prefer the term SDP to SPD in the VRP classifications). Similar formulations may be found in (Dell'Amico et al., 2006) considering homogeneous vehicles, and in (Avci and Topaloglu, 2016) and (Keçeci et al., 2021) for heterogeneous vehicles. A compact formulation for capacitated heterogeneous mVRP with a single depot may be found in (Golden et al., 2008, p. 8) which considers only a delivery problem; while another single vehicle problem on (Golden et al., 2008, p. 366) is also considered with both simultaneous as well as separate pickup and delivery.

[5]https://en.wikipedia.org/wiki/Lp_space#The_p-norm_in_finite_dimensions

3.1. Description of the problem

Here we consider a problem where heterogeneous vehicles performs both PickUp and Delivery simultaneously. We have reduced a major part of the formulation from (Avci and Topaloglu, 2016), where they consider different layers (variable k) for each individual vehicle. We consider Network Layers specific to each Vehicle Type such that each vehicle may travel in its Type-Layer. The requirement of different Network Layers for each Vehicle Type is to establish the Vehicle Type and road compatibility. Therefore for each Network Layer (i.e. each Vehicle Type) we may have different distance matrices. The vehicles are capacity constrained and each vehicle is allowed to cater atmost once to any unique relief point for performing delivery of goods (generally relief materials) and picking up stranded evacuees during the same stoppage. The same vehicle is to be used for delivery of the relief materials (which would include food, water, dry clothes, medicine, etc.) and for evacuation of highly vulnerable victims as well as livestock. The final destination point of the evacuees is considered to be the same depot which is assumed to have an unlimited amount of resources as well as capacity to cater to any number of evacuees. Since the pickup and delivery quantities must be in the same units as that of the vehicle capacity, this may be considered to be weight or volume as per specific use cases. In high emergency situations where time is of the highest priority, a time-minimization approach may be considered. Here we consider that a single vehicle may take only one trip, and minimize the total cost of all trips. The cost of any trip is dependant on the Vehicle Type and is proportional to the total length of a route and a fixed cost specific to the type of vehicle dispatched.

3.2. Mixed-Integer Linear Programming Formulation

In this section, we present a formulation for the problem described in Section 3.1. Table 2 details the parameters used for the formulation and Table 3 highlights the type and meaning of the decision variables used in the MILP. The model is defined as follows:

$$Minimize \sum_{k\in T}\sum_{i\in N_0}\sum_{\substack{j\in N_0\\(i\neq j)}} C_{ijk}x_{ijk}V_k + \sum_{j\in N}\sum_{k\in T} x_{0jk}F_k \tag{1}$$

Table 2: Notations of sets and parameters used

N	Set of all Nodes/Relief Points, where $\|N\| = n$
N_0	Set of all Relief Points and the Depot, where $\|N_0\| = n + 1$. (Depot being the 0^{th} Node)
p_i	Denotes the pickup quantity associated with Node i
d_i	Denotes the delivery quantity associated with Node i
C	Distance matrix consisting of network specific distances between two Nodes wrt. Vehicle Types. Each element may be referenced by C_{ijk} which is the shortest distance required to travel from node i to node j, $(i \neq j)$ for vehicles of type k. The vehicle type specific network layers are generated for each vehicle type according to road compatibility.
T	Set of all Vehicle Types, generally the k subscript is used to refer to this
Q_k	Denotes the capacity of a vehicle of type k
H_k	Denotes the number of vehicles available of its respective vehicle type k
F_k	Denotes the fixed cost related to dispatching a vehicle of type k
V_k	Denotes the variable cost per unit length of route related to dispatching a vehicle of type k

Table 3: Notations of decision variables used in the exact formulation

x_{ijk}	Binary decision variable which refers to whether a k^{th} type of vehicle is journeying from node i to node j, $(i \neq j)$ on layer k, $\begin{cases} 0 & \text{value means no Vehicle of type } k \text{ travels from } i \text{ to } j \\ 1 & \text{suggests a single Vehicle of type } k \text{ travels from } i \text{ to } j \end{cases}$
y_{ijk}	Continuous variable referring to the amount of pickup (humans) being carried by a vehicle of type k, on its specific layer also of the corresponding type k, between nodes i and j, $(i \neq j)$.
z_{ijk}	Continuous variable referring to the amount of delivery (Food, Water, Medicine, Sanitary Items, etc.) being carried by a vehicle type of k, on its specific network layer k, between nodes i and j, $(i \neq j)$.

subject to the following constraints:

$$\sum_{\substack{j\in N_0\\(i\neq j)}}\sum_{k\in T} x_{ijk} \leq 1, \quad \forall i \in N, \tag{2}$$

$$\sum_{\substack{j\in N_0\\(i\neq j)}} (x_{ijk} - x_{jik}) = 0, \quad \forall k \in T,\ \forall i \in N_0, \tag{3}$$

$$\sum_{j\in N} x_{0jk} \leq H_k, \quad \forall k \in T, \tag{4}$$

$$y_{0jk} = 0, \quad \forall j \in N,\ \forall k \in T, \tag{5}$$

$$z_{i0k} = 0, \quad \forall i \in N,\ \forall k \in T, \tag{6}$$

$$\sum_{k\in T}\sum_{\substack{j\in N_0\\(i\neq j)}} (y_{ijk} - y_{jik}) = p_i, \quad \forall i \in N, \tag{7}$$

$$\sum_{k\in T}\sum_{\substack{j\in N_0\\(i\neq j)}} (z_{jik} - z_{ijk}) = d_i, \quad \forall i \in N, \tag{8}$$

$$y_{ijk} + z_{ijk} \leq x_{ijk}Q_k, \quad \forall i, j \in N_0,\ (i \neq j),\ \forall k \in T, \tag{9}$$

$$\sum_{i\in N}\sum_{k\in T} y_{i0k} = \sum_{i\in N} p_i, \tag{10}$$

$$\sum_{i\in N}\sum_{k\in T} z_{0ik} = \sum_{i\in N} d_i, \tag{11}$$

$$y_{ijk}, z_{ijk} \geq 0, \quad \forall i, j \in N_0,\ (i \neq j),\ \forall k \in T, \tag{12}$$

$$x_{ijk} = \{0, 1\}, \quad \forall i, j \in N_0,\ (i \neq j),\ \forall k \in T. \tag{13}$$

We consider the objective function to be the total cost for all vehicle tours as shown in Eq. 1, which is minimized. Here V_k is a variable cost which may be compared to distance dependant elements like fuel or emissions. F_k is the fixed cost which may be considered as the per tour expense and towards maintaining and sending a crew consisting of the driver and personnel helping in Disaster Relief. Eq. 2 constraints a single vehicle to be allowed to visit any relief centre/node. Removing this constraint shall allow multiple vehicles to tend a Relief Point through different paths which would be able to provide better objective values. However, we considered this constraint since catering to different vehicles at different times might prove cumbersome as it wouldn't enforce simultaneous deliveries and pickups. Eq. 3 ensures the same number of each type of vehicles entering any node also leaves it. Eq. 4 ensures at most H_k vehicles are allowed to exit the depot for the specific vehicle layer type k. Flow limitation constraints of pickup and delivery are provided to get the exact values in the respective flow variables. Eq. 5 assigns the value 0 to all outgoing pickup values from the Node and Eq. 6 assigns the value 0 to all incoming delivery values from the Node. Resource flow equations 7 and 8 for each relief point ensures the pickup and delivery constraints are satisfied. Eq. 9 limits the vehicle capacities according to the vehicle's type. Equations 12 and 13 are the fundamental definitions of the types of variables used for the decision making process through computation.

3.3. Changes made in regards to existing formulations

1. as is available in Avci and Topaloglu (2016): The present formulation is more compact. Avci and Topaloglu used different Vehicle Layers (for each Vehicle) even when the problem was of simultaneous delivery and pickup with any Node to be visited by only a single vehicle. We replaced each vehicle's layer with Layers pertaining to each Vehicle Type, since the network, costs and distance matrix would be different for different types of vehicles. While considering the road network to be different for different Vehicle Types (in the cases where the distance matrix for

different vehicle types are not equal, the distance calculations here use p values which are unique for each Vehicle Type), we needed a slight modification in their objective function calculation to change c_{ij} to c_{ijk} (Equation 1 in (Avci and Topaloglu, 2016)). We also observed an error in the their Flowchart representing their heuristic's logical sequence (Fig. 2 in (Avci and Topaloglu, 2016)). Unlike as mentioned in their paper's text as well as in the algorithm (Fig. 1 in (Avci and Topaloglu, 2016)), their Flowchart does not ever use the *age* variable and therefore may be modified (however this does not make the Flowchart's present logic absolutely dysfunctional, but makes the heuristic terminate much earlier with generally worse results than the Algorithm's logic).

2. as is available in Golden et al. (2008): We have expanded on this compact formulation, which only considers a delivery problem. Since we are also considering simultaneous pickups, we therefore require other constraints ensuring this.

3. as is available in Keçeci et al. (2021): We find this formulation to be very good and seems to outperform our developed formulations. However, we were able to incorporate this formulation for our computations after two minor corrections:

   (a) c_{ij} should be changed to c_{ijk} in their Objective function (Equation 1 in (Keçeci et al., 2021)) since it would automatically vary with k as per their equation $c_{ijk} = \theta_k l_{ij}$ in (Section 3.1 in (Keçeci et al., 2021))

   (b) Equation 7 in (Keçeci et al., 2021) should be modified as Eq. 14 below:

$$\sum_{\substack{j \in N \\ (i \neq j)}} (z_{ji} - z_{ij}) = d_i \quad \forall i \in N \setminus \{0\}, \tag{14}$$

3.4. Redundant Constraints and the need for Heuristics due to drawbacks of Open-Sourced Exact Solvers

Equations 10 and 11 are redundant constraints which ensure the sum of all flows from and to the origin (Depot) is equal to the total pickup or delivery. The proof of redundancy of Eq. 10 (similar to the equation 9 in (Avci and Topaloglu, 2016)) may be obtained by adding all the above pick-up constraints, together found in Eq. 7:

$$\sum_{i\in N}\sum_{k\in T}\sum_{\substack{j\in N_0\\(i\neq j)}}(y_{ijk}-y_{jik})=\sum_{i\in N}p_i \tag{15}$$

Now, splitting only the ranges in the summation, we get

$$\sum_{i\in N}\sum_{k\in T}\sum_{\substack{j\in N\\(i\neq j)}}(y_{ijk}-y_{jik})+\sum_{i\in N}\sum_{k\in T}\sum_{j\in 0}(y_{ijk}-y_{jik})=\sum_{i\in N}p_i \tag{16}$$

The left hand term on the LHS is zero since all the same variables are being added and substracted :

$$0+\sum_{i\in N}\sum_{k\in T}(y_{i0k}-y_{0ik})=\sum_{i\in N}p_i \tag{17}$$

$$\sum_{i\in N}\sum_{k\in T}y_{i0k}-\sum_{i\in N}\sum_{k\in T}y_{0ik}=\sum_{i\in N}p_i \tag{18}$$

The right hand term in the LHS of Eq. 18 is the sum of all the individual variables, each of which was nullified in Eq. 5,

$$\sum_{i\in N}\sum_{k\in T}y_{i0k}-0=\sum_{i\in N}p_i \tag{19}$$

Ultimately we reach the constraint in Eq. 10 which is the same as Eq. 19 deriving it from two other set of equations. Similarly, Eq. 11 may also be proved to be redundant. The reason of using similar redundant constraints in equations 9 and 10 in (Avci and Topaloglu, 2016) is not explicitly mentioned.

These two redundant constraints may be retained when using open-sourced solvers like COIN-OR CBC provided within PuLP package of Python, since they are expected to decrease the solution time; however the results have been inconclusive when using the commercial Gurobi solver. Improvement in solution time may be seen in Table 4 where one instance is solved using COIN-OR CBC solver using PuLP module of Python programming language;

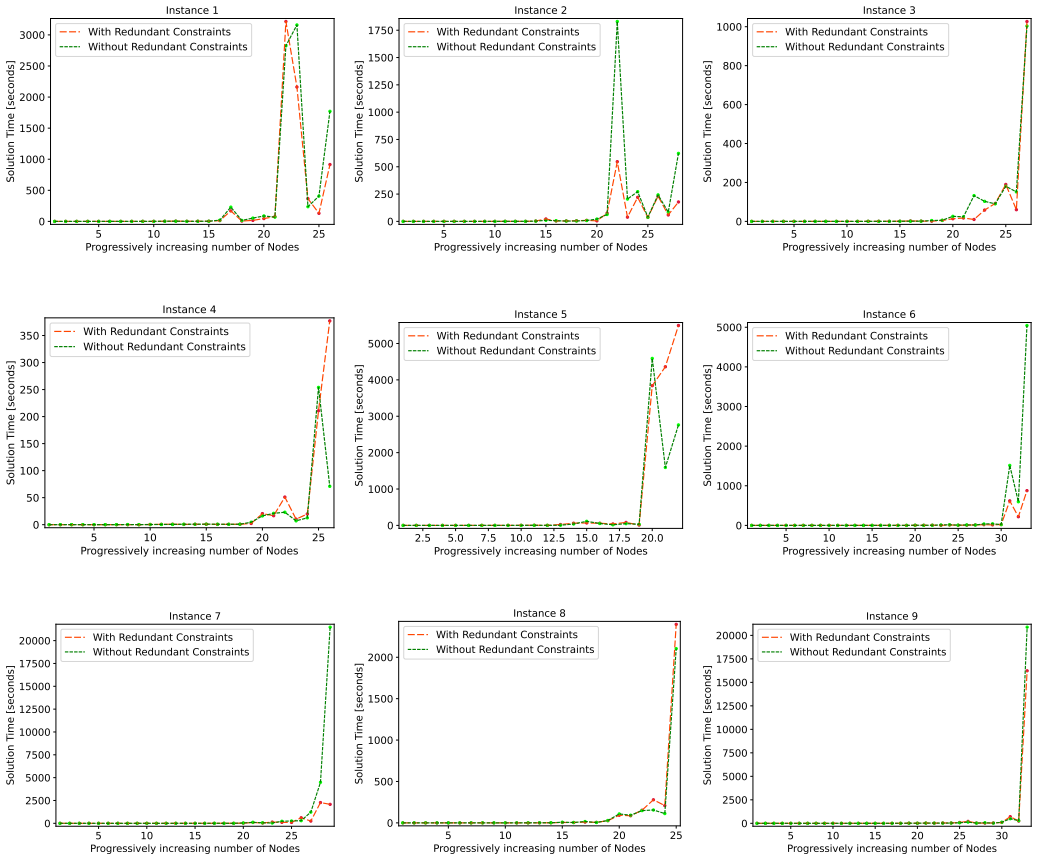


Figure 1: Comparing Gurobi solution times with and without the redundant constraints in Eq. 10 and 11

by progressively including more Nodes for 9 types of generated data. The objective values in both cases were identical. Many other instances were further created for similar tests by progressively increasing the Number of Nodes in these individual mHFVRPSDP problems and searching for optimality. These were solved using the Gurobi commercial solver as shown in Fig. 1. These results seem inconclusive and how redundant constraints helping to achieve optimal solutions faster needs to be a subject of future enquiry.

Previously, Telgen (1983) has identified redundant constraints while referring to Bradley et al. (1983) which discusses automatic detection of structural redundancy. Berbee et al. (1987) develop algorithms to identify non-

Table 4: For each of these individual problem instances, all 4 Vehicle Types were considered with 1 vehicle in each type. Data taken from (Avci and Topaloglu, 2016) Table 2 by progressively including new Nodes.

Number of Nodes used (All Vehicles have been considered)	Optimal Objective Value WITH Constraints (10) and (11)	Time Taken for Solver to compute WITH Constraints (10) and (11)	Optimal Objective Value WITHOUT Constraints (10) and (11)	Time Taken for Solver to compute WITHOUT Constraints (10) and (11)
1	118.67	2	118.67	0.08
2	231.48	0.24	231.48	0.37
3	231.87	0.71	231.87	0.53
4	272.81	1.51	272.81	0.54
5	331.81	1.8	331.80	2.37
6	335.28	2.36	335.28	1.79
7	377.04	4.97	377.04	19.35
8	379.86	9.02	379.86	27.9
9	431.27	59.36	431.27	242.88
10	442.61	127.62	442.61	995.23
11	449.58	249.86	449.58	441.62
12	451.89	2320	451.89	8017.19
13	467.65	9656.34	467.65	28089.86

[Computed on Intel® i3 2.1 GHz processor with 8 GB RAM and Windows 11 64-bit Operating System]

redundant linear inequalities using probabilistic preprocessors while referring to (Rabin, 1976); while Caron et al. (1992) do a performance analysis on similar algorithms detecting necessary constraints. Strategies for classification of linear constraints as redundant may be found in Caron et al. (1989) where they produce numerical savings in the required computational effort to classify constraints; and analyse random methods for detecting necessary linear inequalities in Caron and McDonald (1989). Gal (1992) provide a degeneracy and redundancy determining procedure highlighting their impact in post-optimal analyses. Charles and Dutta (2006) also talk about identification of redundant constraints albeit in Fractional Programming problems. Reporting of problem specific redundant constraints have been previously investigated as in (Jr and Tseng, 2003); as well as in (Asarin et al., 2010) where redundant constraints were used for "refinements" S. and P. (2010) discuss redundant constraints, classifying them (Definition 2.1) while highlighting they may consume extra computational effort. Wojtyra and Frączek (2012) provide redundant constraint handling methods of elimination, pseudoinversion and augmented Lagrangian's for kinematics problems; while Zhu et al. (2021) develop a method to remove redundant constraints.

In our problem, it can be observed that the open-source solver takes a lot of time to solve for small instances of the problems considered. During disaster management, this would be a cause of concern since good solutions are required within less time and for this we develop some heuristics.

4. Proposed Heuristic Algorithm

We develop three Heuristics named ATSAH (ATS Algorithm Heuristic), ATSA-MT (ATS Algorithm with Machine Tuning) and ATSA-MT-JF; where ATS refers to the naming convention of the Heuristic in (Avci and Topaloglu, 2016) and the next A refers to Algorithm which is the flavor of the ATS implementation we have considered ((Avci and Topaloglu, 2016) has subtle differences among their heuristic implementations as described in their Flowchart, Algorithm and Text of the paper.) MT here refers to Machine Tuning of the mean values of the Edge selection criterias which are dynamic logics. JF refers to the continued approach of expanding and contracting the range of values for individual parameters affecting selection criterias of Edges. For each individual problem, the edge selection criterias needs to be flexible enough to find good solutions. This machine tuning helps to converge upon new logics for selecting daughter edges. This tuning would become spe-

cific to the problem's data which improves with the Heuristic's search. No tabu list is necessary since the developed heuristics shall not always give a deterministic result for same Node sequence. Below is the description of the heuristic components and their integration:

4.1. Node Sequence

A Node Sequence is a linear array of the Nodes. Any Node Sequence must start with the Depot, which is the 0^{th} Node, and then include all the Nodes in any order. The initial Node Sequence is the sequence of Nodes starting from 0 to the maximum Node Number.

4.2. Neighbourhood Structures

To generate new Node Sequences from the set of all Nodes we use the following eight different Neighbourhood Structures expanding on the original four as used by (Avci and Topaloglu, 2016):

1. Shuffle Random: Generating a randomly shuffled Node Sequence from the existing Node Sequence
2. Reversal: Reversing an arbitrary length (we kept it > 3 to distinguish this from Adjascent Swap discussed herinbelow) of the Node Sequence, not including the first Node
3. Single Insertion: Taking one Node from the Node Sequence and placing it anywhere else
4. General Swap: Swapping any two Nodes of the Node Sequence
5. Adjascent Swap: Swapping adjacent Nodes of the Node Sequence

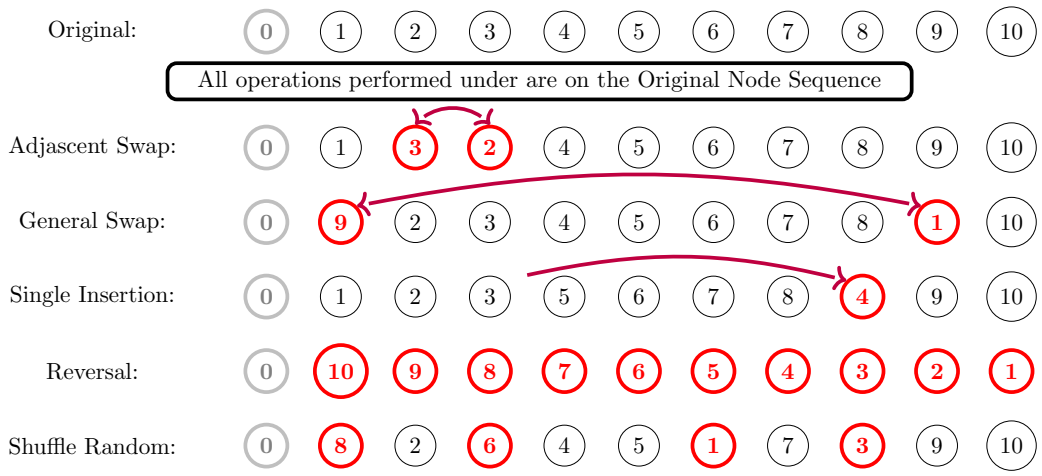


Figure 2: Neighbourhood Structures 1 to 5: First 4 taken from Avci and Topaloglu (2016)

6. SubArray Transplant allowing intermediate Operations:

   (a) Generate the SubArray randomly using either of the below methods:

      i. Take a random length of consecutive elements from the Node Sequence

      ii. Take elements from arbitrary locations in the Node Sequence, such that the number of elements may be equal to a generated random length less than the length of the Node Sequence

   For all the Heuristic solutions discussed below, the probability of using the first method was double that of using the second method.

   (b) Remove the Sub-Array from the Node Sequence creating a new shorter Trimmed Node Sequence without the Sub-Array elements.

   (c) Perform either of the below operations on the SubArray:

      i. Reverse the SubArray

      ii. Shuffle the SubArray elements randomly

   For all the computed Heuristic solutions; either of the above operations were allowed to happen with separate one-third probabilities; thereby allowing the SubArray to remain intact as well.

   (d) Perform Operations on the Trimmed Node Sequence without Sub-Array elements:

i. Reverse the Trimmed Node Sequence

ii. Shuffle the Trimmed Node Sequence randomly

For all the computed Heuristic solutions; either of the above operations were allowed to happen with separate one-third probabilities; thereby allowing the Trimmed Node Sequence to remain intact as well.

(e) Insertion of the SubArray within the Trimmed Node Sequence is done using either of the following ways:

i. Single Bulk Insertion of the SubArray, in a reversed manner, at a random location in the Trimmed Node Sequence.

ii. Insert each element of the SubArray at random locations in the Trimmed Node Sequence

In our case, the first method had double the probability of being used than the second. At the end of this process, we get a new Node Sequence containing all the original elements only once.

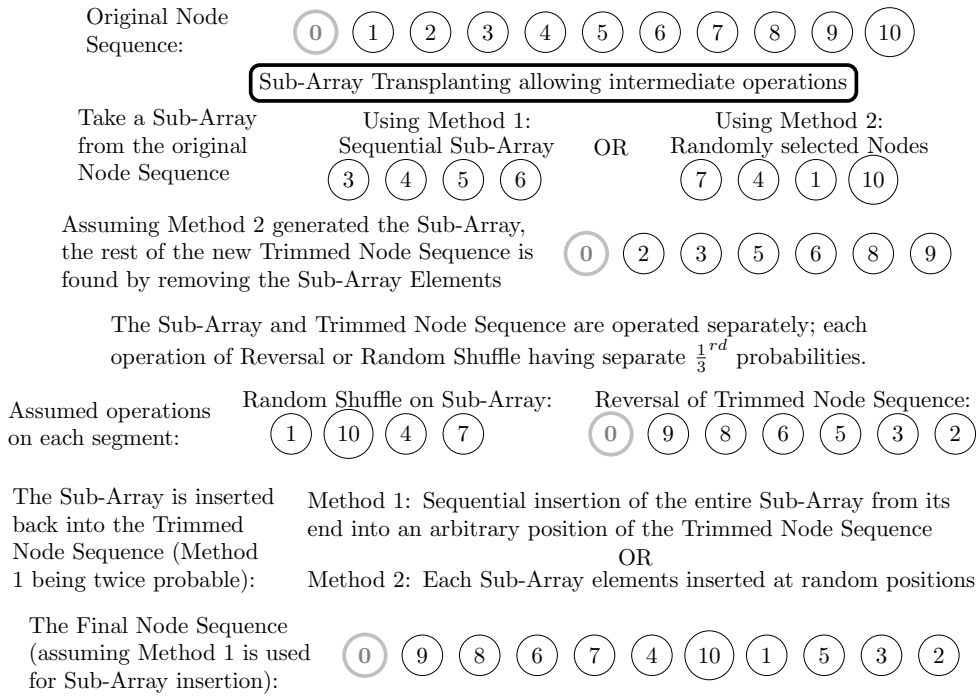


Figure 3: Neighbourhood Structure 6: Sub-Array Transplant

7. Linear Cake Cutting and Accumulation: Each of the previous 6 Neighbourhood Structures are performed only on some limited portion of the Node Sequence with separate probabilities of $\frac{1}{3}^{rd}$ usage of any Neighbourhood Structure. The selected portions are allowed to overlap amongst themselves.

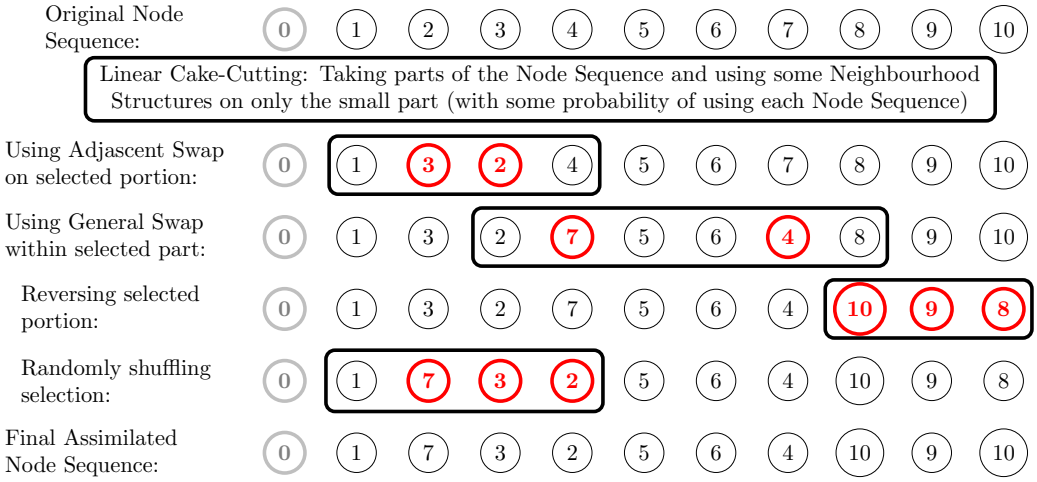


Figure 4: Neighbourhood Structure 7: Explanation of Linear Cake-Cutting

8. Linear Cake Cutting once without Overlap: A random portion of the Node Sequence is chosen. On this chosen portion, each of the first 6 Neighbourhood structures are performed with a probability of $\frac{1}{3}$.

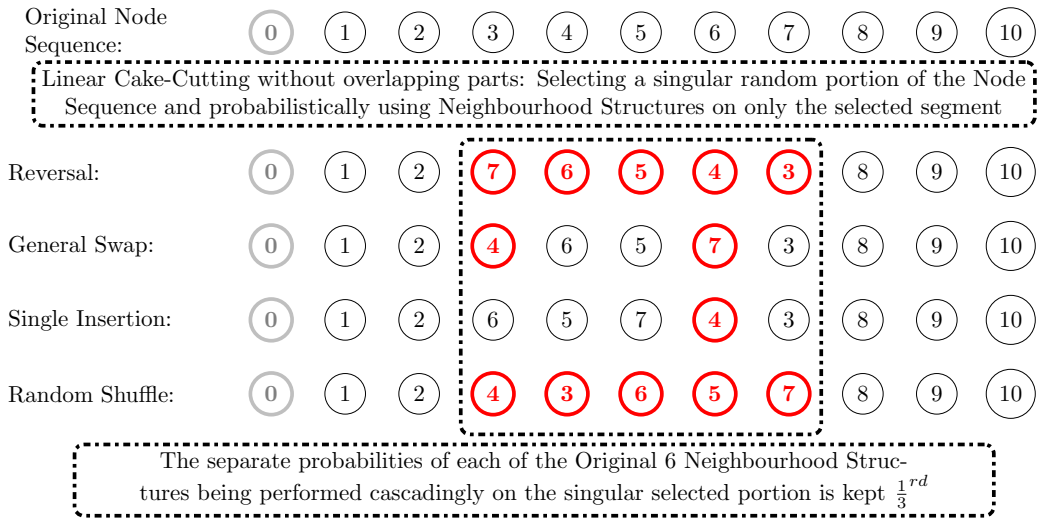


Figure 5: Neighbourhood Structure 8: Showcasing Linear Cake-Cutting without overlapping parts

In all the above cases, it needs to be ensured that the first Node remains the Depot. The working of all the above Neighbourhood structures are shown in Figures 2,3,4 and 5 operating on an Original Node Sequence, with the first 4 Neighbourhood Structures as taken from (Avci and Topaloglu, 2016).

4.3. Decoding Mechanism

The representation of a feasible route into Edges of the Node Sequence is taken from (Avci and Topaloglu, 2016). We first describe the nomenclature used and then the process utilized.

1. Edges: Each edge is a directed connection from one Node of the Node sequence to another subsequent Node. Since the Node sequence is an array, the Edges always start from a lower positional number and end at any Node which is in a higher array position. An Edge contains minimum 2 Nodes, one starting and one stopping Node and its uniqueness is identified by the following information:

   (a) Starting Node

   (b) Stopping Node

   (c) Vehicle Type Used: Each edge represents a tour being traversed by a Vehicle of among the specified types.

(d) Cost of the Edge: This is an additional information we retain to calculate the objective value. To determine the cost of the Edge we explain what the Edge represents. Each Edge represents a tour which starts from the Depot, sequentially covers all Nodes which are present next in the Node Sequence (at higher array positions) than the starting Node of the Edge, upto the last Node of the edge which is the stopping Node itself, and then back to the Depot. If this route is not feasible for any Vehicle Type, then the Edge would not exist. In case this is feasible, the cost of this tour is the cost of the Edge. In this SDP problem, we allow any Node to be visited by only one Vehicle and therefore use the following process to check feasibility of an Edge:

i. Initialise a DynamicVehicleCapacityLeft variable which would initially have the value equal to the capacity of the concerned Vehicle Type. We shall keep track of this variable at each Node of the route, including the Depot. If it's value becomes negative ever, then the route is infeasible and the Edge is not constructed; otherwise we get the unique feasible route as represented by it's Edge.

ii. At the Depot, all the deliveries need to be loaded into corresponding vehicles. Therefore the DynamicVehicleCapacityLeft variable is updated by substracting the sum of all demands of the route (i.e. for all Nodes sequentially at higher array positions than the starting Edge position, upto the last Node at which the Edge ends).

iii. In each subsequent Node, the Node's delivery amount is added to the Vehicle's left over capacity and the respective pickup quantity is reduced. Therefore for each Node we update the DynamicVehicleCapacityLeft variable by adding the Delivery quantity for that Node and then substracting the PickUp quantity.

2. Vehicles remaining to be used of each Type: Initially, this is the information regarding the number of vehicles of each type which are available to be used for the emergency operation. As we use up some of these Vehicles for creating some routes (Edges) we shall keep updating this. We will therefore only be creating Edges for those Vehicle Types which are available here. (In case of ATSAH where multiple Mother

Graphs are developed due to the potential of more than one Daughter Edge selection, this information of remaining vehicles must be unique for each Graph and therefore stored separately.)

3. Traversed Graphs Set: This is the set consisting of all Final Graphs.

4. Mother Graph: Each Mother Graph consists of some Edges. One of these Edges must start from the first Depot Node and end at another Node down the array of the Node Sequence (say Node L). The next Edge starts from this end Node (Node L), and ends further down the array of the same Node Sequence. Thus the Mother Graph is a continuous series of such connected Edges, the first of which starts at the Depot. Completed Mother Graphs are moved to a set of Final Graphs. By completed, we mean that the Mother Graph's set of Edges are such that one of them starts at the Depot, and ends at another Node, from which another Edge starts; and this process ultimately continues such that no more Nodes are left, i.e. there is an Edge ending at the last Node of the Node Sequence. These completed Mother Graphs will not give rise to new Daughter Edges and are therefore moved to the Traversed Graphs Set.

   Each Mother Graph also has another information of the number of Vehicles left to be used of each type. Whenever a new Daughter Edge is added to its Mother Graph, this information gets updated, (i.e. the number of Vehicles of the Daughter Edge's Vehicle Type, is reduced by 1). Each Mother Graph also stores the ending Node position of its last Edge. This is updated every time a new Daughter Edge is added to it; in which case the ending Node of the Mother Graph becomes the same ending Node of the inserted Daughter Edge. This information helps to create the next Edge as this Node will serve as the starting Node for subsequent Daughter Edge(s). When Mother Graphs become completed and are moved to the Traversed Graphs Set, this information becomes redundant and may be dropped.

5. Daughter Edge: A single Mother Graph can generate multiple Edges termed (Daughter Edges), all of which start from it's ending Node. A few of these Edges are carefully collected (using the Edge selection strategies as mentioned below) and each collected Daughter Edge is added into it's own copy of it's corresponding Mother Graph from

which it was generated. Each of these few new Mother Graphs continue this process of expanding themselves by adding a selected number of Daughter Edges into their own separate copies unless the Graph becomes completely traversed. [Completely traversed refers to the last Node of the Node Sequence having been reached (i.e. an Edge has it's end point as the last Node) and so no new Daughter Edges can be formed. Since completely traversed Mother Graphs cannot generate new Daughter Edges, they are considered to have become a Final Graph and are added to the Traversed Graphs Set].

6. Final Graph: Each Final Graph is a set of Edges. One of these Edges of a Final Graph starts at the Depot and ends at another Node from where the next Edge starts; and this process continues until the last Edge ends at the last Node of the Node Sequence. Unlike Mother Graphs, Final Graphs have no leftover Nodes to consider further.
   The leftover Vehicles for each Final Graph represents the final unutilized Vehicles. Each Final Graph represents a feasible solution to the problem.

7. Parent Graphs Set: This is a set, where each element represents a Mother Graph. Each of these Mother Graphs generate multiple Daughter Edges. A few of these Daughter Edges are selected according to the Edge Selection criterias. Each selected Daughter Edge is added to a copy of its Mother Graph and this updated Mother Graph is added to either Traversed Graphs Set or Next Parent Graphs Set, depending on whether the selected Daughter Edge ended at the last Node of the Node Sequence or not respectively.

8. Next Parent Graphs Set: This is a set, where each element represents a Mother Graph. Every time new Daughter Edges are selected and few are added to it's respective copy of it's Mother Graph, it is checked whether these new Mother Graphs can further generate Daughter Edges [in which case the updated copy of the Mother Graph containing the new Daughter Edge is added to the set of Next Parent Graphs Set], or whether the last Node in the Node Sequence has been reached [in which case it is moved to the set of Final Graphs]. Therefore this set serves as a dynamic temporary set of next Mother Graphs. So after all operations on Parent Graphs Set is over, we initialize Next Parent

Graphs Set to be the Parent Graphs Set and continue until no Mother Graphs enter this dynamic temporary set of Next Parent Graphs Set.

Initially all three sets, (i.e. Parent Graphs Set, Next Parent Graphs Set and Traversed Graphs Set) are empty. The algorithm starts generating Mother Graphs and keeps populating the Parent Graphs Set. It expands each Mother Graph present in the Parent Graphs Set by generating Daughter Edges for that Mother Graph, selecting few Daughter Edges greedily or through the progressive Machine-Tuned criteria, adding these selected Daughter Edges to its own respective copy of it's Mother Graph, and dumping each newly updated Mother Graph into Next Parent Graphs Set/Traversed Graphs Set.

This process is shown in Fig. 6 where from a single Mother Graph (rectangular boxed), new Graphs are being created by extending new Daughter Edges. The different types of arrows represent Edges formed by different Vehicle Types. The (red) crossed out Edges have their corresponding routes as infeasible and therefore these two Graphs are rejected/not formed. The left column of the figure represents the Parent Graphs Set consisting of multiple Mother Graph. The third Graph from top-right in the figure, is completely traversed and is moved to Traversed Graphs Set. The rest of the four incomplete Graphs on the right are a part of Next Parent Graphs Set. After all Mother Graphs in the Parent Graphs Set have been considered; Next Parent Graphs Set becomes equal to Parent Graphs Set. In case of ATSA-MT, expansion of the solo Mother Graph is done by selecting only one among the feasible Daughter Edges; whereas in the case of ATSAH at most two new Daughter Edges can be selected from each Mother Graph which therefore may keeps more than one Mother Graph in the Parent Graphs Set. This process is repeated until the Parent Graphs Set become null.

Once there are no elements within the Parent Graphs Set; the Traversed Graphs Set is taken into consideration. Each Final Graph consists of a number of Edges which have its correspondingly associated costs. All these Edge costs are added to get the total cost of it's corresponding Final Graph. Then the cost of each Final Graph within Traversed Graphs Set, is compared and the least costly Final Graph is chosen as the solution of the Decoding Mechanism.

In case the number of Vehicles are not sufficient, the left-over Vehicles remaining to be used for each Type will attain 0 values for a Mother Graph even when it's not complete, (i.e. there are Nodes left to be considered and therefore these left-over Nodes are not part of any Vehicle's tour). Such

Mother Graphs need not be proceeded further with, and would be dropped off; since there would be no Vehicle left of any Type for new Daughter Edge creation.

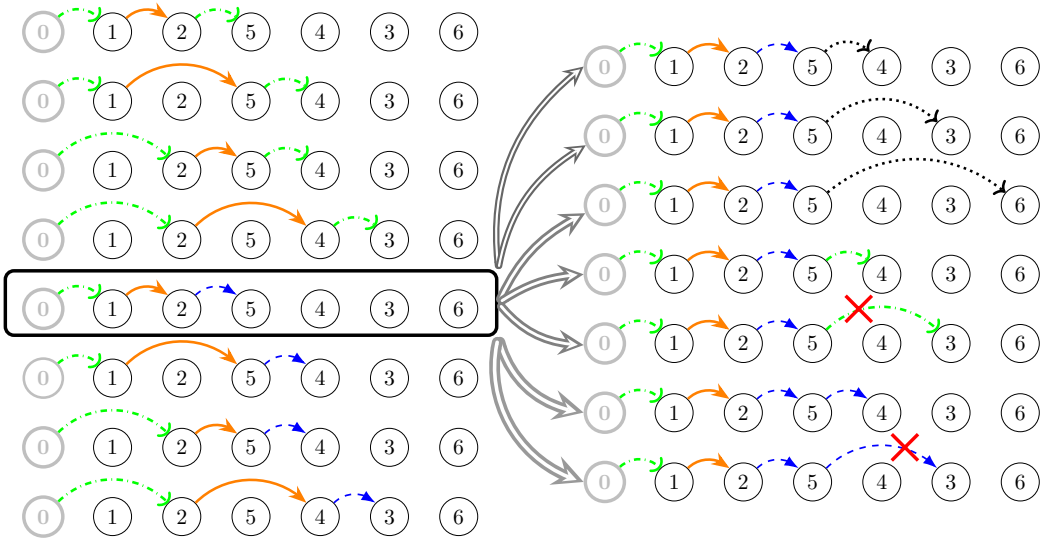


Figure 6: Expansion of a single Mother Graph from within the Parent Graphs Set on the left is being shown by addition of potentially feasible Daughter Edges. Infeasible Daughter Edges have been crossed out (in red). Each unique type of arrow for the edge connections represent a unique Vehicle Type.

4.4. Starting the Decoding Mechanism

We start by iterating over all Mother Graphs present in Parent Graphs Set. At the beginning, since the Parent Graphs Set would be null, we may create an initial Mother Graph with no Edges, left-over Vehicles to be used as equal to the original availability, and the starting Node Location as the Depot Node. From this information, new Edges are created for this initial Mother Graph and dumped to the set of Daughter Edges from which some are selected using Edge Selection Strategies. When these selected Edges are added to new copies of this initial Mother Graph for each selected Edge, to generate new Mother Graphs for each new selected Daughter Edge, the newly generated Mother Graphs have updated information regarding the left-over Vehicle Types as well as the next starting Node, which would be the ending Node of the selected Daughter Edge. It must be noticed that new Mother Graphs are created for each selected Edge and then each of these

Mother Graphs are further allowed to generate Daughter Edges and create new Mother Graphs for each of the few selected Daughter Edges. If the number of Daughter Edge creation is big (even a constant 2); the processing time taken would increase enormously. We may estimate the number of combinations of Mother Graphs which may be created when considering all Daughter Edges subsequently and thereby discuss the Edge selection strategies used and it's importance in speeding up the solution. [For some arbitrary vehicle capable of visiting around 20 Nodes out of total 200 Nodes, there would be more than 20^{12} unique Final Graphs possible, if we select all Daughter Edges. therefore selecting one or two of the best logical Daughter Edges has helped in speeding up the ATSAH. For ATSA-MT, we only select 1 Daughter Edge].

4.5. Edge selection strategy for ATSAH

From among the set of Daughter Edges, we select a few Edges using a host of Edge selection strategies. It is to be noted that these Edge selection strategies must play a key role in improving the solution time. For each of the selected Daughter Edges, it is checked whether the stopping Node is the last Node of the Node sequence. If the last Node of the Node Sequence is also the ending Node of the Edge, then this respective selected Daughter Edge is added to a copy of its Mother Graph and this updated Mother Graph is added to Traversed Graphs Set. If the Daughter Edge doesn't end at the last Node of the Node Sequence, then this selected Daughter Edge is added to a copy of its Mother Graph and this updated Mother Graph is added to Next Parent Graphs Set.
Logically accepting the best Daughter Edges is a challenging task and further improvement must be done in this regard.We employ the following selection strategies:

(a) Accepting only the Longest Edge: The length of a Edge is equal to the number of Nodes it encompasses within its starting and ending Nodes. Since these Edge selection criterias are only applicable for Daughter Edges from the same Mother Graph, the longest Daughter Edge must have its ending Node position in the array of Node Sequence as the highest among the other Daughter Edges. The length of an Edge may be calculated by finding the difference between the Edge's ending Node position in the Node Sequence and the Edge's starting Node position in the Node Sequence.

(b) Accepting the minimum Cost Edge with respect to the Number of Nodes in that Edge: The Edge Cost is divided with the Edge Length and the Edge which has the lowest of this value is selected.

(c) Accepting the minimum Cost Edge with respect to the cumulative Number of Nodes: Cumulative Number of Nodes refers to the addition of integers starting from 1 upto the length of each respective Edge. The Edge Cost is divided by this number and the Edge with the lowest value is taken.

(d) Accepting the Edge with the minimum Distributed Cost.

$$DistributedCost = \frac{Variable\,Cost\,of\,Edge}{Cumulative\,Number\,of\,Nodes} + \frac{Fixed\,Cost\,of\,Edge}{Number\,of\,Nodes} \tag{20}$$

where, the Cumulative Number of Nodes (CN) is obtained from

$$CN = L\left(\frac{L+1}{2}\right) \tag{21}$$

L being the number of Nodes in the Edge, i.e. the Edge Length. The Variable Cost of an Edge is obtained by subtracting the Fixed Edge Cost from the Total Edge Cost F_k, where k is the Vehicle Type of that Edge.

(e) Using any of the above (a), (b), (c), (d) Edge selection methods with uniform probability of strategy selection.

(f) Accepting either the Longest or the Second Longest Edge: In case of the ATSAH for which the results have been reported, the Longest Edge is selected with double probability than that of selecting the second Longest Edge.

(g) Accepting the Edge with the minimum cost with respect to number of nodes, or the second-minimum: In case of the ATSAH for which the results have been reported, the Edge with this minimum value was selected with double probability than that of selecting the Edge with the second-minimum value.

(*h*) Accepting the Edge with the minimum Cost with respect to cumulative number of Nodes, or the Second-Minimum: In case of the ATSAH for which the results have been reported, the Edge with this minimum value was selected with double probability than that of selecting the Edge with the second-minimum value.

(*i*) Accepting the Edge with the minimum Distributed Cost, or the second-minimum, with the Edge with this minimum values being selected with double probability.

(*j*) Using any of the above (f), (g), (h), (i) Edge selection methods with uniform probability of strategy selection.

(*k*) Using any of the above (f), (g), (h), (i) Edge selection methods with uniform probability of strategy selection, and selecting both the minimum and the second minimum Edges with certain probabilities. These probabilities may be adjusted further and for the ATSAH solutions reported, the probability of accepting the second-minimum-Edge was $\frac{14}{27}$ and that of accepting the Edge with the minimum value was kept at $\frac{21}{27}$. It was ensured that a minimum of one Edge is selected.

(*l*) Random Acceptance: This strategy randomly accepts one Daughter Edge.

(*m*) Random Acceptance 2: This strategy randomly accepts atleast one Daughter Edge. It may accept another random Daughter Edge, with a small probability. In case of the ATSAH solutions discussed here-in-below, this probability was taken 10%. Increasing this probability will increase solution time drastically and may not improve the Objective Value much.

(*n*) Using any of the above Edge selection methods with uniform probability of strategy selection. For this case, a copy of the above Edge selection criterias were created so that each individual internal parameter could be altered. For the copy of selection strategy mentioned in (*k*), the probability of accepting the Edge with minimum-cost was updated to $\frac{407}{500}$, and the probability of accepting the Edge with second-minimum cost was updated to $\frac{275}{500}$, ensuring atleast a single Edge is always selected. The probability of accepting the second another random Daughter Edge in copy of strategy *l* was also updated to be 25%.

4.6. Edge selection strategy for ATSA-MT

From among the set of Daughter Edges, a single Edge is selected using a progressively developing Edge Selection strategy. Unlike the methods as used in ATSAH, the logics for selecting the Daughter Edges is self-tuned by the algorithm. For each of the selected Daughter Edges, it is checked whether the stopping Node is the last Node of the Node sequence. If the last Node of the Node Sequence is also the ending Node of the Edge, then this respective selected Daughter Edge is added to a copy of its Mother Graph and this updated Mother Graph (which has now become a Final Graph) is added to Traversed Graphs Set. If the Daughter Edge doesn't end at the last Node of the Node Sequence, then this selected Daughter Edge is added to a copy of its Mother Graph and this updated Mother Graph is added to Next Parent Graphs Set.

Logically accepting the best Daughter Edges is a vital portion of this ATSA-MT Heuristic. The following parameters of an Edge were identified:

1. Length ($\mathscr{L}$): The length of a Edge is equal to the number of Nodes it encompasses within its starting and ending Nodes. The length of an Edge may be calculated by finding the difference between the Edge's ending Node position in the Node Sequence and the Edge's starting Node position in the Node Sequence.

2. Fixed Cost ($\mathscr{F}$): Each Edge has an associated cost and its Vehicle Type. From it's Vehicle Type the Fixed Cost may be easily determined.

3. Variable Cost ($\mathscr{V}$): The Fixed Cost of the Edge may be substracted from its Total Cost to obtain the Variable Cost of the Edge.

The general relationship among the above parameters was developed into the following expressions:

Expression 1:

$$a\mathscr{F}^b\mathscr{V}^c\mathscr{L}^d + e\mathscr{F}^f\mathscr{V}^g + h\mathscr{V}^i\mathscr{L}^j + k\mathscr{L}^l\mathscr{F}^m + n\mathscr{F}^o + p\mathscr{V}^q + r\mathscr{L}^s \quad (22)$$

Expression 2:

$$h\mathscr{V}^i\mathscr{L}^j + k\mathscr{L}^l\mathscr{F}^m \quad (23)$$

Expression 3:

$$a\mathscr{F}^b\mathscr{V}^c\mathscr{L}^d + h\mathscr{V}^i\mathscr{L}^j + k\mathscr{L}^l\mathscr{F}^m \quad (24)$$

Table 5: Starting Mean and S.D. of parameters in Equations 22, 23 and 24

	a	b	c	d	h	i	j	k	l	m
Range of selecting the initial Mean [μ] from a uniform random distribution	[0,1]	[0,1]	[0,1]	[-1.25,1]	[0,2]	[0,1]	[-2,1]	[0,2]	[-2,1]	[0,1]
S.D. [σ] for ATSA-MT-10	1.25	1	1	1	1.25	1	1	1.25	1	1
S.D. [σ] for ATSA-MT-6					1	1.25	1.5	1.25	1.5	1

Table 6: The starting Means and S.D.s of extra parameters in Equation 22 is provided. All initial Means are selected randomly from a uniform distribution whose limits are provided here.

	e	f	g	n	o	p	q	r	s
Range for Mean [μ]	$[\frac{-1}{4},\frac{1}{4}]$	$[\frac{-1}{4},\frac{1}{4}]$	$[\frac{-1}{4},\frac{1}{4}]$	$[\frac{-1}{5},\frac{1}{5}]$	$[\frac{-1}{4},\frac{1}{4}]$	$[\frac{-1}{5},\frac{1}{5}]$	$[\frac{-1}{4},\frac{1}{4}]$	$[\frac{-1}{2},\frac{1}{2}]$	[-2,2]
S.D. [σ] for ATSA-MT-19	0.75	0.75	0.75	0.75	0.75	0.75	0.75	1	1

The ATSA-MT Heuristic is divided into three separate flavors depending on which expression among equations 23, 24 or 25 is used for edge selections. The flavor of this Heuristic with Expression 1 is termed as ATSA-MT-19. Expression 2, although an oversimplification, provided very good results and is termed ATSA-MT-6. Expression 3 is used as another flavor of this Heuristic and is named ATSA-MT-10. Apart from the difference in the equations for the Edge selections, the 3 flavors of ATSA-MT are identical.

The starting values of the exponents and the weights in the above expressions were taken from a Normal distribution. At the start of the Heuristic, the Mean and Standard Deviations (S.D.) for all parameters in the considered expressions are provided as mentioned in Tables 5 and 6.

Whenever the Decoding Mechanism is able to provide a new best solution, the Mean of the distribution is shifted towards the new value which was able to provide the best solution. This shift is made dependant on the toughness of the solution obtained which is judged by the number of iterations (F) taken to obtain this new best solution.

When a new best solution is obtained from a Decoding Mechanism, the parameters of Table 5 have their mean and standard deviations updated. The sample expression for the updation of Mean (μ_a) and S.D. (σ_a) of the

first parameter a is shown in equations 25 and 26.

$$\mu_a^{new} = \mu_a + (a - \mu_a)\frac{F}{F_{iter}} \tag{25}$$

$$\sigma_a^{new} = \sigma_a \left(1 - \frac{F}{F_{iter}}\right) \tag{26}$$

F_{iter} is the maximum number of iterations without any improvement, after which the ATSA-MT Heuristic terminates. It is calculated as,

$$F_{iter} = 2000 \left(1 - \left(\frac{|N|}{1500}\right)^{0.1} + \left(1 - \frac{|N|}{1000}\right)^{5}\right) \tag{27}$$

where $|N|$ is the size of the problem, i.e. the number of Nodes / Relief Points. Such a function, as in Eq. 27 is used instead of using simpler linear functions like Eq. 28

$$F_{iter} = 7500 - 6|N| \tag{28}$$

to allow faster convergence for large instances which would otherwise take an extremely long time to terminate. For extremely high number of Nodes, Eq. 27 may be altered accordingly.
Two Decoding Mechanisms are constructed:

1. ODM: For each iteration of this Original Decoding Mechanism, the same parameter values are used during it's entirety for selecting all Daughter Edges. If the final objective value was found better than the best solution obtained as yet, then the above parameter update process as mentioned in Eq. 25 and Eq. 26 is followed.

2. RDM: This Randomised Decoding Mechanism generates new parameter values (from Normal Distribution based on the most recent updates in the Means and S.D.s of each parameter) during the Daughter Edge selection from the single Mother Graph. The parameter values cannot be updated in this case since the used parameter values would be distinct during each Daughter Edge selection.

Only a single Edge is selected in the ATSA-MT Heuristics from among the dump of all Daughter Edges.

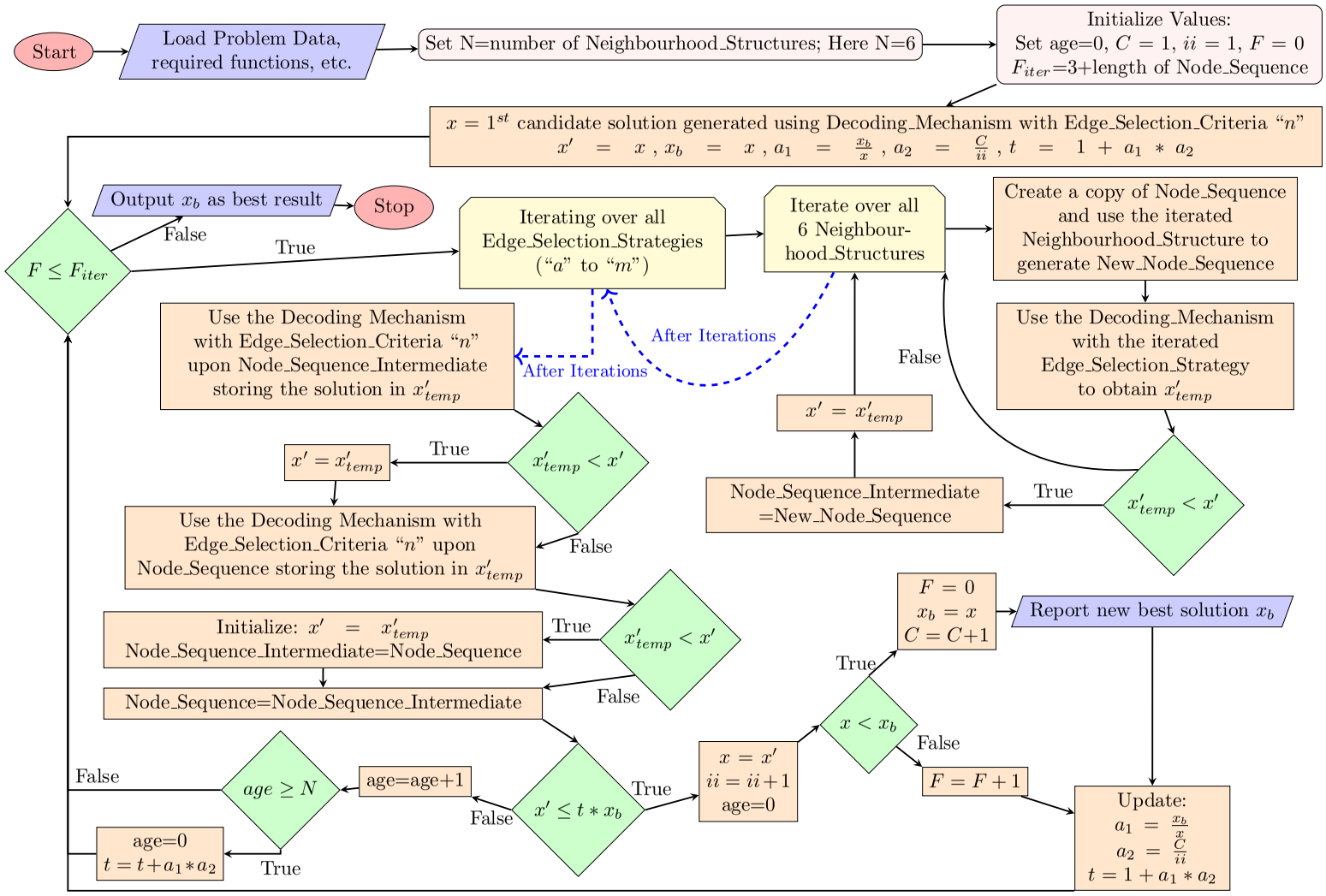


Figure 7: Flowchart representing ATSAH Logic (first 6 Neighbourhood Structures are used here)

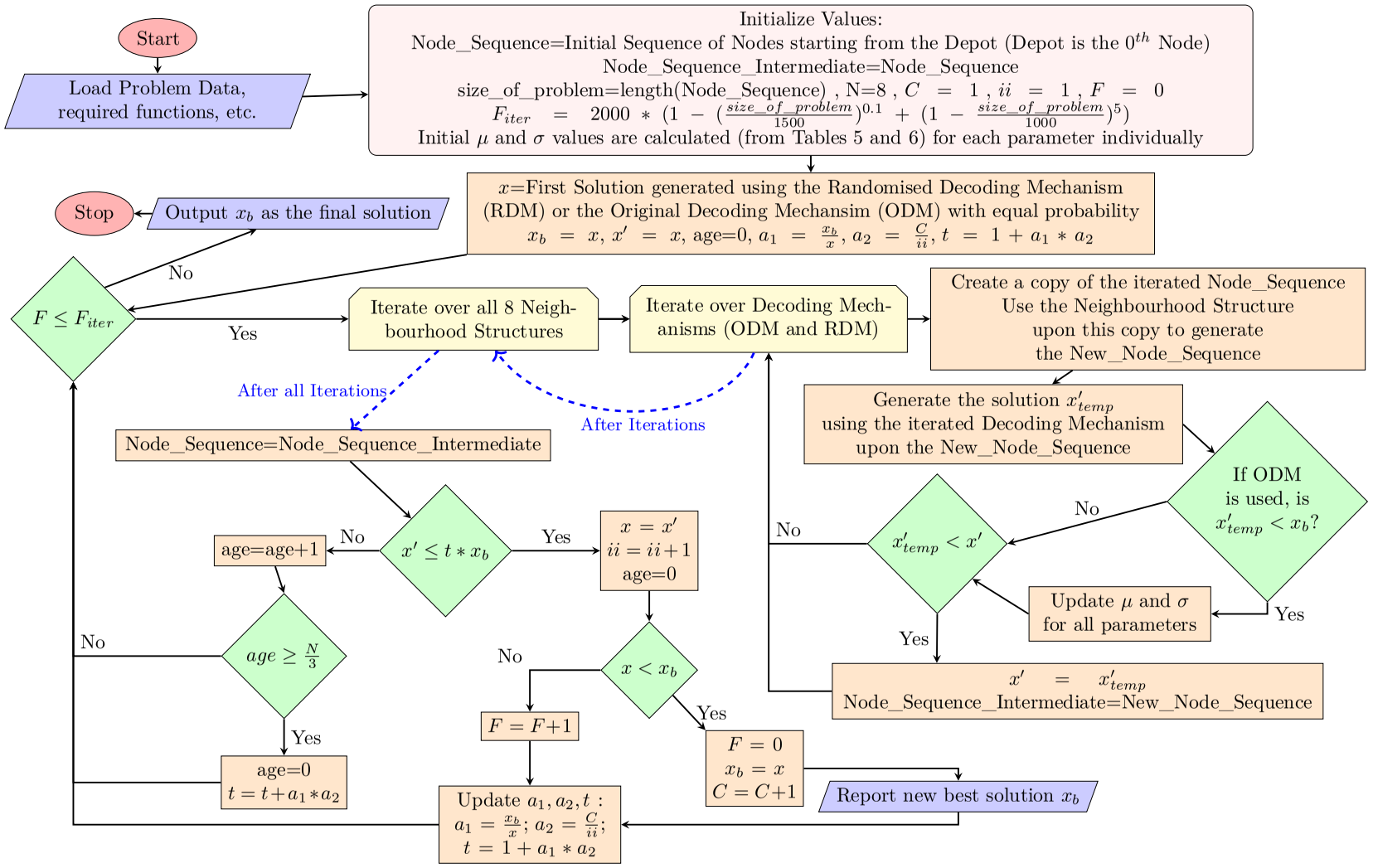


Figure 8: Flowchart representing ATSA-MT Logic (all 8 Neighbourhood Structures are used here)

4.7. Continuing the Decoding Mechanism

For each Mother Graph in Parent Graphs Set, we generate new Daughter Edges as follows: (Each Daughter Edge starts from the starting Node, and only uses each Vehicle Type remaining to be used; both these information being available in each Mother Graph.)

1. Iterate over available Vehicle Types (i.e. Vehicle Types which have positive number of Vehicles available to be used)

2. For each Vehicle Type available we start with creating the smallest Edge from the starting Node and keep increasing the Edge length keeping the same starting Node. In case there are no pre-existing Edges, the starting Node is the first Depot Node of the Node-sequence.
   The starting Node for a Daughter Edge is the same as the stopping Node of the last Edge of a Mother Graph.

3. The first Edge is constructed starting from the starting Node and ending at it's next Node in the Node Sequence. If this is found to be feasible, then this Edge is retained in a dump/set of Daughter Edges. We then include the next Node of the Node Sequence and see whether this increased Edge is feasible (i.e. the ending Node of this Edge is increased to be the next Node of the Node sequence). If this new Daughter Edge is also found feasible, it is similarly dumped into the set of Daughter Edges. This process of increasing the Edge Length by taking in more consecutive Nodes from the Node Sequence is carried out until we find an infeasible Edge. Once we find an infeasible Edge (i.e. an Edge whose underlying route is infeasible due to constraint violation), it is rejected, and we move on to the next Vehicle Type in the iteration.

4. It is to be noted that the starting Node has not yet changed. The starting Node and remaining Vehicles of each Type to be used, which are information specific to each Graph, are updated during addition of the Graph into the Next Parent Graphs Set (and this information may be dropped when adding any Graph into the Traversed Graphs Set). From the above iterations, we get multiple Edges from the same starting Node for some/all of the available Vehicle Types. All these Edges are dumped into the same set of Daughter Edges.

5. Some Daughter Edges are selected, using the Edge selection criterias, to be added to their respective copy of the Mother Graphs with updated information of left-over Vehicles yet to be used of each type and the new updated starting Node for subsequent Edge creation. After this process, the set of Daughter Edges is emptied.

6. These updated copies of Mother Graphs are added into Traversed Graphs Set if the Mother Graph contains all Nodes in the sequence, i.e. no Node is left to be traversed by any vehicle. Otherwise, they are added to Next Parent Graphs Set.

7. The Parent Graphs Set is now emptied and it takes all members from Next Parent Graphs Set for further generation of Daughter Edges. The set of Next Parent Graphs Set is then cleared.

The Decoding Mechanism is continued till no Mother Graphs exist. After that, each Final Graph from the Traversed Graphs Set is valued. The value of a Final Graph is the sum of the cost of all it's Edges, which is also the Objective Function Value. The lowest cost Final Graph is the output of the Decoding Mechanism.
For the ATSAH, the last Edge selection strategy (n) is used to generate the initial solution as well as once during each iteration of F as highlighted in the flowchart in Fig. 7.
The acceptance of a solution within a threshold range has been taken from Avci and Topaloglu (2016). The calculation of this threshold uses values of a_1 and a_2 where:

$$a_1 = \frac{x_b}{x} \tag{29}$$

$$a_2 = \frac{C}{ii} \tag{30}$$

Here ii keeps track of the number of solutions obtained within the threshold and C keeps track of the number of best solutions obtained. t is used to calculate the threshold and it is generally updated as

$$t = 1 + a_1 a_2 \tag{31}$$

, alongwith the updation of a_1 and a_2. However if the solution obtained is t times above the best solution x_b, consecutively for G iterations; then the threshold is increased by $a_1 a_2$ as in Eq. 32.

$$t = t + a_1 a_2 \tag{32}$$

where G is equal to the number of Neighbourhood Structures in case of ATSAH; and it is equal to $\frac{1}{3}$rd the number of Neighbourhood Structures for ATSA-MT (and ATSA-MT-JF). G is represented as *age* in the figures 7 and 8.

4.8. Comparing with existing Heuristics

The developed ATSA-MT and ATSAH is compared with Avci and Topaloglu (2016)'s ATS and HLS Heuristic as follows:

1. ATSAH considers a limited number of vehicles unlike ATS or HLS. ATSAH can also be made to work like HLS or ATS if the number of vehicles provided are very large. Considerable improvement may be seen in this case.

2. ATSAH creates Daughter Edges in the same way as they are being created in HLS or ATS. However, in HLS or ATS all the Daughter Edges are added to the same Graph. Since ATSAH considers a limited number of Vehicles, separate graphs are created with the information of available-v/s-utilized Vehicles for that Graph.

3. Due to the fact that separate Graphs are necessary for keeping track of the number of Vehicles used and left, we need to limit the number of Graphs being generated. This is because, each Graph has the potential of generating multiple other Graphs which can further generate multiple other Graphs and this cascading process would become memory intensive. Therefore the process of Next Mother Graph creation is limited by choosing a few of the best Edges (in case of ATSAH) or the best Daughter Edge(in case of ATSA-MT).

4. The selection strategy of choosing Daughter Edges is novel and multiple other logics could be introduced for extending the ATSAH. The ATSA-MT allows the algorithm to itself create a logic based on the solutions it has encountered and this ensures that the Heuristic is able to tune itself for any problem.

4.9. Incorporating a nature inspired JellyFishing search extending ATSA-MT

Extensions of ATSA-MT might be done by including a bounce (or multiple bounces), where a bounce would refer to reinstating the initial S.D. values in the ATSA-MT once $F = F_{iter}/2$ is reached and restarting the same heuristic run with $F = 0$. The limitations of the ATSA-MT Heuristic were observed when the parameter Standard Deviations take extremely low values of the order 10^{-26}. Such extremely low values reduce the search space of tuning an individual parameter within an extremely low region which might prove disadvantageous especially when a certain combination of the parameter values having the potential to provide the optimal solution for a specific Node Sequence is not able to converge into taking that combination due to the direction of the initial tuning process which led the parameter values into a ever-tightening sub-optimal region. Although this is also necessary for deep searches at times, such situations would be tough to escape. If the direction of the small changes in the Standard Deviation values is made to reverse or bounce everytime it reaches certain extremes or bounds, then the search technique might be improved. In case the S.D. values are made equal to their starting values in the middle of the Heuristic, (say when $F = \frac{F_{iter}}{2}$), this would potentially solve this problem, but the change would seem to happen with a jerk in the Standard Deviation.

Here, we take the inspiration of the motion[6] of the Bell of an immortal[7] jellyfish's hood and compare it with the S.D. curve. Just as the JellyFish propels water to move starting a Squeeze Phase, this ATSA-MT-JF heuristic would also start similarly with the initial standard deviation values. However, once the S.D. values of a parameter breaches the lower bound, the S.D. value of that parameter would start to increase just like the relax phase of the immortal jellyfish's Bell. This process continues indefinitely solving the problem of extensive (Relax Phase) -vs- intensive (Squeeze Phase) search.

To incorporate this, we introduce a new variable of Search Direction (D) for each of the parameters considered (maximum 19 possible parameters if Eq. 23 is used, with a search direction for each of those parameters). This Search Direction takes a value of 1 when the Standard Deviation of the corresponding parameter is less than 10^{-9}, indicating an increase in the S.D.

[6]https://www.newscientist.com/article/2264056-jellyfish-push-off-a-pocket-of-water-under-their-bell-to-swim-faster/#:~:text=Even%20though%20they%20lack%20fins,can%20propel%20themselves%20

[7]https://en.wikipedia.org/wiki/Turritopsis_dohrnii

of that parameter. When the Standard Deviation for a parameter is greater than 10^0, the corresponding Search Direction takes a value of -1, indicating decrease in it's S.D. In any other case, the value of any Search Direction remains unchanged.

The standard deviation values are checked, and each Search Direction is updated if necessary, after each new best solution find. The process for updating the mean values is further improved as shown in Eq. 33, so as to traverse about 90% of difference in the parameter value from its mean, within 10% of the maximum allowed iterations i.e. F_{iter}.

$$\mu_a^{new} = \mu_a + (a - \mu_a) \left(\frac{1 - 100^{-3F/F_{iter}} + \left(\frac{F}{F_{iter}}\right)^{\frac{1}{15}}}{2} \right)^{\frac{1}{2}} \tag{33}$$

An alternative formula for a similar function is provided in Eq. 34 which is indirectly obtained by solving Eq. 35 for p with the condition that at $x = 0.1$, $y = 0.9$; which may be compared to a non-linear interpolation; $x = \frac{F}{F_{iter}}$ being the fractional distance during the interpolation and y being the corresponding fractional value evaluation associated with x.

$$\mu_a^{new} = \mu_a + (a - \mu_a) \left(1 - 10^{\frac{\log_{10}\left(1 - \left(\frac{F}{F_{iter}}\right)^{\log_{10} 2}\right)}{\log_{10} 2}} \right) \tag{34}$$

$$(y-0)^p + (x-1)^p = 1^p \xrightarrow{Transforming} x^p + (1-y)^p = 1 \tag{35}$$

The new formula for updating the standard deviation of parameter a is shown in Eq. 36, where D_a is the Search Direction for parameter a, and $R(0,1)$ is a random number generator between the values of 0 and 1. The random numbers in the factor multiplied with the Search Direction help in desynchronizing the Squeeze and Relax phases of one JellyFish with the others, where the structure of the Hood/Bell of a single JellyFish represents the standard deviation of one parameter.

$$\sigma_a^{new} = \sigma_a \left(\frac{1 + D_a \frac{F - R(0,1)}{F_{iter} + R(0,1)}}{1 - D_a \frac{F - R(0,1)}{F_{iter} + R(0,1)}} \right) \tag{36}$$

As an alternative to Eq. 36; any of equations 37 or 38 may be used, utilizing either of the factors.

$$\sigma_a^{new} = \sigma_a \left(1 + D_a \frac{F - R(0,1)}{F_{iter} + R(0,1)}\right) \tag{37}$$

$$\sigma_a^{new} = \sigma_a / \left(1 - D_a \frac{F - R(0,1)}{F_{iter} + R(0,1)}\right) \tag{38}$$

5. Results and discussion

The formulation and developed heuristics were tested using Python 3.10.4. For all Exact Formulations, Gurobi Optimizer version 9.5.2 build v9.5.2rc0 (win64) having Academic license (for non-commercial use only) was used by setting parameter TimeLimit to 86400 (seconds).

The computed results are presented in [8] [9] Tables 7, 8, 9, 10 and 11. The number of vehicles used for each type are mentioned for each instance individually. The Vehicle Type parameters have been originally taken from Avci and Topaloglu (2016); where we have introduced a new parameter of p-value (Table 1) to account for Vehicle to Road compatibility and resulting detours taken by big vehicles. In each of the instances, the Depot (0^{th} Node) is not included while stating the No. of Nodes (problem size).

The datasets used for the computations are obtained/generated as below:

1. Table 7 uses the same data as indicated in Table 2. of (Avci and Topaloglu, 2016). The Nodes are progressively increased and all distances used are Euclidean. Figures 9a, 9b, 9c, 9d, 9e and 9f shows the progressive changes in the parameter values (from red to blue, through green) for the best outcomes among the 30 runs of each ATS-MT and ATSA-MT-JF Heuristic flavors for the 6^{th} instance (35 Nodes case). Fig. 9g represents the progressive slowdown in the rate of decrease in the best objective value, for best run among all heuristics. Fig. 9h

[8] # in the mentioned results represents no feasible solution could be found by Gurobi within the specified time limit of 24 hours for the respective exact formulations considered

[9] ## in the mentioned results represents the formulation did not run after 24 hours (possibly due to memory issues)

represents the changes in F untill it reaches the termination criteria of F_{iter} (corresponding to Fig. 9g). (We were able to find better objective value than previously reported in Avci and Topaloglu (2016) during both the exact and heuristic runs.)

2. For Table 8, the data is generated in a similar process as in Table 3 of (Avci and Topaloglu, 2016). The Latitude and Longitude values are generated randomly from a uniform distribution within the range of [0,50]; and the PickUp and Delivery values are generated randomly from a uniform distribution within the range [0,100]. The distances between all Nodes for all Vehicle Types are Euclidean.

3. Table 9 the data is generated in a similar process as for the Table 5 in (Avci and Topaloglu, 2016). The Latitude and Longitude values are generated randomly from a uniform distribution within the range of [0,100]; and the PickUp and Delivery values are generated randomly from a uniform distribution within the range [0,50]. Euclidean distances are calculated among the Nodes for all Vehicle Types used. The results for the largest instance for this case is presented in Fig. 11.

4. For Table 10; the base data of Table 2 in (Avci and Topaloglu, 2016) is altered such that the Laitude and PickUp values are interchanged as well as the Longitude and Delivery values are swapped. Here too, the nodes are progressively increased and the best heuristic results considering all 35 Nodes are presented in Fig. 10.

5. For Table 11, the Latitude, Longitude, PickUp and Delivery values are generated randomly from a uniform distribution within the range of [0,125]. In this case, the distances among Nodes are varied as the Vehicle Size increases, softly indicating that larger vehicles have less road options. This planar distance[10] (which also satisfies the triangular inequality for each individual Network layer) is calculated by varying the p value in the L^pNorm for respective Vehicle Types as available in Table 1. This introduction of varying p-values approximates the deviations in the actual road distance from the Euclidean distance. This assumption is practical due to large Vehicles not being able to ply on the smallest of roads and therefore might often have to detour via

[10]https://en.wikipedia.org/wiki/Minkowski_distance

compatible roads only. Assuming that the distance is to be calculated between points A and B, the calculation is explained in Eq. 39.

$$\begin{aligned} Minkowski\ Distance_{AB} = L^pNorm_{AB} = \\ ((|A_{Latitude} - B_{Latitude}|)^p + (|A_{Longitude} - B_{Longitude}|)^p)^{\frac{1}{p}} \end{aligned} \tag{39}$$

Substituting $p = 1$ provides the Manhattan distance between points A and B, and putting $p = 2$ gives the Euclidean distance. The results of the largest Instance for this data-type is presented in Fig. 12.

Due to the higher problem sizes requiring lot more time with previously discussed configurations of the Heuristic, the following changes were made for Instances 14 to 34:

1. ATSAH termination criteria was changed to $F_{iter} = 25 + \frac{|N|}{5}$
2. ATSA-MT and ATSA-MT-JF termination criterias were changed to Eq. 40

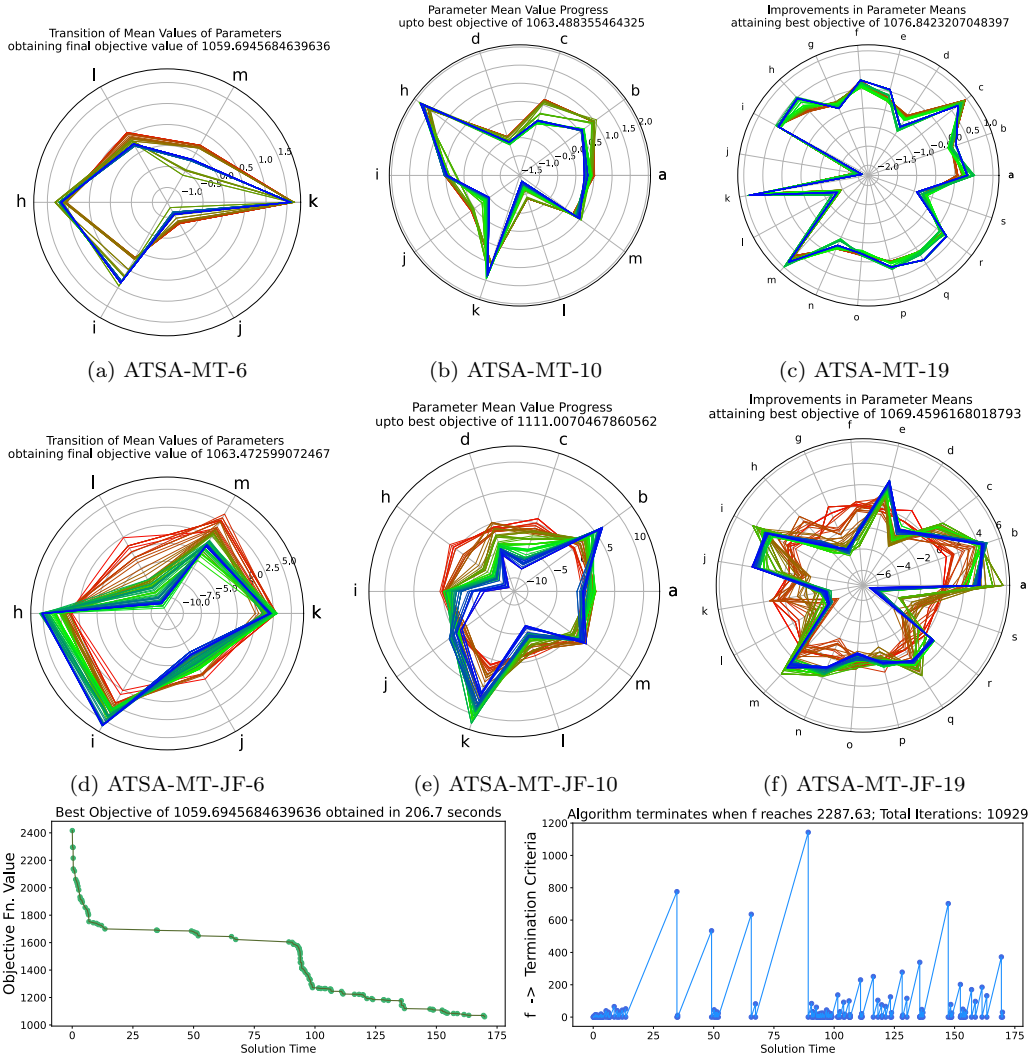
$$F_{iter} = 3250\left(1 - \left(\frac{|N|}{2750}\right)^{0.1} + \left(1 - \frac{|N|}{750}\right)^5\right) \tag{40}$$

3. The ATSAH probabilities for selecting more than one Daughter Edge was reduced for higher No. of Nodes since this caused the Heuristic to get stuck due to exponential increase in the probabilistic number of Mother Graphs being generated. Edge selection criteria (n) was retained with its probabilities, but the probability in (m), i.e. Random Acceptance 2, was changed from 10% to 1%. The probabilities in Edge Selection Criteria (k) was altered so that the probability of accepting the minimum costly Edge was 0.713 and that of selecting the second-minimum costly Edge was 0.314, while ensuring atleast one Edge was always selected.

5.1. Observations

Incase initial feasible solutions are not being found during generation of the first feasible solution even after using RDM and ODM with equal probabilities, a Neighbourhood Structure (like Shuffle Random or SubArray

Transplant) may be used on the original Node Sequence. For very large instances, the F_{iter} calculation needs to be altered so that the algorithm does not pre-terminate. In case the Heuristic needs to be run for a very long time, new additional satisfying termination criterias based on the rate of decrease of the objective function value may be used. During actual usage in real scenarios, it is recommended to run each of ATSA-MTs (ATSA-MT-6, ATSA-MT-10 and ATSA-MT-19) as well as each of ATSA-MT-JFs (ATSA-MT-JF-6, ATSA-MT-JF-10 and ATSA-MT-JF-19) once, and use the best solution among the six runs.

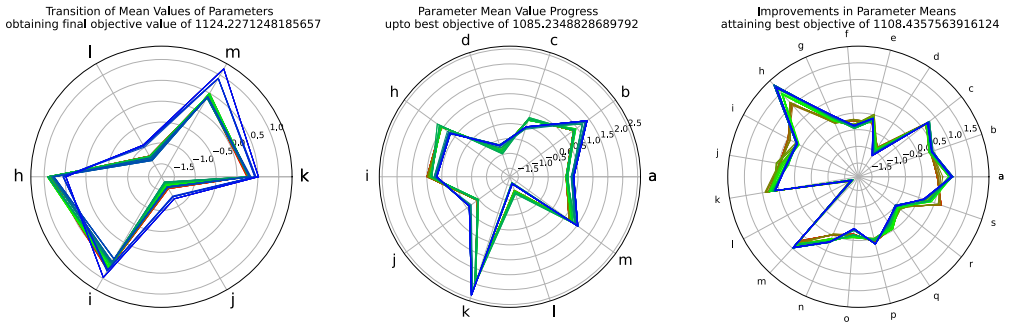


(a) ATSA-MT-6 (b) ATSA-MT-10 (c) ATSA-MT-19

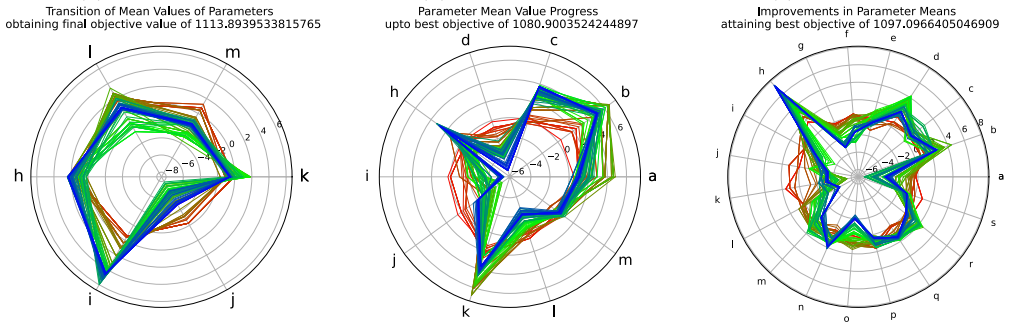
(d) ATSA-MT-JF-6 (e) ATSA-MT-JF-10 (f) ATSA-MT-JF-19

(g) Progressive slowdown of Objective decrease (h) Changes in F before it reaches the F_{iter} threshold

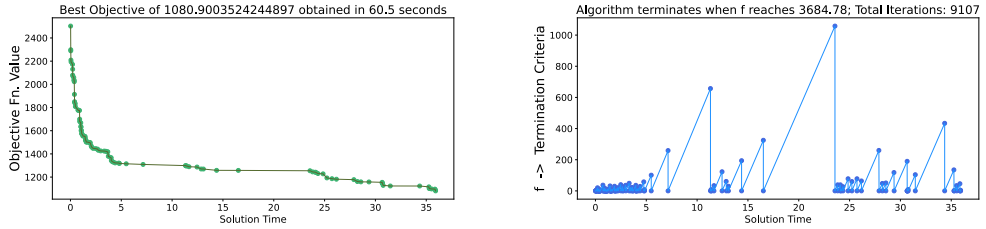
Figure 9: Best MachineTuning and JellyFishing results of the 6^{th} instance of 35 Nodes as in Table 7



(a) ATSA-MT-6 (b) ATSA-MT-10 (c) ATSA-MT-19

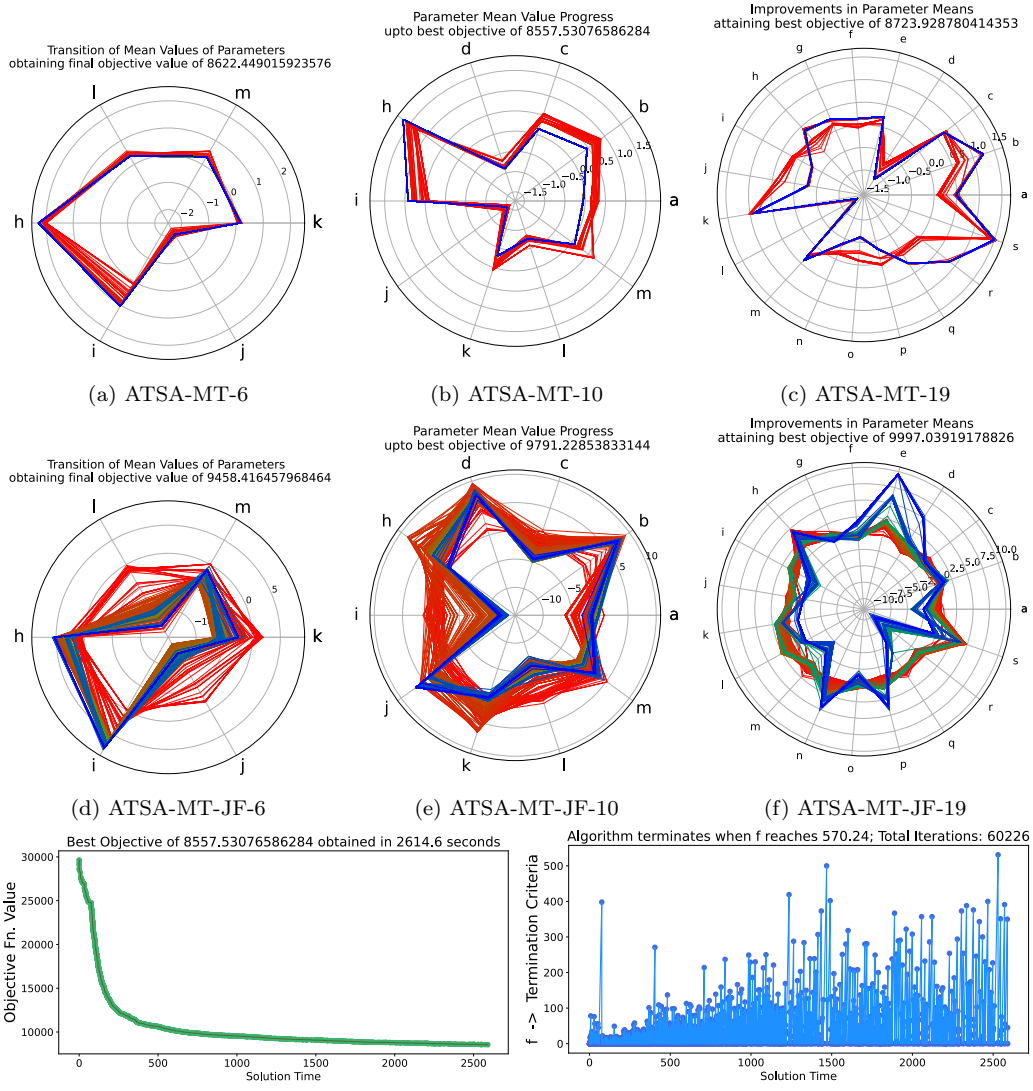


(d) ATSA-MT-JF-6 (e) ATSA-MT-JF-10 (f) ATSA-MT-JF-19



(g) Progressive slowdown of Objective decrease (h) Changes in F before it reaches the F_{iter} threshold

Figure 10: Best MachineTuning and JellyFishing results of Instance 23 in Table 10

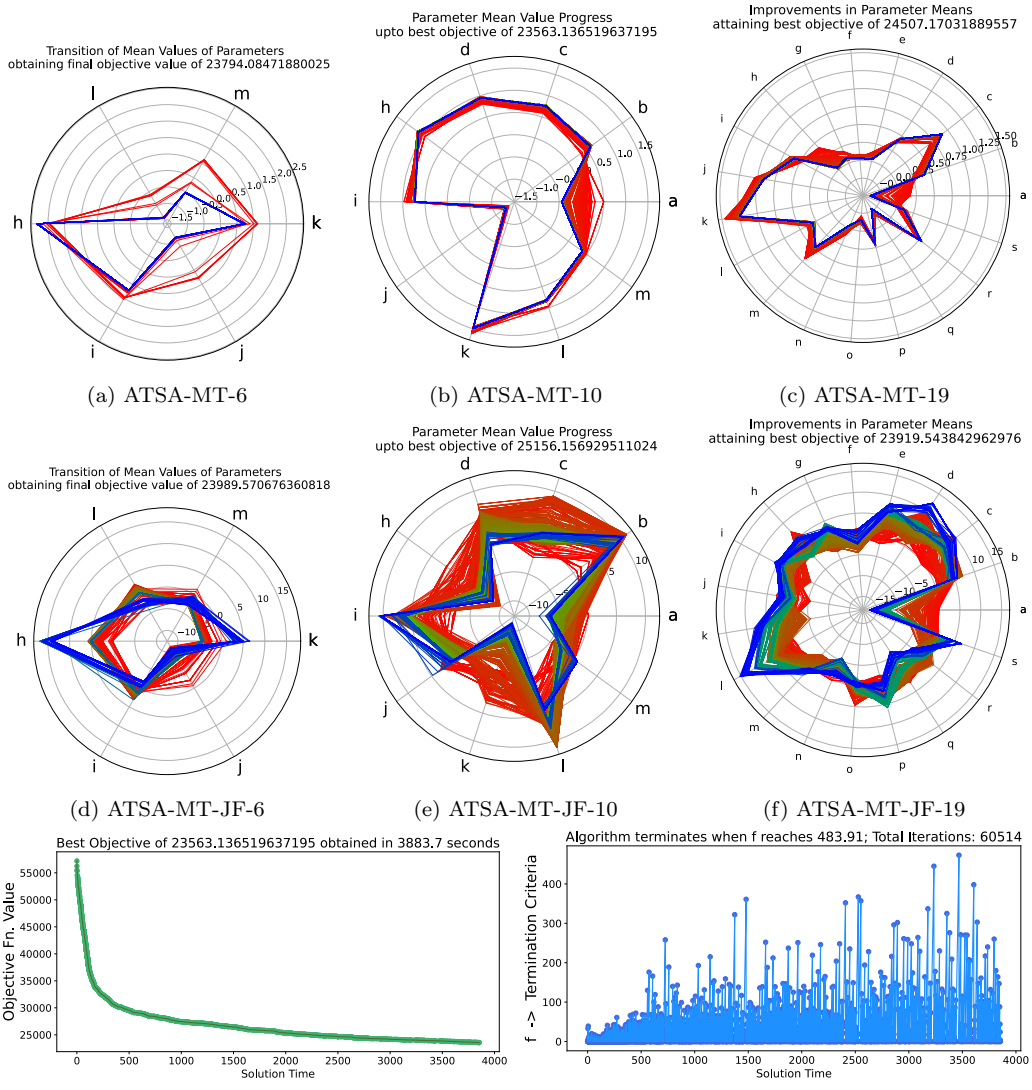


(a) ATSA-MT-6 (b) ATSA-MT-10 (c) ATSA-MT-19

(d) ATSA-MT-JF-6 (e) ATSA-MT-JF-10 (f) ATSA-MT-JF-19

(g) Progressive slowdown of Objective decrease (h) Changes in F before it reaches the F_{iter} threshold

Figure 11: Best MachineTuning and JellyFishing results of Instance 16 in Table 9



(a) ATSA-MT-6 (b) ATSA-MT-10 (c) ATSA-MT-19

(d) ATSA-MT-JF-6 (e) ATSA-MT-JF-10 (f) ATSA-MT-JF-19

(g) Progressive slowdown of Objective decrease (h) Changes in F before it reaches the F_{iter} threshold

Figure 12: Best MachineTuning and JellyFishing results of Instance 34 in Table 11

Table 7: Results from the data provided in Avci and Topaloglu (2016) Table 2 by sequentially considering upto the indicated Node No.s

(a) Results of the 4 Exact Formulations and ATSAHeuristic

Sl. No.	No. of Node	Considered Vehicle Types	Available Vehicles of each Type	Avci and Topaloglu (2016) (Max. Time = 86400 seconds)			Keçeci et al. (2021) (Max. Time = 86400 seconds)			Our Exact Formulation (Max. Time = 86400 seconds)			Formulating without Redundants (Max. Time = 86400 seconds)			ATSAH (30 runs)				
																Objective Value			Solution Time	
				Objective Value	Solution Time	MIP Gap	Objective Value	Solution Time	MIP Gap	Objective Value	Solution Time	MIP Gap	Objective Value	Solution Time	MIP Gap	Min.	Avg.	S.D.	Avg.	S.D.
1	10	1,2,3,4	1,1,1,1	442.61	1.22	0 %	442.61	1.07	0 %	442.61	0.64	0 %	442.61	0.63	0 %	442.61	444.42	4.22	0.51	0.15
2	15	1,2,3,4	1,1,1,1	615.05	14.21	0 %	615.05	2.09	0 %	615.05	17.49	0 %	615.05	2.88	0 %	615.05	665.62	33.93	1.34	0.39
3	20	1,2,3,4	2,2,2,2	721.75	6196.05	0 %	721.75	492.73	0 %	721.75	1315.28	0 %	721.75	2733.47	0 %	723.61	781.61	41.41	2.64	0.77
4	25	1,2,3,4	2,2,2,2	823.94	86400.48	5.33 %	823.94	25815.97	0 %	823.94	86400.77	2.28 %	823.94	50821.90	0 %	826.65	914.12	41.02	5.25	1.50
5	30	1,2,3,4	3,3,3,3	942.94	86400.28	12.49 %	904.65	86400.75	1.09 %	942.94	86400.47	7.11 %	918.92	86400.79	3.96 %	966.45	1134.54	110.86	8.26	2.77
6	35	1,2,3,4	3,3,3,3	1084.86	86400.28	11.99 %	1057.17	86400.4	4.44 %	1062.60	86400.48	6.87 %	1057.17	86400.58	4.63 %	1110.67	1333.36	143.87	11.36	4.62

(b) Results of the 3 Flavors of the ATSA-MachineTuning, each run 30 times for individual instances

Sl. No.	No. of Node	Considered Vehicle Types	Available Vehicles of each Type	ATSA-MT-6 (30 runs)					ATSA-MT-10 (30 runs)					ATSA-MT-19 (30 runs)				
				Objective Value			Solution Time		Objective Value			Solution Time		Objective Value			Solution Time	
				Min.	Avg.	S.D.	Avg.	S.D.	Min.	Avg.	S.D.	Avg.	S.D.	Min.	Avg.	S.D.	Avg.	S.D.
1	10	1,2,3,4	1,1,1,1	442.61	444.42	5.24	18.50	5.55	442.61	444.16	2.81	23.11	7.10	442.61	445.32	5.42	28.41	9.42
2	15	1,2,3,4	1,1,1,1	615.05	637.32	18.43	34.09	11.54	615.05	645.07	33.25	37.53	14.04	615.05	654.39	43.17	47.74	12.79
3	20	1,2,3,4	2,2,2,2	723.55	766.00	40.31	63.12	18.72	723.61	787.26	50.69	70.61	17.10	722.92	797.51	44.36	94.93	35.36
4	25	1,2,3,4	2,2,2,2	825.30	892.40	40.08	89.80	25.08	824.52	893.65	58.02	115.50	33.47	825.42	908.39	71.72	137.13	33.92
5	30	1,2,3,4	3,3,3,3	922.19	1055.44	103.30	137.95	47.92	938.65	1052.48	81.08	159.47	65.07	933.07	1041.36	83.85	196.73	64.11
6	35	1,2,3,4	3,3,3,3	1059.69	1137.53	34.64	162.74	64.95	1063.49	1136.19	36.79	228.52	63.13	1076.84	1137.69	41.51	245.44	71.56

(c) Results of the 3 Flavors of the ATSA-MT-JellyFishing, each run 30 times for individual instances

Sl. No.	No. of Node	Considered Vehicle Types	Available Vehicles of each Type	ATSA-MT-JF-6 (30 runs)					ATSA-MT-JF-10 (30 runs)					ATSA-MT-JF-19 (30 runs)				
				Objective Value			Solution Time		Objective Value			Solution Time		Objective Value			Solution Time	
				Min.	Avg.	S.D.	Avg.	S.D.	Min.	Avg.	S.D.	Avg.	S.D.	Min.	Avg.	S.D.	Avg.	S.D.
1	10	1,2,3,4	1,1,1,1	442.61	443.83	3.43	14.93	1.88	442.61	443.29	2.02	19.68	4.37	442.61	443.29	2.02	23.51	4.82
2	15	1,2,3,4	1,1,1,1	615.05	627.41	18.15	26.65	7.45	615.05	634.56	17.24	30.37	9.37	615.05	637.27	26.53	38.05	10.49
3	20	1,2,3,4	2,2,2,2	722.92	775.64	37.27	38.88	13.50	723.61	767.08	42.58	41.28	15.50	723.55	776.25	48.09	65.16	20.66
4	25	1,2,3,4	2,2,2,2	826.36	910.91	62.91	53.95	16.46	824.39	910.95	57.67	68.06	20.72	824.96	908.25	57.35	85.00	36.17
5	30	1,2,3,4	3,3,3,3	921.91	1011.86	56.10	80.82	29.76	933.07	1058.96	78.24	92.01	31.11	907.46	1058.21	92.39	134.99	64.29
6	35	1,2,3,4	3,3,3,3	1063.47	1188.57	93.06	96.67	44.92	1111.01	1255.85	129.34	129.90	59.20	1069.46	1204.11	98.43	170.09	82.66

[Computed on Intel® x64-based processor Xeon® W-2133, 3.6 GHz, Installed RAM 16.0 GB (15.7 GB usable) having 64-bit Windows 11 Pro Operating System for Workstations (Version:21H2 ; OS Build:22000.795), manufactured by Dell®]

Table 8: Computed results from data similarly generated as in Avci and Topaloglu (2016) Table 3

(a) Results of the 4 Exact Formulations (allowed to run upto 86400 seconds) and ATSAHeuristic

Sl. No.	No. of Node	Vehicle Types	No. of Vehicle	Avci and Topaloglu (2016)			Keçeci et al. (2021)			Our Exact Formulation WITH Redundant constraints 10 and 11			Our Exact Formulation WITHOUT Redundant constraints 10 and 11			ATSAH (30 runs)				
																Objective Value			Solution Time	
				Objective Value	Solution Time	Gap % or Best Bound	Objective Value	Solution Time	Gap % or Best Bound	Objective Value	Solution Time	Gap % or Best Bound	Objective Value	Solution Time	Gap % or Best Bound	Min.	Avg.	S.D.	Avg.	S.D.
7	50	2	10	986.81	86400.28	7.69%	959.37	86400.4	2.25%	961.87	86400.13	2.46%	980.36	86400.21	4.26%	1024.1	1139.2	50.11	10.53	2.96
8	75	3	12	#	86400.19	1392.83	1517.54	86400.22	3.52%	1525.08	86400.2	3.91%	1514.39	86400.21	3.35%	1747.19	1929.44	192.48	31.5	16.49
9	100	2	25	#	86400.72	1854.52	2218.6	86400.22	11.12%	2179.41	86400.24	9.7%	2179.23	86400.18	9.76%	2558.75	2780.24	140.46	125.16	46.04
10	150	1,2,3,4	9,9,9,9	#	86400.42	2396.76	3532.33	86400.19	29.04%	#	86400.27	2470.99	3488.62	86400.54	28.31%	3672.84	3888.88	155.55	804.61	288.91
11	200	1,3	12,13	#	86400.5	2738.89	#	86400.16	2881	#	86400.29	2846.29	#	86400.7	2868.46	4132.25	4755.31	505	3756.47	1403.36

(b) Results of the 3 Flavors of the ATSA-MachineTuning, each run 30 times for individual instances

Sl. No.	No. of Node	Considered Vehicle Types	Available Vehicles of each Type	ATSA-MT-6 (30 runs)					ATSA-MT-10 (30 runs)					ATSA-MT-19 (30 runs)				
				Objective Value			Solution Time		Objective Value			Solution Time		Objective Value			Solution Time	
				Min.	Avg.	S.D.	Avg.	S.D.	Min.	Avg.	S.D.	Avg.	S.D.	Min.	Avg.	S.D.	Avg.	S.D.
7	50	2	10	1004.92	1054.4	26.96	79.32	18.93	1017.56	1064.38	28.84	91.13	24.17	989.18	1071.09	72.15	111.39	33.12
8	75	3	12	1588.54	1739.9	127.35	159.32	77.47	1583.86	1732.33	97.59	190.59	67.25	1579.24	1701.69	96.54	228.98	66.83
9	100	2	25	2319.22	2445.02	61.33	305.14	73.23	2357.58	2523.06	229.95	320.22	112.85	2399.97	2516.36	142.37	432.53	158.61
10	150	1,2,3,4	9,9,9,9	3096.5	4075.91	1181.16	1076.28	731.14	3104.97	3841.33	1042.9	1461.97	855.74	3102.44	4031.03	1141.13	1629.93	1025.97
11	200	1,3	12,13	3808.99	4252.24	726.04	1050.65	529.73	3746.99	4494.32	995.84	1157.62	689.41	3817.47	4146.35	626.89	1642.35	636.37

(c) Results of the 3 Flavors of the ATSA-MT-JellyFishing, each run 30 times for individual instances

Sl. No.	No. of Node	Considered Vehicle Types	Available Vehicles of each Type	ATSA-MT-JF-6 (30 runs)					ATSA-MT-JF-10 (30 runs)					ATSA-MT-JF-19 (30 runs)				
				Objective Value			Solution Time		Objective Value			Solution Time		Objective Value			Solution Time	
				Min.	Avg.	S.D.	Avg.	S.D.	Min.	Avg.	S.D.	Avg.	S.D.	Min.	Avg.	S.D.	Avg.	S.D.
7	50	2	10	1026.04	1062.88	25.1	35.87	9.64	990.08	1063.15	41.49	41.87	12.02	1000.82	1056.56	39.6	53.41	14.09
8	75	3	12	1613.41	1717.86	75.8	76.43	24.55	1591.73	1716.52	68.61	81.12	38.35	1608.64	1732.75	87.98	113.98	33.24
9	100	2	25	2370.69	2489.25	71.57	159.01	50.41	2329.35	2473.35	69.11	202.13	58.23	2352.61	2467.04	63.81	251.16	76.66
10	150	1,2,3,4	9,9,9,9	3076.9	3705.05	831.76	634.33	321.79	3122.78	3482.54	530.77	984.21	435	3140.84	3458.85	499.02	1120.96	458.88
11	200	1,3	12,13	3741.05	3989.53	289.02	609.05	195.05	3812.22	3953.94	85.01	697.43	196.76	3772.79	4062.33	541.85	862.55	270.51

[Computed on Intel® x64-based processor Xeon® W-2133 and W-2223, 3.6 GHz, Installed RAM 16.0 GB (15.7 GB usable) having 64-bit Windows 11 Pro Operating System for Workstations (Version:21H2 ; OS Build:22000.795 and 22000.856), manufactured by Dell®]

Table 9: Computed results from data similarly generated as in Avci and Topaloglu (2016) Table 5

(a) Results of the 4 Exact Formulations (allowed to run upto 86400 seconds) and ATSAHeuristic

Sl. No.	No. of Node	Vehicle Types	No. of Vehicle	Avci and Topaloglu (2016)			Keçeci et al. (2021)			Our Exact Formulation WITH Redundant constraints 10 and 11			Our Exact Formulation WITHOUT Redundant constraints 10 and 11			ATSAH (30 runs)				
																Objective Value			Solution Time	
				Objective Value	Solution Time	Gap % or Best Bound	Objective Value	Solution Time	Gap % or Best Bound	Objective Value	Solution Time	Gap % or Best Bound	Objective Value	Solution Time	Gap % or Best Bound	Min.	Avg.	S.D.	Avg.	S.D.
12	250	3	29	#	86401.03	2269.31	3370.71	86400.21	10.66%	3436.27	86400.28	12.38%	3395.40	86400.22	11.64%	5253.99	8076.05	1910.59	419.48	380.83
13	300	2	36	##			4263.32	86400.74	14.71%	4296.32	86400.39	15.37%	4222.99	86400.26	13.92%	6173.18	9655.58	2384.47	3490.07	3273.68
14	350	2,3	20,20	##			10711.20	86401.19	63.04%	#	86400.73	3933.09	#	86400.67	3934.35	9106.10	10228.43	470.51	1695.35	421.29
15	400	2	45	##			5913.16	86401.01	21.3%	5742.34	86400.55	19.26%	5874.51	86400.56	21.11%	11015.53	14384.79	2861.96	775.64	493.97
16	450	3	45	##			7441.60	86400.48	27.4%	6999.94	86400.12	23.13%	6912.91	86401.01	22.22%	11946.55	17602.65	4519.23	572.66	521.24

(b) Results of the 3 Flavors of the ATSA-MachineTuning, each run 30 times for individual instances

Sl. No.	No. of Node	Considered Vehicle Types	Available Vehicles of each Type	ATSA-MT-6 (30 runs)					ATSA-MT-10 (30 runs)					ATSA-MT-19 (30 runs)				
				Objective Value			Solution Time		Objective Value			Solution Time		Objective Value			Solution Time	
				Min.	Avg.	S.D.	Avg.	S.D.	Min.	Avg.	S.D.	Avg.	S.D.	Min.	Avg.	S.D.	Avg.	S.D.
12	250	3	29	4072.1	5227.23	1410.6	650.96	266.86	4241.95	4869.09	361.23	788.98	213.55	4367.39	4888.24	293.06	886.67	246.05
13	300	2	36	5465.4	6321.13	1208.18	1270.52	364.61	5436.88	6550.32	1927.95	1300.25	470.12	5342.47	6651.93	1916.12	1671.04	670
14	350	2,3	20,20	6540.63	9569.26	4726.29	3335.22	1741.95	6422.75	10510	5348.4	3379.36	2291.31	6479.06	9021.92	4325.6	4612.58	2207.18
15	400	2	45	7052.21	8418.26	1768.66	2788.09	975.03	7335.76	9954.05	4465.37	2696.67	1182.88	7566.61	9047.98	2966.23	3624.89	1159.27
16	450	3	45	8622.45	11284.3	4612.58	1508.3	646.97	8557.53	10946.71	4383.58	1643.31	680.3	8723.93	13013.39	6078.06	1539.93	867.94

(c) Results of the 3 Flavors of the ATSA-MT-JellyFishing, each run 30 times for individual instances

Sl. No.	No. of Node	Considered Vehicle Types	Available Vehicles of each Type	ATSA-MT-JF-6 (30 runs)					ATSA-MT-JF-10 (30 runs)					ATSA-MT-JF-19 (30 runs)				
				Objective Value			Solution Time		Objective Value			Solution Time		Objective Value			Solution Time	
				Min.	Avg.	S.D.	Avg.	S.D.	Min.	Avg.	S.D.	Avg.	S.D.	Min.	Avg.	S.D.	Avg.	S.D.
12	250	3	29	4112.51	4976.22	333.27	634.69	157.72	4470.04	5081.84	787.04	686.72	199.72	4571.93	4930.63	214.06	876.37	176.31
13	300	2	36	5493.88	6008.62	263.08	1509.29	383.03	5641.18	6445.89	1494.81	1477.53	366.87	5591.73	6068.08	282.07	2050.55	528.88
14	350	2,3	20,20	7081.49	8367.31	2404.27	3450.08	1266.13	6568.84	7940.89	2189.62	3464.22	1108.05	6825.33	8258.74	2992.47	3697.12	1220.75
15	400	2	45	7738.94	8929.14	2591.16	2338.06	748.61	7558.17	8559.39	410.67	2381.78	571.98	8005.76	9143.1	2483.81	2672.46	856.52
16	450	3	45	9458.42	11236.81	2548.35	785.42	280.74	9791.23	11931.27	3991.71	813.9	336.77	9997.04	11870.49	3169.98	917.6	374.56

[Computed on Intel® x64-based processor Xeon® W-2133 and W-2223, 3.6 GHz, Installed RAM 16.0 GB (15.7 GB usable) having 64-bit Windows 11 Pro Operating System for Workstations (Version:21H2 ; OS Build:22000.795 and 22000.856), manufactured by Dell®]

Table 10: Computed results from the data provided in Avci and Topaloglu (2016) Table 2 by sequentially considering upto the indicated Node No.s while using the p values from Table 1 for the Vehicle Type based Node-to-Node distance calculation

(a) Results of the 4 Exact Formulations and ATSAHeuristic

Sl. No.	No. of Node	Considered Vehicle Types	Available Vehicles of each Type	Avci and Topaloglu (2016) (Max. Time = 86400 seconds)			Keçeci et al. (2021) (Max. Time = 86400 seconds)			Our Exact Formulation (Max. Time = 86400 seconds)			Formulating without Redundants (Max. Time = 86400 seconds)			ATSAH (30 runs)				
				Objective Value	Solution Time	MIP Gap	Objective Value	Solution Time	MIP Gap	Objective Value	Solution Time	MIP Gap	Objective Value	Solution Time	MIP Gap	Objective Value			Solution Time	
																Min.	Avg.	S.D.	Avg.	S.D.
17	5	1,2,3,4	1,1,1,1	341	0.09	0%	341	0.14	0%	341	0.16	0%	341	0.19	0%	341	341	0	0.16	0.01
18	10	1,2,3,4	2,2,2,2	508.55	4.78	0%	508.55	2.7	0%	508.55	1.36	0%	508.55	5.09	0%	508.55	525.04	14.35	0.58	0.15
19	15	1,2,3,4	3,3,3,3	606.33	1047.03	0%	606.33	30.79	0%	606.33	85.57	0%	606.33	57.62	0%	606.33	655.84	39.89	0.89	0.35
20	20	1,2,3,4	4,4,4,4	700.73	86400.34	3.94%	700.73	173.24	0%	700.73	2159.34	0%	700.73	1504.72	0%	741.29	856.65	76.86	1.99	1.03
21	25	1,2,3,4	3,3,3,3	808.03	86400.1	5.18%	808.03	2544.06	0%	808.03	45808.81	0%	808.03	5170.14	0%	920.35	1048.25	74.36	2.68	1.12
22	30	1,2,3,4	2,2,2,2	948.34	86400.84	7.24%	974.74	86403.46	6.97%	974.74	86403.43	9.51%	937.95	86400.78	2.13%	1022.71	1210.66	121.88	3.68	1.58
23	35	1,2,3,4	1,1,1,1	1080.9	86401.82	8.3%	1080.9	86400.77	7.31%	1106.42	86400.31	10.8%	1079.86	86402.33	8.45%	1148.69	1334.37	89.23	3.92	1.61

(b) Results of the 3 Flavors of the ATSA-MachineTuning, each run 30 times for individual instances

Sl. No.	No. of Node	Considered Vehicle Types	Available Vehicles of each Type	ATSA-MT-6 (30 runs)					ATSA-MT-10 (30 runs)					ATSA-MT-19 (30 runs)				
				Objective Value			Solution Time		Objective Value			Solution Time		Objective Value			Solution Time	
				Min.	Avg.	S.D.	Avg.	S.D.	Min.	Avg.	S.D.	Avg.	S.D.	Min.	Avg.	S.D.	Avg.	S.D.
17	5	1,2,3,4	1,1,1,1	341	341	0	7.84	0.46	341	341	0	9.43	0.56	341	341	0	11.77	0.75
18	10	1,2,3,4	2,2,2,2	508.55	514.13	5.01	23.36	8.25	508.55	518.13	9.27	31.05	11.13	508.55	517.21	9.01	33.09	8.98
19	15	1,2,3,4	3,3,3,3	606.33	634.97	32.51	43.51	18.86	606.33	635.95	36.23	52.94	15.28	606.33	636.2	33.44	70.16	17.9
20	20	1,2,3,4	4,4,4,4	734.67	795.12	37.17	56.68	17.2	700.73	784.99	31.88	77.71	24.12	725.36	789.87	33.74	92.62	24.3
21	25	1,2,3,4	3,3,3,3	808.03	948.38	71.03	81.36	22.11	808.03	937.48	81.4	103.49	29.64	859.38	975.3	80.95	127.39	39.11
22	30	1,2,3,4	2,2,2,2	952.39	1098	100.48	93.75	37.88	999.33	1152.13	141.39	99.37	34.3	983.37	1113.81	63.68	146.2	43.51
23	35	1,2,3,4	1,1,1,1	1124.23	1231.86	101.04	67.96	22.91	1085.23	1220.79	49.84	72.21	20.99	1108.44	1216.86	87.07	84.92	39.65

(c) Results of the 3 Flavors of the ATSA-MT-JellyFishing, each run 30 times for individual instances

Sl. No.	No. of Node	Considered Vehicle Types	Available Vehicles of each Type	ATSA-MT-JF-6 (30 runs)					ATSA-MT-JF-10 (30 runs)					ATSA-MT-JF-19 (30 runs)				
				Objective Value			Solution Time		Objective Value			Solution Time		Objective Value			Solution Time	
				Min.	Avg.	S.D.	Avg.	S.D.	Min.	Avg.	S.D.	Avg.	S.D.	Min.	Avg.	S.D.	Avg.	S.D.
17	5	1,2,3,4	1,1,1,1	341	341	0	7.36	0.53	341	341	0	8.48	0.71	341	341	0	10.76	0.68
18	10	1,2,3,4	2,2,2,2	508.55	514.56	7.84	18.72	5.41	508.55	517.95	13.01	20.86	8.45	508.55	519.86	11.9	26.24	10.11
19	15	1,2,3,4	3,3,3,3	606.33	627.78	32.26	28.26	9.6	606.33	637.57	35.34	32.08	11.38	606.33	631.6	25.76	38.11	12.76
20	20	1,2,3,4	4,4,4,4	749.12	786.1	33.16	35.77	7.92	749.12	788.11	28.5	43.08	12.27	720.34	787.38	28.85	53.09	17.14
21	25	1,2,3,4	3,3,3,3	826.28	927.96	50.26	50.79	19.86	808.03	920.91	61.23	56.46	21.1	841.75	946.58	61.2	65.88	21.55
22	30	1,2,3,4	2,2,2,2	996.35	1082.18	56.57	61.57	29.57	972.97	1096.22	97	71.1	29.4	989.63	1111.8	72.52	104.66	38.05
23	35	1,2,3,4	1,1,1,1	1113.89	1209.75	53.41	37.25	12.07	1080.9	1191.1	51.27	46.55	13.16	1097.1	1203.53	46.21	46.8	14.25

[Computed on Intel® Core(TM), i5-8500 CPU @ 3 GHz, x64-based processor with installed RAM 8 GB (7.78 GB usable) having 64-bit Windows 11 Pro Operating System (Version:21H2 ; OS Build:22000.978), manufactured by HP®Inc.] (HP ProDesk 600 G4 PCI MT)

Table 11: Results from generated data while using the p values from Table 1 for the Vehicle Type based Node-to-Node distance calculation

(a) Results of the 4 Exact Formulations and ATSAHeuristic

Sl. No.	No. of Node	Considered Vehicle Types	Available Vehicles of each Type	Avci and Topaloglu (2016) (Max. Time = 86400 seconds)			Keçeci et al. (2021) (Max. Time = 86400 seconds)			Our Exact Formulation (Max. Time = 86400 seconds)			Formulating without Redundants (Max. Time = 86400 seconds)			ATSAH (30 runs)				
																Objective Value			Solution Time	
				Objective Value	Solution Time	MIP Gap	Objective Value	Solution Time	MIP Gap	Objective Value	Solution Time	MIP Gap	Objective Value	Solution Time	MIP Gap	Min.	Avg.	S.D.	Avg.	S.D.
24	66	1,2,3	5,7,6	#	86400.96	2352.05	2624.21	86400.15	6.05%	2684.73	86400.29	8.63%	2731.35	86400.28	9.62%	3679.22	4150.36	342.79	11.49	5.55
25	128	2,3,4	11,8,10	#	86400.49	3567.93	4923.3	86400.15	22.21%	4818.19	86400.23	20.94%	4731.3	86401	19.73%	6870.95	7526.1	472.36	68.72	20.02
26	175	1,3,4	21,19,31	##			7084.43	86400.28	18.88%	#	86400.19	5665.89	8290.42	86400.26	31.53%	10637.69	11939.18	1684.11	105.32	52.17
27	225	1,2,4	23,26,13	##			#	86400.26	6251.08	#	86400.44	6138.32	#	86400.35	6138.88	12062.85	13473.09	681.63	228.25	70.5
28	275	2,3	32,38	##			#	86400.23	9208.92	#	86400.36	9079.22	#	86400.35	9054.96	15795.91	16679.7	618.82	370.51	78.04
29	325	1,2,3,4	15,20,20,15	##			#	86400.5	8283.11	#	86400.6	8070.6	#	86400.59	8072.52	16781.82	18175.22	919.37	806.69	187.38
30	375	1,4	22,33	##			22023.72	86400.91	42.9%	#	86400.42	12399.8	#	86400.49	12384.4	22119.41	23917.77	1094.28	711.54	177.44
31	425	2,3,4	19,14,30	##			#	86400.51	11558.73	#	86400.83	11246.7	#	86400.83	11246.75	23116.64	24807.66	1080.41	1273.73	421.15
32	475	1,2,3	31,28,25	##			#	86400.58	15498.1	#	86401.05	14983.72	#	86401.04	14985.31	27750.58	29783.14	884.84	1775.63	531.99
33	512	1,2,4	25,27,29	##			#	86400.27	15815.61	#	86401.46	15401.49	#	86401.33	15409.97	30556.27	31895.66	1154.87	1998.59	474.13
34	555	1,3,4	23,27,25	##			#	86400.5	14464.41	#	86401.22	14120.19	#	86401.17	14122.21	31956.18	40626.26	9979.12	1413.6	1146.23

(b) Results of the 3 Flavors of the ATSA-MachineTuning, each run 15 times for individual instances

Sl. No.	No. of Node	Considered Vehicle Types	Available Vehicles of each Type	ATSA-MT-6 (30 runs)					ATSA-MT-10 (30 runs)					ATSA-MT-19 (30 runs)				
				Objective Value			Solution Time		Objective Value			Solution Time		Objective Value			Solution Time	
				Min.	Avg.	S.D.	Avg.	S.D.	Min.	Avg.	S.D.	Avg.	S.D.	Min.	Avg.	S.D.	Avg.	S.D.
24	66	1,2,3	5,7,6	3002.29	3179.17	145.64	212.28	51	2945.57	3357.17	458.95	232.92	100.64	3004.95	3239.03	187.71	277.16	92.77
25	128	2,3,4	11,8,10	5406.5	6292.36	1699.85	692.16	321.44	5166.06	6354.71	1568.45	651.12	302.07	5259.34	6052.26	1274.78	1010	424.37
26	175	1,3,4	21,19,31	7896.64	10294.71	3057.49	1063.48	699.69	8094.68	10625.97	3006.8	1175.48	882.22	7820.07	10964.51	3268.95	1534.91	1140.45
27	225	1,2,4	23,26,13	9903.23	13853.55	3841.01	968.64	934.83	9840.43	12941.29	3539.39	1343.25	1094.46	9654.5	13982.54	4202.75	1201.8	1034.66
28	275	2,3	32,38	12570.27	17038.9	5360.46	844.96	652.94	12308.12	18510.47	6011.57	924.41	806.68	12539.19	15566.77	5126.48	1766.36	959.47
29	325	1,2,3,4	15,20,20,15	13203.62	16239.98	4777.53	2490.94	1272.21	12779.87	18447.23	7102.38	2350.43	1606.47	12462.32	15781.03	5168.48	3434.29	1759.15
30	375	1,4	22,33	18112.37	27448.88	9271.27	863	858.87	17608.23	25590.11	9689.21	1380.88	1016.13	18343.26	27987.02	9874.16	1153.3	1052.61
31	425	2,3,4	19,14,30	18151.18	31566.42	10355.37	1179.52	1428.94	18305.79	28696.2	10274.69	1428.36	1286.63	18253.74	31685.76	10204.96	1368.85	1650.48
32	475	1,2,3	31,28,25	22590.61	37445.19	9968.49	901.8	1200.42	22349.67	33107.41	10121.17	1561.11	1532.39	23189.02	36851.46	9549.98	1096.13	1420.1
33	512	1,2,4	25,27,29	24148.51	41620.74	11851.18	887.72	1162.7	24863.55	41672.76	13130.8	1159.93	1398.51	25083.94	45509.04	10163.18	888.87	1462.66
34	555	1,3,4	23,27,25	23794.08	40096.06	12993.4	1308.84	1411.79	23563.14	39968.12	12894.71	1457.64	1721.61	24507.17	37450.72	13165.8	1700.94	1626.91

(c) Results of the 3 Flavors of the ATSA-MT-JellyFishing, each run 15 times for individual instances

Sl. No.	No. of Node	Considered Vehicle Types	Available Vehicles of each Type	ATSA-MT-JF-6 (30 runs)					ATSA-MT-JF-10 (30 runs)					ATSA-MT-JF-19 (30 runs)				
				Objective Value			Solution Time		Objective Value			Solution Time		Objective Value			Solution Time	
				Min.	Avg.	S.D.	Avg.	S.D.	Min.	Avg.	S.D.	Avg.	S.D.	Min.	Avg.	S.D.	Avg.	S.D.
24	66	1,2,3	5,7,6	2967.2	3238.15	177.29	141.34	55.01	2944.27	3286.33	255.98	150.51	55.66	3030.66	3277.56	436.29	224.43	74.2
25	128	2,3,4	11,8,10	5358.02	5679.75	447.12	713.6	224.56	5143.93	5563.96	218.43	749.89	164.31	5225.59	5585.57	213.63	917	267.16
26	175	1,3,4	21,19,31	7878.29	8798.39	687.03	1269.97	391.53	8082.82	8661.4	299.01	1236.18	331.89	7879.45	8811.09	322.07	1600	505.21
27	225	1,2,4	23,26,13	9984.45	10825.25	1108.73	1520.8	612.59	9868.54	10702.55	1118.36	1785.67	604.18	9356.55	10440.39	1279.99	2096.64	716.29
28	275	2,3	32,38	12791.69	15080.29	3937.29	1088.41	457.75	12661.29	14604.48	3563.88	1223.14	458.19	12499.63	13166.15	371.13	1814.5	316
29	325	1,2,3,4	15,20,20,15	12904.02	13956.46	349.45	2095.75	316.19	13500.35	14570.42	1774.11	2181.52	608.72	13270.76	14880.35	3667.03	3066.31	1028.1
30	375	1,4	22,33	18333.23	22860.7	7268.71	1136.94	649.3	18232.8	23454.13	7614.77	926.91	537.94	18012.02	21573.17	6065.11	1531.38	704.21
31	425	2,3,4	19,14,30	19087.89	25138.13	8161.44	1140.73	684.34	19186.99	25002.09	7997.4	1226.76	714.26	18937.51	27413.46	9105.43	1338.07	1094.02
32	475	1,2,3	31,28,25	23941.51	31371.45	8538.09	936.29	619.83	23475.21	29314.86	8156.53	1360.18	813.29	23606.26	33755.54	9514.49	1161.83	1011.53
33	512	1,2,4	25,27,29	25561.01	32531.53	9124.51	1212.78	747.87	26060.17	32643.86	9289.77	1349.01	760.26	25964.67	33117.74	10006.23	1717.59	1159.53
34	555	1,3,4	23,27,25	23989.57	35709.51	11705.88	1060.65	854.82	25156.16	30092.82	8620.81	1533.25	614.5	23919.54	27871.31	6845.42	2145.54	1005.01

[Computed on Intel® Core(TM), i5-10500 CPU @ 3.1 GHz, x64-based processor with installed RAM 16 GB (15.8 GB usable) having 64-bit Windows 10 Pro Operating System (Version:21H2 ; OS Build:19044), manufactured by HP®Inc.] (HP ProDesk 600 G6 MicroTower PC)

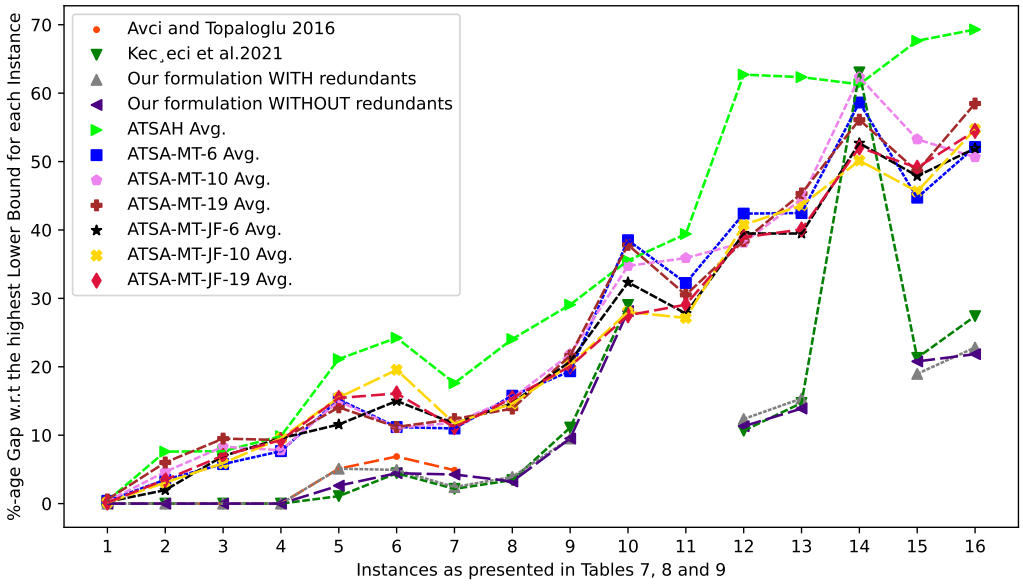


Figure 13: Comparing the Relative Gap % for the Instances with all Euclidean Distances

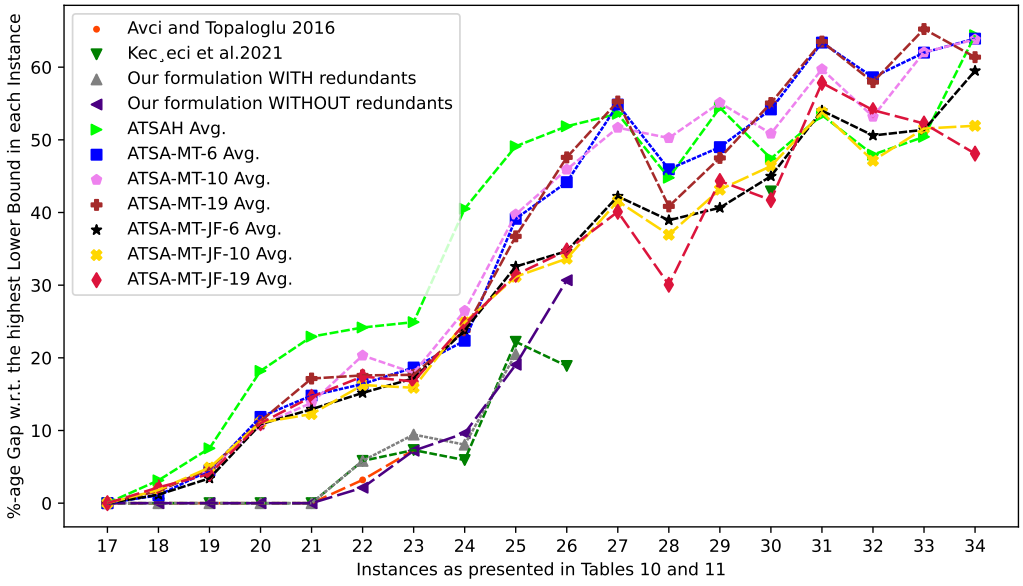


Figure 14: Comparing the Relative Gap % for the Instances with distances computed based on p-values in Table 1

5.2. Remarks

Trends from above computations were generated by finding the Relative Gaps in relation to the highest Lower Bound (LB) obtained in each case. The Relative Gap (also referred to as Fractional Gap or Optimality Gap or MIP Gap [11]) is calculated as in Eq. 41[12] where the Upper Bound (UB) is the (primal) objective value.

$$GAP\ \% = \left(\frac{UB - LB}{UB}\right) \times 100 \quad (41)$$

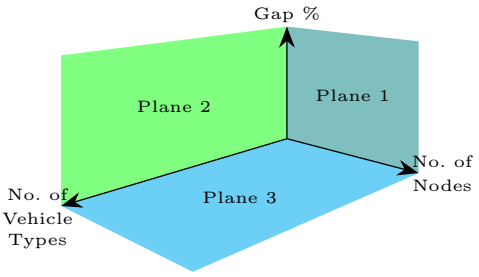


Figure 15: Axes for Contour generation

The trends in figures 13 or 14 form a small part of Plane 1 in Fig. 15. More computations may be carried by future researchers to develop entire contours comparing better formulations and heuristics for each case of the type of Node-to-Node distance calculation used (factors affecting these contours need to be further dwelled upon; like maximum computational time allowed and the specifications of the computer used). Comparison of the contours developed by different formulations would help us judge the best formulation for the considered problem of mHFVRPSDP.

6. Conclusion

The above results provide evidence of superiority of the ATS-MT algorithms above ATSAH. Further, the considered compact exact formulation is highly superior with respect to that developed by Avci and Topaloglu (2016) especially since the problem size does not increase with increase in number of vehicles (but the problem size varies with increase in the number of vehicle types). Further research in understanding more about how redundant constraints may affect the solution time and developing algorithms which may generate appropriate extra redundant constraints may be investigated

[11] https://www.gurobi.com/documentation/9.5/refman/mipgap2.html

[12] COIN-OR CBC calculates this Gap with denominator as LB (allowing it to exceed 100%)

regarding their effect on the optimal solution time. Future Heuristic development by self generating new redundant constraints to decrease solution times as per considered problems seems to be another wide open space to develop upon.

It is also suggested to find the optimal exact formulation which produces the fastest results for this specific problem of mHFVRPSDP so that exact formulation development may be closed and this problem would therefore remain open for heuristic developments and comparisons. Further improvements in this regards might be seen while using only single pickup and delivery decision variables between any two nodes (as in Keçeci et al. (2021)) instead of using it for each vehicle type layer. In this case, the optimal formulation might consist only of Equations 1 to 4 , 42 to 47, and 13; which should show further improvements in solution time as being more compact.

$$y_{0j} = 0, \quad \forall j \in N \tag{42}$$

$$z_{i0} = 0, \quad \forall i \in N \tag{43}$$

$$\sum_{\substack{j \in N_0 \\ (i \neq j)}} (y_{ij} - y_{ji}) = p_i, \quad \forall i \in N, \tag{44}$$

$$\sum_{\substack{j \in N_0 \\ (i \neq j)}} (z_{ji} - z_{ij}) = d_i, \quad \forall i \in N, \tag{45}$$

$$y_{ij} + z_{ij} \leq \sum_{k \in T} x_{ijk} Q_k, \quad \forall i, j \in N_0, \ (i \neq j), \tag{46}$$

$$y_{ij}, z_{ij} \geq 0, \quad \forall i, j \in N_0, \ (i \neq j), \tag{47}$$

This developed heuristic of JellyFishing search may be compared with roughly similar meta-heuristic of simulated annealing (SA); so that the temperature in case of SA algorithms may be varied like a jellyfish's bell instead

of having a steady decrease. Similar forms of JF technique should allow broader searches while in combination with deeper search techniques.

Further extensions of the problem towards a rich VRP would be to consider different load types for both delivery or pickup and including new types of functional depots, and transshipment ports allowing sequential load flow without accumulation. Allowing multiple trips, combined with the operator's choice of using a split or simultaneous delivery or pickup would further allow any node to be assigned any positive demand and pickup value this removing the present constraint of mandatorily having a node's pickup and delivery values below the largest vehicle type's capacity.

Data availability

The generated data-sets which were used to compare the results in Tables 7, 8, 9, 10 and 11 is made available at `https://github.com/SanTanBan/mHFVRPSDP-DataSet`.

Acknowledgements

This research was made possible through a sponsored project (Project Code: AEM) to Kalpana Chawla Space Technology Cell in Indian Institute of Technology Kharagpur by National Remote Sensing Centre under the Indian Space Research Organization.

Appendix

Table 12: Full Forms of the Acronyms used

Acronym	Full Form
ATSAH	ATS is the name of the developed Heuristic in (Avci and Topaloglu, 2016) without the Tabu Search and considering an unlimited number of vehicles of each type. While constraining the number of vehicles in each category we develop upon the ATS while referring to the logic provided in the Algorithm (Fig.1 in Avci and Topaloglu (2016)) instead of the Flowchart. Therefore we tag another A after the ATS referring to ATS-Algorithm. The last H corresponds to the term Heuristic.
ATSA-MT	ATS Algorithm Machine Tuning; this changes upon the predefined logics of edge-selection within the ATSAH allowing the algorithm to itself develop the selection logic based on the specific problem's data
ATSA-MT-JF	ATS Algorithm Machine Tuned Jelly Fishing; this heuristic search was developed with the inspiration of actual JellyFish movements, to combine broad as well as deep searches
mHFVRPSDP	multiple Heterogeneous Fleet Vehicle Routing Problem with Simultaneous Delivery and PickUp
OR	Operation Research
TSP	Travelling Salesman Problem
VRP	Vehicle Routing Problem

References

Achuthan, N.R., Caccetta, L., Hill, S.P., 2003. An improved branch-and-cut algorithm for the capacitated vehicle routing problem. Transportation Science 37, Pages 153–169. doi:10.1287/trsc.37.2.153.15243.

Agarwal, Y.K., Venkateshan, P., 2022. New valid inequalities for the symmetric vehicle routing problem with simultaneous pickup and deliveries. Networks 79, Pages 537–556. doi:10.1002/net.22069.

Ai, T.J., Kachitvichyanukul, V., 2009. A particle swarm optimization for the vehicle routing problem with simultaneous pickup and delivery. Computers & Operations Research 36, Pages 1693–1702. doi:10.1016/j.cor.2008.04.003.

Alabas-Uslu, C., Dengiz, B., 2011. A self-adaptive local search algorithm for the classical vehicle routing problem. Expert Systems with Applications 38, Pages 8990–8998. doi:10.1016/j.eswa.2011.01.116.

Asarin, E., Dang, T., Maler, O., Testylier, R., 2010. Using redundant constraints for refinement, in: Bouajjani, A., Chin, W.N. (Eds.), Automated

Technology for Verification and Analysis, Springer Berlin Heidelberg. pp. Pages 37–51.

Avci, M., Topaloglu, S., 2016. A hybrid metaheuristic algorithm for heterogeneous vehicle routing problem with simultaneous pickup and delivery. Expert Systems with Applications Volume 53, Pages 160–171. doi:10.1016/j.eswa.2016.01.038.

Barma, P.S., Dutta, J., Mukherjee, A., 2019. A 2-opt guided discrete antlion optimization algorithm for multi-depot vehicle routing problem. Decision Making: Applications in Management and Engineering 2, Pages 112–125. doi:10.31181/dmame1902089b.

Berbee, H.C.P., Boender, C.G.E., Rinnooy Ran, A.H.G., Scheffer, C.L., Smith, R.L., Telgen, J., 1987. Hit-and-run algorithms for the identification of nonredundant linear inequalities. Mathematical Programming 37, Pages 184–207. doi:10.1007/BF02591694.

Berbeglia, G., Cordeau, J.F., Gribkovskaia, I., Laporte, G., 2007. Static pickup and delivery problems: a classification scheme and survey. TOP 15, Pages 1–31. doi:10.1007/s11750-007-0009-0.

Bianchessi, N., Righini, G., 2007. Heuristic algorithms for the vehicle routing problem with simultaneous pick-up and delivery. Computers & Operations Research 34, Pages 578–594. doi:10.1016/j.cor.2005.03.014.

Bradley, G.H., Brown, G.G., Graves, G.W., 1983. Structural redundancy in large-scale optimization models, in: Redundancy in Mathematical Programming, Springer Berlin Heidelberg. pp. Pages 145–169.

Caron, R.J., Hlynka, M., McDonald, J.F., 1992. On the best case performance of hit and run methods for detecting necessary constraints. Mathematical Programming 54, Pages 233–249. doi:10.1007/BF01586052.

Caron, R.J., McDonald, J.F., 1989. A new approach to the analysis of random methods for detecting necessary linear inequality constraints. Mathematical Programming 43, Pages 97–102. doi:10.1007/BF01582281.

Caron, R.J., McDonald, J.F., Ponic, C.M., 1989. A degenerate extreme point strategy for the classification of linear constraints as redundant or

necessary. Journal of Optimization Theory and Applications 62, Pages 225–237. doi:10.1007/BF00941055.

Charles, V., Dutta, D., 2006. Identification of redundant objective functions in multi-objective stochastic fractional programming problems. Asia-Pacific Journal of Operational Research 23, Pages 155–170. doi:10.1142/S0217595906000863.

Chen, J.F., Wu, T.H., 2006. Vehicle routing problem with simultaneous deliveries and pickups. Journal of the Operational Research Society 57, Pages 579–587. doi:10.1057/palgrave.jors.2602028.

Christopher, Y., Wahyuningsih, S., Satyananda, D., 2021. Study of variable neighborhood descent and tabu search algorithm in VRPSDP. Journal of Physics: Conference Series 1872, 012002. doi:10.1088/1742-6596/1872/1/012002.

Dantzig, G.B., Ramser, J.H., 1959. The truck dispatching problem. Management Science 6, Pages 80–91. doi:10.1287/mnsc.6.1.80.

Dell'Amico, M., Righini, G., Salani, M., 2006. A branch-and-price approach to the vehicle routing problem with simultaneous distribution and collection. Transportation Science 40, Pages 235–247. doi:10.1287/trsc.1050.0118.

Dethloff, J., 2001. Vehicle routing and reverse logistics: The vehicle routing problem with simultaneous delivery and pick-up. OR-Spektrum 23, Pages 79–96. doi:10.1007/PL00013346.

Du, Y., Fu, S., Lu, C., Zhou, Q., Li, C., 2021. Simultaneous pickup and delivery traveling salesman problem considering the express lockers using attention route planning network. Computational Intelligence and Neuroscience , 5590758doi:10.1155/2021/5590758.

Eksioglu, B., Vural, A.V., Reisman, A., 2009. The vehicle routing problem: A taxonomic review. Computers & Industrial Engineering 57, Pages 1472–1483. doi:10.1016/j.cie.2009.05.009.

Elshaer, R., Awad, H., 2020. A taxonomic review of metaheuristic algorithms for solving the vehicle routing problem and its variants. Computers & Industrial Engineering 140, 106242. doi:10.1016/j.cie.2019.106242.

Fard, M.K., Akbari, M., 2013. A hybrid tabu search algorithm for the vehicle routing problem with simultaneous pickup and delivery and maximum tour time length, pp. Pages 801–810. doi:10.5897/AJBM2013.1607.

Gajpal, Y., Abad, P., 2009. An ant colony system (acs) for vehicle routing problem with simultaneous delivery and pickup. Computers & Operations Research 36, Pages 3215–3223. doi:10.1016/j.cor.2009.02.017.

Gajpal, Y., Abad, P., 2010. Saving-based algorithms for vehicle routing problem with simultaneous pickup and delivery. Journal of the Operational Research Society 61, Pages 1498–1509. doi:10.1057/jors.2009.83.

Gal, T., 1992. Weakly redundant constraints and their impact on postoptimal analyses in lp. European Journal of Operational Research 60, Pages 315–326. doi:10.1016/0377-2217(92)90083-L.

Gendreau, M., Laporte, G., Vigo, D., 1999. Heuristics for the traveling salesman problem with pickup and delivery. Computers & Operations Research 26, Pages 699–714. doi:10.1016/S0305-0548(98)00085-9.

Goksal, F.P., Karaoglan, I., Altiparmak, F., 2013. A hybrid discrete particle swarm optimization for vehicle routing problem with simultaneous pickup and delivery. Computers & Industrial Engineering 65, Pages 39–53. doi:10.1016/j.cie.2012.01.005.

Golden, B., Raghavan, S., Wasil, E. (Eds.), 2008. The Vehicle Routing Problem: Latest Advances and New Challenges. 1 ed., Springer New York, NY. doi:10.1007/978-0-387-77778-8.

Gutiérrez-Sánchez, A., Rocha-Medina, L.B., 2022. Vrp variants applicable to collecting donations and similar problems: A taxonomic review. Computers & Industrial Engineering 164, 107887. doi:10.1016/j.cie.2021.107887.

Hornstra, R.P., Silva, A., Roodbergen, K.J., Coelho, L.C., 2020. The vehicle routing problem with simultaneous pickup and delivery and handling costs. Computers & Operations Research 115, 104858. doi:10.1016/j.cor.2019.104858.

Johal, I., 2014. A new three phase method (sdp method) for the multi-objective vehicle routing problem with simultaneous delivery and pickup (vrpsdp).

Jr, E.F.S., Tseng, F.T., 2003. On 'redundant' constraints in stafford's milp model for the flowshop problem. Journal of the Operational Research Society 54, Pages 1102–1105. doi:10.1057/palgrave.jors.2601615.

Kalayci, C.B., Kaya, C., 2016. An ant colony system empowered variable neighborhood search algorithm for the vehicle routing problem with simultaneous pickup and delivery. Expert Systems with Applications 66, Pages 163–175. doi:10.1016/j.eswa.2016.09.017.

Keçeci, B., Altıparmak, F., İmdat Kara, 2021. A mathematical formulation and heuristic approach for the heterogeneous fixed fleet vehicle routing problem with simultaneous pickup and delivery. Journal of Industrial and Management Optimization 17, Pages 1069–1100. doi:10.3934/jimo.2020012.

Koch, H., Bortfeldt, A., Wäscher, G., 2018. A hybrid algorithm for the vehicle routing problem with backhauls, time windows and three-dimensional loading constraints. OR Spectrum 40, Pages 1029–1075. doi:10.1007/s00291-018-0506-6.

Çağrı Koç, Bektaş, T., Jabali, O., Laporte, G., 2016. Thirty years of heterogeneous vehicle routing. European Journal of Operational Research 249, Pages 1–21. doi:10.1016/j.ejor.2015.07.020.

Çağrı Koç, Laporte, G., İlknur Tükenmez, 2020. A review of vehicle routing with simultaneous pickup and delivery. Computers & Operations Research 122, 104987. doi:10.1016/j.cor.2020.104987.

Li, F., Golden, B., Wasil, E., 2007. A record-to-record travel algorithm for solving the heterogeneous fleet vehicle routing problem. Computers & Operations Research 34, Pages 2734–2742. doi:10.1016/j.cor.2005.10.015.

Liu, S., 2013. A hybrid population heuristic for the heterogeneous vehicle routing problems. Transportation Research Part E: Logistics and Transportation Review 54, Pages 67–78. doi:10.1016/j.tre.2013.03.010.

Liu, S., Huang, W., Ma, H., 2009. An effective genetic algorithm for the fleet size and mix vehicle routing problems. Transportation Research Part E: Logistics and Transportation Review 45, Pages 434–445. doi:10.1016/j.tre.2008.10.003.

Meliani, Y., Hani, Y., Elhaq, S.L., Mhamedi, A.E., 2019. A developed tabu search algorithm for heterogeneous fleet vehicle routing problem. IFAC-PapersOnLine 52, Pages 1051–1056. doi:10.1016/j.ifacol.2019.11.334.

Min, H., 1989. The multiple vehicle routing problem with simultaneous delivery and pick-up points. Transportation Research Part A: General 23, Pages 377–386. doi:10.1016/0191-2607(89)90085-X.

Mitra, S., 2005. An algorithm for the generalized vehicle routing problem with backhauling. Asia-Pacific Journal of Operational Research 22, Pages 153–169. doi:10.1142/S0217595905000522.

Montané, F.A.T., Galvão, R.D., 2002. Vehicle routing problems with simultaneous pick-up and delivery service. OPSEARCH 39, Pages 19–33. doi:10.1007/BF03398667.

Montané, F.A.T., Galvão, R.D., 2006. A tabu search algorithm for the vehicle routing problem with simultaneous pick-up and delivery service. Computers & Operations Research 33, Pages 595–619. doi:10.1016/j.cor.2004.07.009.

Mosheiov, G., 1994. The travelling salesman problem with pick-up and delivery. European Journal of Operational Research 79, Pages 299–310. doi:10.1016/0377-2217(94)90360-3.

Máximo, V.R., Cordeau, J.F., Nascimento, M.C., 2022. An adaptive iterated local search heuristic for the heterogeneous fleet vehicle routing problem. Computers & Operations Research 148, 105954. doi:10.1016/j.cor.2022.105954.

Nagy, G., Salhi, S., 2005. Heuristic algorithms for single and multiple depot vehicle routing problems with pickups and deliveries. European Journal of Operational Research 162, Pages 126–141. doi:10.1016/j.ejor.2002.11.003.

Ning, T., Wang, J., Han, Y., 2021. Logistics distribution de-carbonization pathways and effect in China: a systematic analysis using VRPSDP model. International Journal of Low-Carbon Technologies 16, Pages 1404–1411. doi:10.1093/ijlct/ctab063.

Parragh, S.N., Doerner, K.F., Hartl, R.F., 2008. A survey on pickup and delivery problems. Journal für Betriebswirtschaft 58, Pages 21–51. doi:10.1007/s11301-008-0033-7.

Pessoa, A., Sadykov, R., Uchoa, E., 2018. Enhanced branch-cut-and-price algorithm for heterogeneous fleet vehicle routing problems. European Journal of Operational Research 270, Pages 530–543. doi:10.1016/j.ejor.2018.04.009.

Prins, C., 2004. A simple and effective evolutionary algorithm for the vehicle routing problem. Computers & Operations Research 31, Pages 1985–2002. doi:10.1016/S0305-0548(03)00158-8.

Prins, C., 2009. Two memetic algorithms for heterogeneous fleet vehicle routing problems. Engineering Applications of Artificial Intelligence 22, Pages 916–928. doi:10.1016/j.engappai.2008.10.006.

Qin, W., Zhuang, Z., Huang, Z., Huang, H., 2021. A novel reinforcement learning-based hyper-heuristic for heterogeneous vehicle routing problem. Computers & Industrial Engineering 156, 107252. doi:10.1016/j.cie.2021.107252.

Rabin, M.O., 1976. Probabilistic algorithms. Algorithms and Complexity .

S., P., P., S., 2010. A comparative study of redundant constraints identification methods in linear programming problems. Mathematical Problems in Engineering , 723402doi:10.1155/2010/723402.

Salhi, S., Nagy, G., 1999. A cluster insertion heuristic for single and multiple depot vehicle routing problems with backhauling. Journal of the Operational Research Society 50, Pages 1034–1042. doi:10.1057/palgrave.jors.2600808.

Subramanian, A., dos Anjos Formiga Cabral, L., Carvalho, G.R., 2007. A hybrid metaheuristic for the vehicle routing problem with simultaneous pick-up and delivery, in: ICIEOM: XIII International Conference on Industrial

Engineering and Operations Management, Energy that moves Production: A Dialogue among Integration, Project and sustainability, pp. Pages 9–11.

Subramanian, A., Penna, P.H.V., Uchoa, E., Ochi, L.S., 2012. A hybrid algorithm for the heterogeneous fleet vehicle routing problem. European Journal of Operational Research 221, Pages 285–295. doi:10.1016/j.ejor.2012.03.016.

Subramanian, A., Uchoa, E., Pessoa, A.A., Ochi, L.S., 2013. Branch-cut-and-price for the vehicle routing problem with simultaneous pickup and delivery. Optimization Letters 7, Pages 1569–1581. doi:10.1007/s11590-012-0570-9.

Taillard, E.D., 1999. A heuristic column generation method for the heterogeneous fleet vrp. RAIRO - Operations Research 33, 1–14. doi:10.1051/ro:1999101.

Tan, S.Y., Yeh, W.C., 2021. The vehicle routing problem: State-of-the-art classification and review. Applied Sciences 11. doi:10.3390/app112110295.

Telgen, J., 1983. Identifying redundant constraints and implicit equalities in systems of linear constraints. Management Science 29, Pages 1209–1222. doi:10.1287/mnsc.29.10.1209.

Utama, D., Widjonarko, B., Widodo, D., 2022. A novel hybrid jellyfish algorithm for minimizing fuel consumption capacitated vehicle routing problem. Bulletin of Electrical Engineering and Informatics 11. doi:10.11591/eei.v11i3.3263.

Vural, A.V., 2003. A GA based meta-heuristic for capacited vehicle routing problem with simultaneous pick-up and deliveries. Master's thesis. Graduate School of Engineering and Natural Sciences. Sabanci University, Turkey.

Wojtyra, M., Frączek, J., 2012. Comparison of selected methods of handling redundant constraints in multibody systems simulations. Journal of Computational and Nonlinear Dynamics 8. doi:10.1115/1.4006958. 021007.

Zachariadis, E.E., Kiranoudis, C.T., 2011. A local search metaheuristic algorithm for the vehicle routing problem with simultaneous pick-ups

and deliveries. Expert Systems with Applications 38, Pages 2717–2726. doi:10.1016/j.eswa.2010.08.061.

Zhang, T., Tian, W.X., Zhang, Y.J., Zheng, X.C., 2007. Rlc_acs: An improved ant colony algorithm for vrpsdp, in: 2007 International Conference on Machine Learning and Cybernetics, pp. Pages 978–983. doi:10.1109/ICMLC.2007.4370284.

Zhu, F., Ning, Y., Chen, X., Zhao, Y., Gang, Y., 2021. On removing potential redundant constraints for svor learning. Applied Soft Computing 102, 106941. doi:10.1016/j.asoc.2020.106941.

Öztaş, T., Tuş, A., 2022. A hybrid metaheuristic algorithm based on iterated local search for vehicle routing problem with simultaneous pickup and delivery. Expert Systems with Applications 202, 117401. doi:10.1016/j.eswa.2022.117401.