Chapter 4: Multi-Depot Vehicle Routing Problem

4.1. A Multi-Depot Vehicle Routing Problem for Surplus Food Recovery and Redistribution

4.1.1. Introduction

Food banks engage in the procurement of surplus or residual food from prospective institutional or individual donors, subsequently deploying a distribution strategy that either involves direct delivery to beneficiaries (referred to as the front-end model) or facilitation through charitable agencies (Bazerghi et al., 2016; Rey et al., 2018). The significant operational challenges confronting a typical food bank encompasses the intricate processes of collecting, sorting, processing, distributing within constrained time range, and storing. The overall problem is further compounded by limited resources such as storage and logistics.

Recent study (Dubey & Tanksale, 2022) emphasizes that Indian food banks grapple with challenges stemming from scarcity of manpower, financial constraints, insufficient storage and transportation facilities, deficiencies in collection and distribution infrastructure. The study delineated that the absence of tactical and operational-level planning significantly impedes the seamless functioning of food banks. Consequently, it becomes imperative for food banks to maximize the number of beneficiaries served, while minimizing the overall cost incurred throughout the collection and redistribution of donated food. This optimization necessitates effective utilization of scarce resources and the implementation of efficient operational planning, particularly in the formulation of transportation schedules, to enhance the overall effectiveness of food banks in serving communities.

The day-to-day operational dynamics of food banks, encompassing the collection and distribution of surplus and leftover food, can be conceptualized as a variant of the Vehicle

Routing Problem (Nair et al., 2018; Rey et al., 2018). The VRP is a combinatorial optimization problem wherein vehicles start their routes from depots, traverse designated paths to fulfil the requirements of a predefined set of customers, and ultimately return back to the depot (Corona-Gutiérrez et al., 2022; Koç & Laporte, 2018). In the specific context of food bank operations, the analogous structure involves a defined set of recipients or beneficiaries (e.g., orphanages, old-age homes, slum areas, charitable agencies, etc.) to which food items are intended for delivery and it is facilitated through a fleet of vehicles. However, the availability of food is not centralized at the depots instead, it typically necessitates collection from a set of diverse locations such as restaurants, catering service providers, supermarkets and hotels. In the given context of food recovery and redistribution the depots are the locations from where volunteers start their journey and since there could be several such hubs, we have considered multi-depot (MD) variant of VRP. Food banks operating on the front-end model commonly employ the same fleet of vehicles for both the recovery and subsequent redistribution of surplus or donated food. Mostly, these vehicles are owned by the volunteers but we have seen few cases where food banks rents vehicles for performing their activities. This context of problem aligns with the pickup and delivery variant of VRP. The perishable nature of donated food presents a pervasive concern for food banks, particularly in the case of ready-to-eat food (cooked meal), where distribution to beneficiaries must occur on the same day or within a few hours. Similarly, donors adhere to strict time constraints for the retrieval of donated food due to their perishable nature. This situation highlights the need for incorporating time windows (TW) into the logistics of collection and then distribution food items. Notably, the constrained availability of vehicles and their capacities necessitates a specific approach where donations or demands can be fulfilled in multi-trips to the same node, requiring the splitting of pickup or delivery

demands. Therefore, split pick-up and split-delivery variant deems fit for the problem context. Moreover, splitting the pickup and delivery demands, necessitates multiple visits to pick-up or delivery locations in the routes. Therefore, following these intricacies the food coolection and distribution problem of food banks can be modeled as Multi-Depot Vehicle Routing Problem with Time – Windows, Split Pickup and Split Deliveries (MDVRP-TW-SP-SD).

Existing literature has made noteworthy contributions by considering these distinctive features of the VRP to address the operational intricacies faced by food banks. The details of these work are presented in Table 2.2 of the literature review section.

The motivation for the current study arises from the operational problem involved in the recovery and redistribution of donated food faced by food banks operating within the Indian landscape. As elucidated earlier, we conceptualize the fundamental problem as a pickup delivery variant of the VRP with time windows consideration. However, in consideration of several distinctive attributes inherent to this context, we propose an augmented variant, specifically, a multi-depot VRP with split-pickup and split-delivery, with time-windows considerations. The multi-depot VRP variant appears to fit more appropriately in countries where food banks lack institutionalization. For example, numerous small and large-scale food banks operate independently in India, devoid of a central governing body, thus highlighting the dearth of coordination. Remarkably, the notion of splitting on both the supply and demand sides in the context of food banks, a novel aspect of the problem, has been under-explored in the existing literature on VRP problems.

In recent efforts, the Indian Food Sharing Alliance (Indian Food Sharing Alliance, n.d.) has undertaken initiatives to address this issue by establishing a unified platform for various food banks, fostering collaboration and coordination among them. This approach envisions each

location equipped with storage and transportation capacities within the network of food banks as a depot, enabling the formulation of efficient routes. Such a strategy facilitates the sharing of resources and information, collectively working towards a common objective.

The major contributions of this chapter are as follows.

- We introduce a comprehensive variant of the Vehicle Routing Problem (VRP) tailored specifically for addressing the operational challenges inherent in surplus food recovery and redistribution within Indian food banks operating on the front-end model. We illustrate the application of this proposed model through a detailed case study of the Robin Hood Army, a prominent not-for-profit organization dedicated to food recovery and redistribution in India.
- Our work extends the existing body of literature by proposing a novel variant of the VRP, termed the multi-depot (MD) VRP with time windows, split-pickup, and split delivery (MDVRP-TW-SP-SD), featuring a heterogeneous fleet of vehicles. To the best of our knowledge, this particular variant of the MDVRP has not been previously investigated in the scholarly literature.
- We present a robust solution methodology based on genetic algorithms and hybridized with local search techniques, designed to efficiently address the complexities of the MDVRP-TW-SP-SD problem. Our approach demonstrates notable efficacy in solving instances with up to 240 nodes in the network.
- Additionally, we expand the available dataset for the MDVRP by developing 180 new instances, leveraging the 20 benchmark instances proposed by Cordeau et al. (2001). This contribution enhances the resources available for future research endeavors and

facilitates the validation and comparison of emerging solution methodologies within the domain of the proposed variant of multi-depot vehicle routing.

4.1.2. Model Formulation

4.1.2.1. Problem Description

The MDVRP-TW-SP-SD is defined on a multi-graph $G = (N, S, A)$, where N is the vertex set of customers containing both pickup as well as delivery requests, S is the vertex set for the depots and A is the arc set. The vertex set N, consists of set of pickup nodes, $P = \{1,2, \dots . , p\}$ and set of delivery nodes, $D = \{p + 1, p + 2, \dots . , d\}$ that is $N = P \cup D$. Each node $i \in N$ has a discrete demand request θ_i, service time τ_i, and an associated time window $[a_i, b_i]$ within which the node i has to be served. The demand of pickup nodes is taken as $+\theta_i$ and demand of delivery nodes is taken as $-\theta_i$. In the proposed case study, the parameter θ_i is the demand of node $i \in N$ i.e., quantity of food needs to be picked up ($\theta_i > 0$) or delivered ($\theta_i < 0$). The unit of θ_i is number of servings i.e., a pickup demand of $\theta_i = 10$ depicts it can serve 10 beneficiaries. A penalty cost ϕ is incurred for each unit of unmet demand of pickup or delivery node. The set $V = N \cup S$ is the set of all nodes in the network where $S = \{d + 1, d + 2, \dots , n\}$ is the set of all depots. The total number of nodes in the network is $|V| = n$. The arc set $A = \{(i, j): i, j \in V\}$ consists of all the arcs in the network and each arc (i, j) has a non-negative travel time T_{ij} and cost C_{ij}. The arc lengths C_{ij} follow triangular inequality and are assumed to be symmetrical. Each depot, $s \in S$ has a heterogeneous fleet of vehicles, $K = \{1,2, \dots , m\}$ with capacity C_{ks} and $|K| = m$. A fixed cost f_{ks} will be incurred if the vehicle is selected for pickup/delivery operation. Each vehicle $k \in K$ starts its route from a depot and needs to return to the same depot. Multiple vehicles of the same or different depot can visit the same customer

i.e., splitting of pickup /demand request is allowed. The MDVRP-TW-SP-SD is described in Figure 4.1.

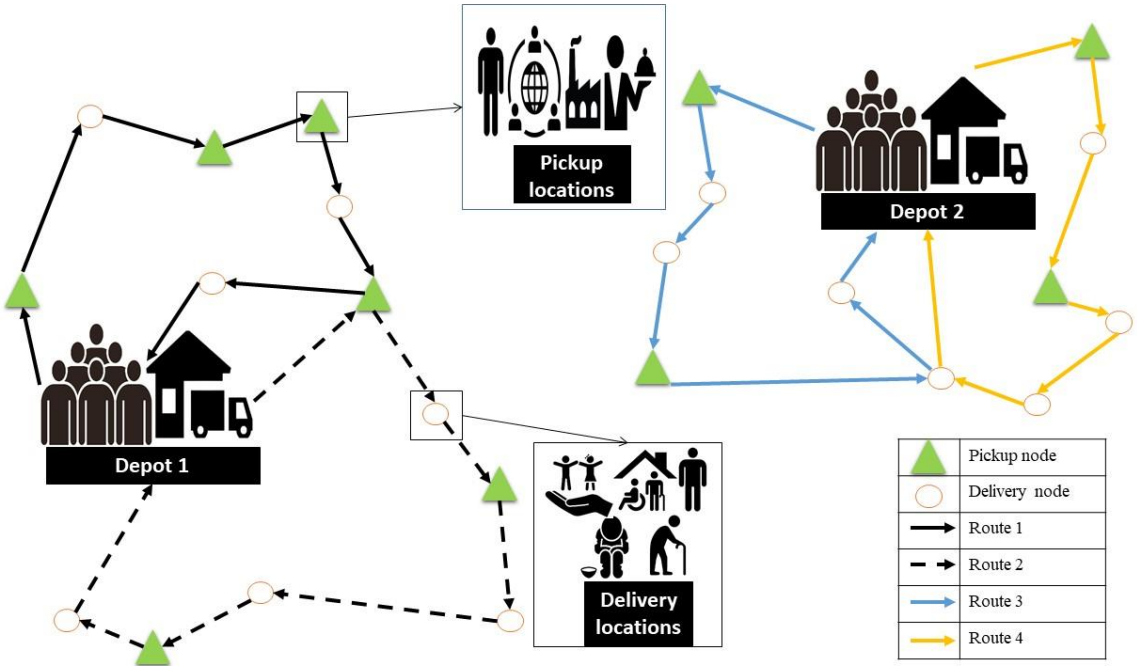


Figure 4.1:. Problem description of MDVRP-TW-SP-SD

Given the network as described above, the problem is to determine the optimal number of vehicles from each depot to be hired and their optimal routes (sequence for visiting pickup and demand nodes) such that the total cost of hiring the vehicles, routing cost and penalty for unmet demand are minimized. The problem is formulated as a mixed-integer programming problem using the notation given in Table 4.1.

4.1.2.2. Notation

Table 4.1: Description of sets, indices and parameters used in the model formulation

Sets and indices		
P	Set of nodes having pickup requests	$P = \{1,2, \ldots . , p\}$
D	Set of nodes having delivery needs	$D = \{p + 1, p + 2, \ldots . , d\}$

$\boldsymbol{N}$	Set of pickup and delivery nodes	$N = P \cup D = \{1,2, \dots . , d\}$
$\boldsymbol{S}$	Set of depots	$S = \{d + 1, d + 2, \dots , n\}$
$\boldsymbol{V}$	Set of all nodes in the network	$V = N \cup S = \{1,2, \dots . , n\}, \lvert V\rvert = n$
$\boldsymbol{K}$	Set of vehicles available at each depot	$K = \{1,2, \dots , m\}, \lvert K\rvert = m$
$\boldsymbol{A}$	Set of all arcs in the network	$A = \{(i,j): i,j \in V\}$

Parameters

$\boldsymbol{C_{ijks}}$	Cost of traversing arc (i,j), $\forall i,j \in A$ using vehicle $k \in K$ from depot $s \in S$
$\boldsymbol{C_{ks}}$	Capacity of vehicle k at depot s, $\forall k \in K$
$\boldsymbol{f_{ks}}$	Fixed cost for vehicle used $\forall k \in K$
ϕ	Penalty cost for not meeting the request of node $i \in N$
$\boldsymbol{T_{ij}}$	Time taken to traverse arc $(i,j) \in A$
$\boldsymbol{\tau_i}$	Service time of node $i \in N$
$[\boldsymbol{a_i}, \boldsymbol{b_i}]$	Time window of node $i \in V$
$\boldsymbol{\theta_i}$	The quantity of food (number of servings) requested at node $i \in N$ $\theta_i > 0, \forall i \in P$, $\theta_i < 0, \forall i \in D$

4.1.2.3. Decision variables

$x_{ijks} = \begin{cases} 1, & \text{if vehicle } k \text{ of depot } s \in S \text{ traverses path node } i \text{ to node } j \\ 0, & \text{otherwise} \end{cases}$

q_{iks}= Load on vehicle k of depot s after servicing node i, $\forall i \in V$

u_{iks}= Quantity picked up/delivered by vehicle k of depot s at node $i, \forall i \in V$

$w_{ks} = \begin{cases} 1, & \text{if vehicle } k \text{ of depot } s \in S \text{ is hired} \\ 0, & \text{otherwise} \end{cases}$

ρ_{ijks}= Quantity carried on arc $(i,j) \in A$ by vehicle $k \in K$ of depot $s \in S$

t_{iks} = Service start time for vehicle $k \in K$ of depot s, starts servicing node $i, \forall i \in N$

π_i=The unmet request of pickup node $i, \forall\, i \in P$

σ_j=The unmet request of delivery node $j, \forall j \in D$

4.1.2.4. Formulation

Minimise:

$$\sum_{s\in S}\sum_{k\in K}\sum_{(i,j)\in A} C_{ij}x_{ijks} + \sum_{s\in S}\sum_{k\in K} f_{ks} * w_{ks} + \phi * (\sum_{i\in P}\pi_i + \sum_{j\in D}\sigma_j) \quad (1)$$

Subject to;

$$\sum_{k\in K}\sum_{j\in N} x_{ijks} \leq |K|, \qquad \forall i = s \in S \quad (2)$$

$$x_{ijks} = 0, \qquad \forall i \in S, \forall s \in S, i = s, \forall j \in D, \forall k \in K \quad (3)$$

$$x_{ijks} = 0, \qquad \forall i \in S, \forall s \in S, i = s, \forall j \in S, \forall k \in K \quad (4)$$

$$x_{ijks} = 0, \qquad \forall i \in S, \forall s \in S, i \neq s, \forall j \in V, \forall k \in K \quad (5)$$

$$\sum_{j\in V} x_{ijks} \leq w_{ks}, \qquad \forall i \in S, \forall s \in S, i = s, \forall k \in K \quad (6)$$

$$\sum_{i\in V} x_{ijks} - \sum_{i\in V} x_{jiks} = 0, \qquad \forall k \in k, \forall j \in V, \forall s \in S \quad (7)$$

$$q_{iks} \leq C_{ks} * w_{ks}, \qquad \forall i \in V, \forall k \in K, \forall s \in S \quad (8)$$

$$\sum_{k\in K}\sum_{s\in S} u_{iks} + \pi_i = \theta_i, \qquad \forall i \in P \quad (9)$$

$$\sum_{k\in K}\sum_{s\in S} u_{iks} + \sigma_j = -\theta_j, \qquad \forall j \in D \quad (10)$$

$$u_{iks} \leq \theta_i * \sum_{j\in V} x_{jiks}, \qquad \forall i \in P, \forall k \in K, \forall s \in S \quad (11)$$

$$u_{iks} \leq -\theta_i * \sum_{j\in V} x_{jiks}, \qquad \forall i \in D, \forall k \in K, \forall s \in S \quad (12)$$

$$\sum_{i\in D} u_{iks} \leq \sum_{i\in P} u_{iks}, \qquad \forall k \in K, \forall s \in S \quad (13)$$

$$q_{iks} - u_{jks} - q_{jks} \leq (1 - x_{ijks}) * M_{ijk}, \qquad \forall i \in V, \forall j \in D\ , \forall\, k \in K, \forall s \in S \quad (14)$$

$$q_{iks} + u_{jks} - q_{jks} \leq (1 - x_{ijks}) * M_{ijk}, \qquad \forall i \in V, \forall j \in P\ , \forall\, k \in K, \forall s \in S \quad (15)$$

$$q_{iks} = 0, \qquad \forall i \in S, \forall s \in S, i = s, \forall k \in K \qquad (16)$$

$$q_{iks} - \rho_{ijks} \leq \left(1 - x_{ijks}\right) * M_{ijk}, \qquad \forall i \in V, \forall j \in V, \forall k \in K, \forall s \in S \qquad (17)$$

$$\rho_{ijks} = 0, \qquad \forall i \in S, \forall j \in V, \forall k \in V, \forall s \in S \qquad (18)$$

$$\rho_{ijks} = 0, \qquad \forall i \in V, \forall j \in S, \forall k \in V, \forall s \in S \qquad (19)$$

$$u_{jks} \leq \sum_{i \in V} \rho_{ijks}, \qquad \forall j \in D, \forall k \in K, \forall s \in S \qquad (20)$$

$$t_{iks} + \tau_i + T_{ij} - t_{jks} \leq \left(1 - x_{ijks}\right) * M_{ijk}, \qquad \forall i \in V, \forall j \in N, \forall\, k \in K, \forall s \in S \qquad (21)$$

$$a_i \leq\ t_{iks} \leq\ b_i, \qquad \forall i \in V, \forall k \in K, \forall s \in S \qquad (22)$$

$$t_{iks} = 0, \qquad \forall i \in S, \forall k \in K, \forall s \in S \qquad (23)$$

$$x_{ijkS} \in \{0,1\}, \qquad \forall i \in V, \forall j \in V, \forall k \in K, \forall s \in S \qquad (24)$$

$$w_k \in \{0,1\}, \qquad \forall k \in K \qquad (25)$$

$$t_{iks} \geq 0, \qquad \forall i \in V, \forall k \in K, \forall s \in S \qquad (26)$$

$$q_{iks} \geq 0, \qquad \forall i \in V, \forall k \in K, \forall s \in S \qquad (27)$$

$$u_{iks} \geq 0, \qquad \forall i \in V, \forall k \in K, \forall s \in S \qquad (28)$$

$$\rho_{ijks} \geq 0, \qquad \forall i \in V, \forall j \in V, \forall k \in K, \forall s \in S \qquad (29)$$

$$\pi_i \geq 0, \qquad \forall i \in P \qquad (30)$$

$$\sigma_j \geq 0, \qquad \forall j \in D \qquad (31)$$

The objective function (1) minimizes the cost that comprises of fixed cost of hiring the vehicles and routing costs and it seeks to maximize the total demand fulfilled by associating penalties on each unit of unmet pickup as well as delivery requests. Constraint (2) restricts the total number of routes that originate from each depot with respect to the availability of vehicles at the depot. Constraints (3) - (5) restrict the infeasible flows in the network. Constraint (3) imposes restriction on visiting delivery nodes directly from the depot since the load availability at the depot is assumed to be zero. It can be noted that this constraint can be removed in certain

applications where depot has initial stock and can directly meet the delivery demand. Constraint (4) prohibits visits between the depot nodes. Constraint (5) establishes an association of each depot with the vehicles available at that depot. In this constraint, we connect vehicle set *k* with depo set s i.e., if path starts from a depot, then the value of x_{ijks} for all the combination of (i, s): $i \neq s$ for any vehicle *k* and node *j* will be zero. Constraint (6) ensures that a vehicle can be used for pickup/delivery only if it is hired. Constraint (7) expresses the flow conservation i.e. the vehicle that visits the node and the vehicle that departs the node should be same. Constraint (8) ensures the load on the vehicle should not exceed its capacity. Constraints (9) and (10) ensure that the sum of total fulfilled demand and unmet demand should be equal to the actual demand of a pickup and delivery node, respectively. Constraints (11) and (12) ensure that the fulfilled demand of all the pickup or delivery nodes visited by a vehicle of a depot should be less than the actual demand of that node. Constraint (13) ensures that the total delivery amount by a vehicle should not exceed the total pickup quantity by the same vehicle. Constraints (14) and (15) ensure that if vehicle k traverses the path (i, j), load on vehicle after servicing node j is the difference between load of vehicle after leaving node i and fulfilled demand of node j. Constraint (16) ensures that load on vehicle k after servicing depot remains zero. Constraint (17) is similar to Constraints (14) and (15) except accounting for load carried on arc (i, j) by vehicle k of depot s. Constraints (18) and (19) ensure load on arc (i, j) by vehicle k of depot s is zero if vehicle traverses from depot s to any node and vice-versa. Constraint (20) ensures that quantity delivered to any delivery node should not exceed sum of load on arcs from all the nodes to that delivery node. Constraints (21) -(23) impose the time window requirements and precedence order. Finally, Constraints (24) - (31) depict the nature and domain of the decision variables.

4.1.3. Solution Approach

In this section, the overall solution strategy adopted to solve the MDVRP-TW-SP-SD is discussed in detail. The classical VRP itself is an NP-hard problem and its variants such as multi-depot VRP are proven to be computationally intractable (Escobar et al., 2014; Zarandi et al., 2011). In addition, several state-of-the-art solvers like GUROBI fail to solve the VRP and its variants due to substantial computational complexity associated with these problems. Several heuristics and meta-heuristics such as variable neighborhood search (Hesam Sadati et al., 2021), iterative local search (Máximo & Nascimento, 2021), adaptive large neighborhood search (Gu et al., 2019), particle swarm optimization (W. Chen et al., 2021), ant-colony optimization (Chandra Mohan & Baskaran, 2012), tabu-search (Escobar et al., 2014) and genetic algorithm (Anbuudayasankar et al., 2012; Filipec et al., 1997, 2000; Fung et al., 2013; Lei et al., 2020) have been proposed in the literature to solve VRP and its variants (Gasque & Munari, 2022). Due to computational simplicity, flexibility, powerful, scalable and robust nature, the Genetic Algorithm (GA) and its variation is chosen to solve the MDVRP-TW-SP-SD presented in this work. Hybridization of Genetic algorithms attenuated research experts for solving VRP problems (Vidal et al., 2013). Ho et al. (2008) hybridized GA by using the nearest neighbor method. Liu et al. (2009) used edge exchange schemes instead of mutation. Liu et al. (2014) replaced the mutation process with local search techniques. Zhen et al. (2020) hybridized the GA with variable neighborhood descent to accelerate the algorithm. In this paper, we present an Elitist GA and an Elitist GA hybridized with local search to efficiently solve the proposed MDVRP-TW-SP-SD. In addition, we have also used GUROBI solver to solve the problem and for comparison of results. The description of the proposed algorithms is presented below.

4.1.3.1. Elitist Genetic Algorithm

Genetic algorithm is a population-based metaheuristic that follows the theory of natural evolution and the principle of survival of fittest (Deb et al., 2002). GA has been profoundly used to solve a wide range of optimization problems. The GA is iteratively executed following three major operators – selection, crossover and mutation. The algorithm starts with initializing the population with candidate solutions encoded in the form of chromosomes. The fitness function is used to evaluate the fitness score of each chromosome in the population. The fittest individuals are chosen for reproduction to generate offspring. In each generation, these chromosomes undergo the process of crossover to recombine and mutation to generate offspring chromosomes for the next generation. The crossover of chromosomes is analogous to the reproduction process, performed to combine the genetics of two parents for convergence. To diversify the solution space and inhibit premature convergence, mutation is performed on chromosomes. The process of selection, crossover and mutation is performed till the stopping criterion is not met. The fundamentals of the elitist genetic algorithm used to solve the model is described in Algorithm 1.

Algorithm 1: Elitist Genetic Algorithm

Input: Instance data, population size as p, crossover probability as μ, mutation probability as ρ, and generation size as g.

Output: Solution to MDVRP-TW-SP-SD problem

1 Read instance data

2 Initialize the population P with random chromosomes

4 $gen\_size \leftarrow 0$

While $gen\_size < g$ **do**

Convert the chromosomes into feasible route format

Evaluate the fitness function f, of each chromosome in the population

Select $P_1 = \alpha * p$ best chromosomes from P using *selBest*

$P_2 = (1 - \alpha) * p$ chromosomes is mating pool

repeat

Select two parents P_2^i and P_2^{i+1} from population P_2 using roulette wheel for mating

Select rand $\in (0,1)$

If rand < μ :

Apply one-point crossover $C_1, C_2 \leftarrow crossover(P_2^i, P_2^j, \mu)$

$P_2^i, P_2^{i+1} \leftarrow C_1, C_2$

Select rand $\in (0,1)$

If rand < ρ :

Apply inverse mutation $P_2^i \leftarrow InverseMutation(P_2^i, \rho)$

until all chromosomes in P_2 mates

Generate new population $P \leftarrow P_1 \cup P_2$

gen_size++

end while

return the best solution in P

4.1.3.1.1.Encoding, decoding of chromosomes and initialization of population

The chromosomes or individuals in the population are encoded as a sequence of genes i.e., sequence of customers/nodes and its length is the total number of customer nodes in the network. Each gene represents either the pickup node or the delivery node. Each gene is further associated with a list having values as demand of the node, fulfilled demand of the node, remaining demand of the node, vehicle number and depot number. In this study, we do not use the route delimiter i.e., the sequence of genes does not contain the starting and ending point (depot number) of a route. Instead of that the depot numbers are generated randomly. For example, a problem instance with 8 customers and 2 depots, 5 vehicles at each depot and vehicle capacity of 10 units of load can be encoded initially as shown in Figure 4.2.

The above encoding scheme provides the flexibility to incorporate a heterogeneous fleet of vehicles as each vehicle can have different capacities. At any point in time, the status of fulfilled demand and unmet demand of any node can be easily exploited using this representation, and therefore, the concept of splitting the pickup as well as delivery demand can be smoothly done using this encoding technique. The decoding of the chromosome into solution (routes) is relatively easy. Each depot is one by one chosen and then sequenced in route format with respect to vehicles originating from the depot, as shown in the second section of Figure 4.2.

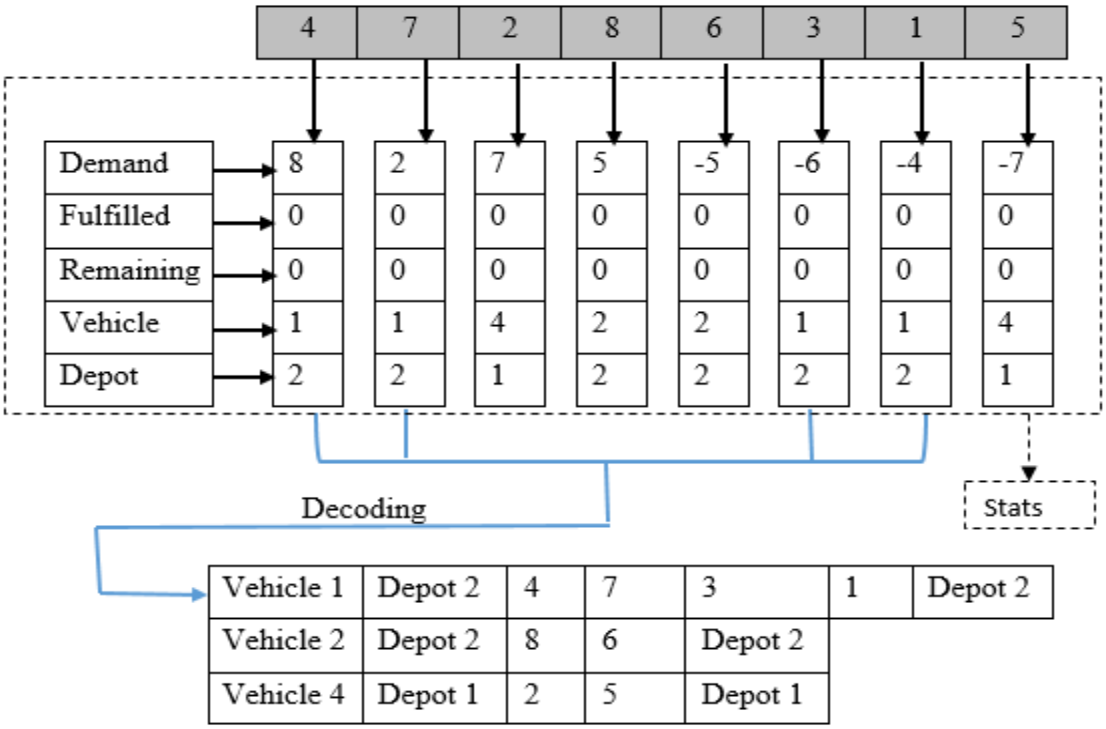


Figure 4.2: Encoding and Decoding of chromosomes

Population in GA can be initialized randomly or using some heuristics or a combination of both. In this study, we have initialized the population with random chromosomes to exploit the large and diverse search space.

4.1.3.1.2.Chromosomes to feasible route conversion

The most critical part of the algorithm is to evaluate the chromosomes for feasibility. As we have initialized the population with random chromosomes, it might be possible that some of these chromosomes do not follow some constraints of the MDVRP-TW-SP-SD. For example, we have assumed that the depot does not carry any stock of the item to be collected or distributed. Therefore, the vehicle that originates from a depot must visit a pickup node first. However, the randomly generated chromosomes might consist of a path directly from a depot to a delivery node. Similarly, it might be possible that the current load on the vehicle is zero and the next node in the chromosome is a delivery node. To deal with such infeasibilities, we

introduce a repair operator to convert the infeasible chromosomes into feasible ones. For example, for the above-mentioned cases, we have used *swap* operator to swap the position of that delivery node with a pickup node to make the chromosome feasible. The decision for splitting the supply and demand is also carried out in this phase of the program. A detailed description of the repair operator is presented following in Subroutine 1.

SUBROUTINE 1: CHROMOSOME TO FEASIBLE ROUTES

Input: Instance, randomly generated chromosome

Output: feasible routes

Initialize the variables and sets: $vehicle\_capacity_k$, $vehicle\_count$, $service\_time$, $elapsed\_time$, $updated\_vehicle\_load$, $remaining\_vehicle\_capacity$, $visited$, $unvisited$, $to\_be\_visited$, $total\_pickup\_demand$, $total\_delivery\_demand$, $pickup\ nodes$, $delivery\ nodes$, $stats$.

$individual \leftarrow chromosome$

$route \leftarrow [\,]$

$remaining\_vehicle\_capacity \leftarrow vehicle\_capacity_k$

$updated\_vehicle\_load \leftarrow 0$

if $stats[0][demand] < 0$:

    $\boldsymbol{swap}(\ individual[0], pickup\ nodes)$

for $customer\_id$ in $individual$:

    if $remaining\_vehicle\_capacity < stats[customer\_id][remaining]$: //case 1

        Split the pickup demand of node $customer\_id$ and update $stats$ variable

        Append $customer\_id$ to $to\_be\_visited$ list

        $remaining\_vehicle\_capacity \leftarrow 0$

    elif $updated\_vehicle\_load < |stats[customer\_id][remaining]|$: //case 2

        Split the delivery demand and update $stats$ variable

        Append $customer\_id$ to $to\_be\_visited$ list

        $updated\_vehicle\_load \leftarrow 0$

    elif $total_pickup_demand > total_$delivery$_demand$: //case 3

Call $excess\_pickup(individual, stats, instance)$

elif $total\_delivery\_demand > total\_pickup\_demand$: //case 4

Call $excess\_delivery(individual, stats, instance)$

elif $remaining\_vehicle\_capacity ==$ 0 && $stats[customer\_id][remaining] < 0$: //case 5

Update $stats$ variable

$remaining\_vehicle\_capacity -= stats[customer\_id][remaining]$

$updated\_vehicle\_load += stats[customer\_id][remaining]$

elif $remaining\_vehicle\_capacity ==$ 0 && $stats[customer\_id][remaining] > 0$: //case 6

Swap $customer\_id$ with any unvisited delivery node

Check for all possible cases and update $stats$ variable

Update $updated\_vehicle\_load$ and $remaining\_vehicle\_capacity$ variable accordingly

elif $updated\_vehicle\_load == 0$ && $remaining\_vehicle\_capacity >$ 0 && $stats[customer\_id][remaining] < 0$: //case 7

Check for all possible cases and update $stats$ variable

Update $updated\_vehicle\_load$ and $remaining\_vehicle\_capacity$ variable accordingly

else:

Update $stats, remaining\_vehicle\_capacity$ and $updated\_vehicle\_load$ variable

$updated\_elapsed\_time = elapsed\_time + travel\_time + service\ \ time$

if $updated\_vehicle\_load \geq 0$ && $updated\_elapsed\_time <$ $depart\_due\_time$:

Append $customer\_id$ to list $route$

Update $elapsed\_time$

else:

If $vehicle\_count \leq max\_vehicle$:

Append route to all_route list

41 $route \leftarrow [\,]$

42 Randomly select a vehicle k from the available vehicle

43 $remaining\ vehicle\ capacity \leftarrow vehicle\ capacity_k$

44 Update flow variables, time window variables

45 If $stats[customer\_id][demand] > 0$:

46 Append customer_id to $route$ list

47 Update flow variables, time window variables

48 Check for possibles cases and update $stats$ variable

49 Append $customer\_id$ to $visited$ list

50 Update remaining $pickup\ nodes$ set and $delivery\ nodes$ set.

51 **if** $route! = [\,]$:

52 Append $route$ to all_route list.

53 **Call** *feasible_remaining_options_multivisit*( )
$(route, to\_be\_visited, stats, updated\_vehicle\_load, instance)$

54 **return** the chromosome into feasible routes format

4.1.3.1.3.Fitness function

The fitness function in GA is used to assign scores/weight to the chromosomes to know the quality of candidate solutions/chromosomes. The fitness function for the chromosomes is defined as the sum of total vehicle hiring cost, routing cost and penalty cost for each unit of unmet demand as defined in the objective function (1) of the model. The candidate solution with a smaller fitness value is considered a better fit since the nature of the fitness function used in this study is of minimization type. The input to the fitness function $\boldsymbol{f}$ is a feasible chromosome generated in the initialization phase or a repaired chromosome obtained from the Subroutine ***1.***

4.1.3.1.4.Selection

Selection in GA is done to reproduce population for next-generation by selecting two parents for mating. The rate of convergence depends upon the selection pressure (Katoch et al., 2021).

In this study, we have used the roulette wheel technique to select parents from sub-population to reproduce new offspring for the next generation, as it outperformed the other selection techniques in our pilot experimentation. In roulette wheel selection, a portion of the wheel is allocated to each chromosome in the population which is inversely proportional to its fitness value. The wheel is then rotated to select two chromosomes that will participate in the formation of offspring using crossover and mutation.

4.1.3.1.5.Crossover and Mutation

Crossover is used to combine the information that lies in two parents in a certain way to create new and hopefully better chromosomes. Crossover is also termed as recombination where two parents are selected from the mating pool and their genes are reordered to obtain offspring which contain the characteristics of both parents. We have used single-point crossover as shown in Figure 4.3. The Repair mechanism as explained in Subroutine 1 is utilized to restore feasibility in the offspring if it arrives after crossover operation.

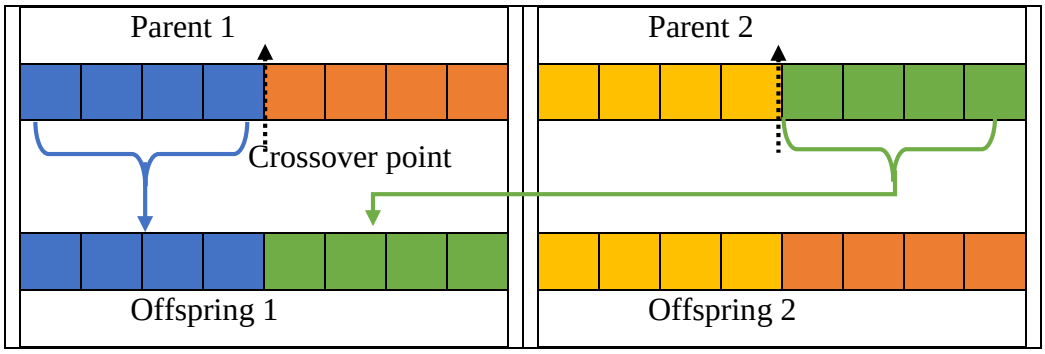


Figure 4.3: Single point crossover for generating offspring

4.1.3.1.6.New population by elitist strategy

Elitism in GA affects the search potential (Ishibuchi et al., 2008) and increases the convergence speed of evolutionary algorithms (Ahn & Ramakrishna, 2003). Elitism ensures most fit individuals/chromosomes (also known as elites) of current generation copy their characteristics

to the next generation. The population is divided into two sections. The first subpopulation consists of $(\alpha * P)$ elites (having minimum fitness scores), and second sub-population is the mating population of size $(1 - \alpha) * P$. The first sub-population comprising of elites is passed directly from the previous generation to the next generation. The second subpopulation undergoes crossover and mutation operation as explained in Steps 4.1.4 and 4.1.5 to generate $(1 - \alpha) * P$ offspring chromosomes. The set of these two subpopulations creates a new population for the next generation.

The above steps of GA are iteratively performed as given in Algorithm 1 till the convergence. Convergence depicts the stopping criteria of the algorithm and we have used number of generations for it.

4.1.3.2. 3-Opt Local Search Heuristic

Local search is the single point search techniques that systematically improve an existing solution by exploring the neighborhood structures through perturbations. The local search algorithms can be solved in polynomial time (Johnson et al., 1988). In optimization problems, local search corresponds to the change of initial solution to an improved solution by performing some allowed tour operations like swap, delete, repair to obtain the neighborhood of the initial solution having better fitness value or cost and this process is repeated until no further improvements can be made. The 2-opt technique was first proposed by Croes (1958) in the context of traveling salesman problem. Later, 3-opt technique was proposed using the theory of 2-opt (Lin, 1965). After that, local search with 2-opt and 3-opt operations are most widely used for combinatorial optimization problems (Mavroidis et al., 2007). The idea behind 3-opt local search is that 3 edges are removed from an edge set to form three sub-tours of the current route and then the removed edges are reconnected in a way to form a new route (Rocki & Suda,

2012). The newly formed route is accepted if it has a lesser cost than the current route and the process is repeated till all combinations of removal and re-insertion of 3 edges are exhausted. There are seven ways to reconnect the route after removing 3 edges from it (excluding the original route). Out of those seven, there exists only four 3-opt moves and the rest are 2-opt moves. The modified procedure for 3-opt local search is shown in **Algorithm 2.**

ALGORITHM 2: 3-OPT LOCAL SEARCH (PATH, INSTANCE, P= -1)

```
count = 0
if p < 0 then
    count = p − 1
else
    count = 0
current_path ← path
evaluate fitness of current_path, fitness(current_path)←fitness(path)
fp = fitness(current_path)*2
while count < p do
    best_path ← current_path
    fitness(best_path) ← fitness(current_path)
    path1 = local search 2 opt (path, instance)
    seed ← current_path
    fitness(seed) ← fitness(current_path)
    for i in range 0 to length (path) − 3 do
        for j in range i + 1 to length (path) − 2 do
            for k in range j + 1 to length (path) − 1 do
                path2 = path[: i + 1] + path[j + 1: k + 1] + path[i + 1: j + 1]
                        + path[k + 1: ]
                path3 = path[: i + 1] + path[j + 1: i + 1] + path[k + 1: j + 1]))
                        + path[k + 1: ]
                path4 = path[: i + 1] + path[k + 1: j + 1] + path[i + 1: j + 1]
                        + path[k + 1: ]
```

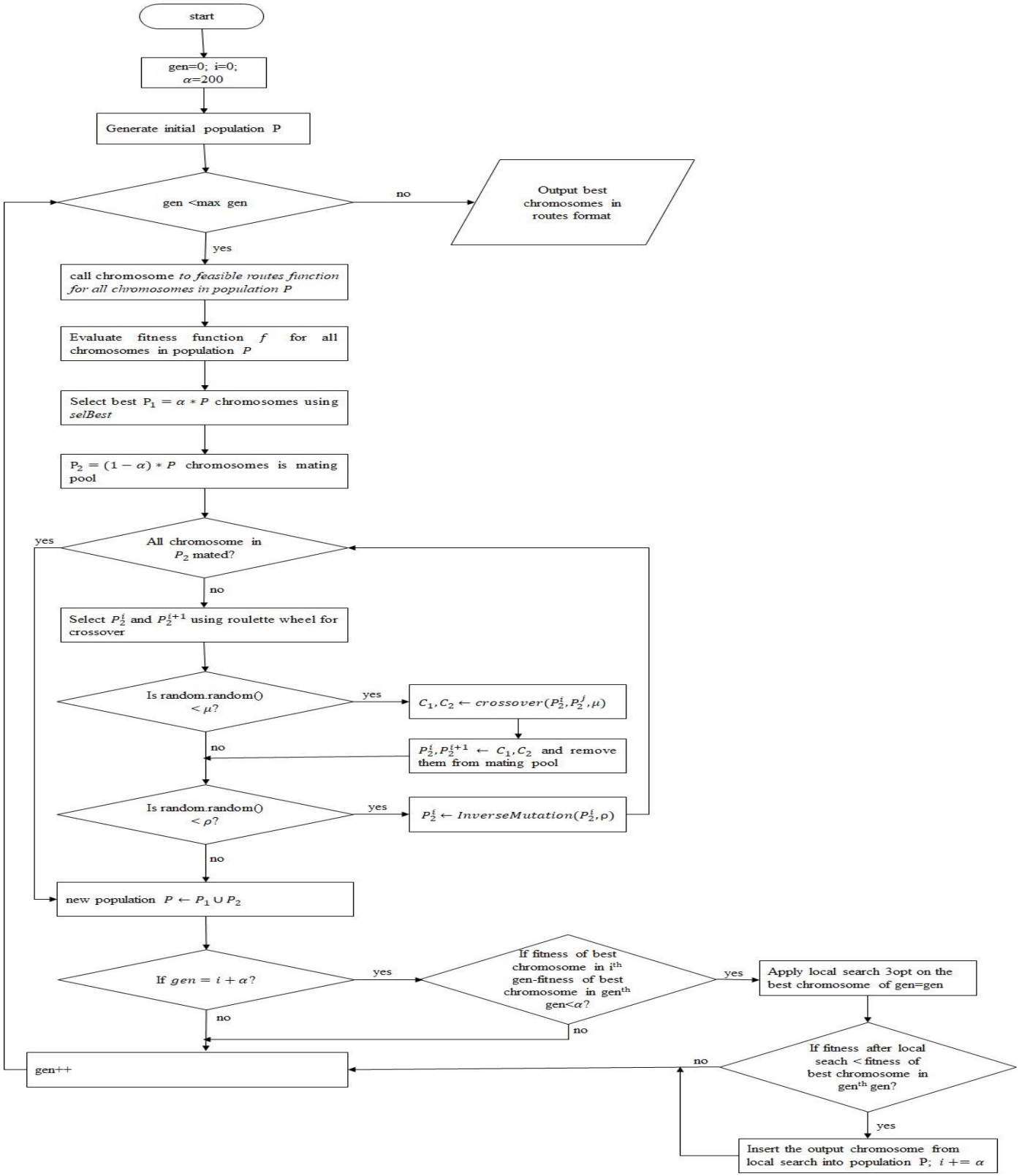
```
21                  path5 = path[: i + 1] + path[j + 1: k + 1] + path[j + 1: i + 1]
                                          + best_path[k + 1: ]
22                  evaluate fitness of path1, path2, path3, path4, path5
23                  best_path ← path with minimum fitness value among [path1, . . , path5]
24              end for
25              if fitness(best_path) < fitness(current_path) then
26                  for n in range 0 to length(current_path) do
27                      current_path[n] = best_path[n]
28                  end for
29                  fitness(current_path) ← fitness(best_path)
                end if
30              best_path ← seed
31              fitness(best_path) ← fitness(seed)
32          end for
33      end for
34      count + 1
35      if fp > fitness(current_path) && p < 0 then
36          fp = fitness(current_path)
37          count = −2 ; p = −1
38      if fitness(current_path) >= fp && p < 0 then
39          count = −1 ; p = −2
40  end while
41  return current_path, fitness(current_path)
```

4.1.3.3. Hybrid Genetic Algorithm with Local Search

In this work, we have hybridized the two metaheuristics discussed above that is elitist GA and 3-opt local search with a view to synergize them. It is observed that due to huge set of constraints in MDVRP-TW-SP-SD, traditional elitist GA converges slowly, especially for larger instances. To accelerate the convergence of GA, we use 3-opt local search in two

different ways. In the first approach, the local search is carried out after a fixed number of generations depending upon the improvement realized and in the second approach, the local search is carried out on the best solution reported in each generation of elitist GA. In the first strategy, the number of breakpoints for the application of 3-opt local search are predefined. If the elitist GA fails to improve the best-known solution by a threshold value within two consecutive breakpoints, the 3-opt local search routine will be invoked. If the improvement exceeds the threshold value, the elitist GA is continued. This strategy is referred to as GA_LS_1 in the remainder of the study. The second strategy is straightforward where irrespective of the improvement realized, 3-opt local search is exercised on the best solution reported in every generation. This strategy is referred hereafter as GA_LS_2. The implementation detail of the hybrid approach is delineated using a flowchart given in Figure 4.4.

Figure 4.4: Flowchart of elitist GA hybridized with local search



4.1.4. Computational Experiments

Our contributions through this work are two-fold. Firstly, we contribute in the research on the food bank operations, particularly surplus food recovery and redistribution. Secondly, we present a generic model for a rich variant of VRP that is MDVRP-TW-SP-SD and contribute to the body of literature in the broad domain of VRP. We therefore design our computational experiments to justify these two aspects. In particular, we first present the results of computational experiments carried on benchmark problem instances. A case of the Robin Hood Army in India is then discussed to demonstrate the applicability of the proposed model to the practice. The broad objectives of our computational experiments are:

- To validate the developed model and to demonstrate its applicability to practice.
- To evaluate the efficacy of proposed algorithms.
- To analyze the performance of the proposed algorithm GA_LS_1 and GA_LS_2 concerning to elitist GA.
- To assess the trade-off between solution quality and computational time for proposed algorithms.
- To generate managerial insights from the case study.

We have conducted all the numerical experimentation on Param Shivay supercomputer having 2*Intel Xeon SKL G-6148 processor, 2.4 GHz processing speed, 25GB/sec memory speed, 384 GB RAM and 212 CPU compute nodes. The proposed algorithms were implemented in python programming language. For deploying GA efficiently, we have used the DEAP framework of python. We have also used GUROBI state-of-the-art-solver to assess the performance of proposed algorithms with respect to it. The entire computational experiment is divided into two parts. The first parts solve the proposed algorithm on benchmark instances of

multi-depot VRP. The description about the benchmark dataset is given in section 4.1.4.1 and later results are presented in section 4.1.4.2. The second part comprises the case study on the Indian food banks which is given in section 4.1.3.

The results of computational experiments are presented in the following sub-sections.

4.1.4.1. Results on the benchmark problem instances

4.1.4.1.1.Dataset

Computational experiments were performed on a total of 180 instances of MDVRP-TW-SP-SD. The problem setting is novel in our case and hence there are no benchmark instances of MDVRP-TW-SP-SD to scrutinize the performance of proposed algorithms. Therefore, we have developed 180 instances of MDVRP-TW-SP-SD based on 20 benchmark instances of MDVRPTW proposed by Cordeau et al. (2001) by manipulating the demands of customer nodes to address the effect of multiple visits to a customer node for splitting the pickup or delivery demand. We have classified the experiments into three classes –

i) Total pickup demand is equal to total delivery demand

ii) Total pickup demand falls short of the total delivery demand, and

iii) Total pickup demand exceeds total delivery demand.

Each class is further divided into three sub-classes having demand ranges between ([0.1,1.2], [0.3,0.8] and [0.8,1.2] of vehicle capacity) and these sub-classes are numbered from 1 to 9. Each sub-class consists of 20 instances obtained by maneuvering 20 benchmark instances of Cordeau et al. (2001) according to the characteristic of respective class and sub-class, yielding a total of 20*3*3=180 instances. The 20 instances are named from p1 to p20. The pickup demand of any customer is randomly generated integer in the interval $[\boldsymbol{\alpha C_k}, \boldsymbol{\beta C_k}]$ in uniform distribution where $\boldsymbol{C_k}$ is the capacity of vehicle $\boldsymbol{k}$ and $[\boldsymbol{\alpha}, \boldsymbol{\beta}]$ is the demand range such that $\mathbf{0} <$

$\boldsymbol{\alpha < \beta < 1.2}$. Similarly, the delivery demand of any customer is a randomly generated integer in a uniform distribution in the interval $[-\boldsymbol{\beta C_k}, -\boldsymbol{\alpha C_k}]$. The description of instances of the dataset is given in Table 4.2.

Table 4.2: Specification of instances in the proposed dataset

Classes (Demand range $[\boldsymbol{\alpha C_k}, \boldsymbol{\beta C_k}]$)	**Sub-classes (Relation between supply and demand)**	**Id**	**Instances**
$\alpha = 0.1$, $\beta = 1.2$	Total pickup=total delivery	1	p1_1 to p20_1
	Total pickup>total delivery	2	p1_2 to p20_2
	Total pickup<total delivery	3	p1_3 to p20_3
$\alpha = 0.3$, $\beta = 0.8$	Total pickup=total delivery	4	p1_4 to p20_4
	Total pickup>total delivery	5	p1_5 to p20_5
	Total pickup<total delivery	6	p1_6 to p20_6
$\alpha = 0.8$, $\beta = 1.2$	Total pickup=total delivery	7	p1_7 to p20_7
	Total pickup>total delivery	8	p1_8 to p20_8
	Total pickup<total delivery	9	p1_9 to p20_9

All the other details such as vehicle capacity, number of vehicles, time window of nodes, locations of nodes, number of depots, service time of the original benchmark instances remain same in the proposed instances given in Table 4.3.

Table 4.3: Specifications of generated instances

Instances	**Depots**	**Vehicles**	**Customers**	**Vehicle Capacity**
p1_1 to p1_9	4	2	48	200
p2_1 to p2_9	4	3	96	195
p3_1 to p3_9	4	4	144	190
p4_1 to p4_9	4	5	192	185
p5_1 to p5_9	4	6	240	180
p6_1 to p6_9	4	7	288	175
p7_1 to p7_9	6	2	72	200

p8_1 to p8_9	6	3	144	190
p9_1 to p9_9	6	4	216	180
p10_1 to p10_9	6	5	288	170
p11_1 to p11_9	4	1	48	200
p12_1 to p12_9	4	2	96	195
p13_1 to p13_9	4	3	144	190
p14_1 to p14_9	4	4	192	185
p15_1 to p15_9	4	5	240	180
p16_1 to p16_9	4	6	288	175
p17_1 to p17_9	6	1	72	200
p18_1 to P18_9	6	2	144	190
p19_1 to p19_9	6	3	216	180
p20_1 to p20_9	6	4	288	170

The fixed cost of vehicles in the dataset is taken as 500 units and this value is motivated from the real cost of renting/hiring a vehicle. The penalty for 1 unit of unmet pickup or delivery demand is taken as 1000 units since this value provides better results in the proposed problem context.

4.1.4.2. Results on test instances

One compute node was assigned for each instance and all the three proposed algorithms were run 5 times for each instance. Moreover, the first sub-class of instances (instance id=1) were also run using the GUROBI solver. The parameters used in the proposed algorithms are population size, crossover probability, mutation probability and number of generations. Parameter tuning is quite complex to perform on a larger dataset. However, we have tried different combinations of values for all four parameters and performed extensive experiments to analyze the performance of proposed algorithms on these values. For this study, the combination of values of parameters used in the experimentation is given in Table 4.4.

Table 4.4: Parameter values used in experimentation

Parameter	Values
Population size	50-100-150
Crossover probability	0.7-0.75-0.8
Mutation probability	0.25-0.2-0.1
Number of Generations	100-500-1000-5000-10000-15000

After experimentation, we analyzed that good results were obtained when crossover probability is 0.7 and mutation probability is 0.2. The population size and number of generations was set according to the algorithm. For elitist GA, more number of generations and larger population size were required as compared to the GA hybridized with local search (GA_LS_1 and GA_LS_2). Larger populations and generation sizes were also used for larger instances.

The best objective function value, average objective function value of five runs, CPU time in seconds for each instance of sub-class Id 1 of the MDVRP-TW-SP-SD using GA, GA_LS_1 and GA_LS_2 are given below in Table 4.5. For elitist GA, the best objective function value and the corresponding CPU time for each instance are reported in columns 2 and 3, respectively. Similar results based on the average of all five runs are presented in columns 4 and 5, respectively. Results for other algorithms are also presented similarly in the respective columns. For all other sub-classes (id=2 to 9) the results are reported in Appendix – B (Table B-1).

It is observed that GA_LS_2 which is hybrid GA with 3-opt local search after each generation takes substantial time. It however provides better results as compared to the other two algorithms. To have a fair understanding of the performance of the algorithm in terms of the best solution value reported, we calculate an effectiveness score. In general, the effectiveness of algorithm A over algorithm B for any problem instance is calculated as follows.

$$\text{Effectiveness}\ (\%) = \left(\frac{Z_A - Z_B}{Z_A}\right) * 100$$

A positive effectiveness score of algorithm A over algorithm B indicates that algorithm B outperforms algorithm A and vice versa. A larger score indicates a significant difference in the performance of the algorithms. Primarily, A%, B%, and C% given in Table 4.5 indicates effectiveness scores of GA over GA_LS_2, GA over GA_LS_1 and GA_LS_1 over GA_LS_2, respectively. The average effectiveness score of GA over GA_LS_2 is 8.741% as reported in Appendix - B (Table B-1). This means that the GA_LS_2 is performing better than GA by around 8% on an average basis. Similarly, the average effectiveness score of GA over GA_LS_1 is 2.248% for the best objective function values. The effectiveness score of GA_LS_1 over GA_LS_2 is 6.667% signifying the outperformance of the GA_LS_2. These percentage values were obtained by calculating the average percentage of effectiveness score (A %, B% and C% respectively) for all the data set (from subclass-ID-1 to 9) Therefore, the additional computational time required for GA_LS_2 pays off in terms of better objective function value on an average basis.

In addition to the proposed algorithm, we have utilized GUROBI solver to solve the model on the proposed instances and imposed a time limit of 24 hours for each instance. As we can see from Table 4.5, GUROBI is unable to load instances p5_1, p6_1, p9_1, p10_1. P15_1, p16_1, p19_1, and p20_1. All these instances have more than 192 customers as shown in Table 4.3. Moreover, it produces solutions with very large optimality gaps even after setting a huge time limit. Thus, GUROBI is failed to solve the test instances to optimality and even to feasibility in many problem instances. This comparison also demonstrated the need for the efficient solution approaches.

Table 4.5: The results of elitist GA, GA_LS_1, GA_LS_2 on sub-class ID-1

Instances	GA				GA_LS_1				GA_LS_2				A (%)	B (%)	C (%)
	Best solution		Average values		Best solution		Average values		Best solution		Average values				
	Obj. value	CPU(s)	Obj. value	CPU(s)	Obj. value	CPU(s)	Obj. value	CPU(s)	Obj. value	CPU(s)	Obj. value	CPU(s)			
p1_1	2222	1249	2424	1335	2178	2017	2598	2114	**2020**	3548	2241	3647	9.09	1.97	7.26
p2_1	**1952**	2088	2889	2597	2504	3069	3021	3128	2479	6787	2675	6875	-27.00	-28.25	0.98
p3_1	3609	3624	3082	3789	2903	4072	3395	4259	**2685**	9479	2825	9645	25.59	19.56	7.50
p4_1	4077	5310	3401	5542	3152	5822	3534	5997	**2808**	12549	3149	13751	31.12	22.69	10.90
p5_1	5984	8318	4918	8475	4749	8435	4953	8542	**4279**	17142	4469	17426	28.49	20.64	9.89
p6_1	**4148**	10144	4994	11254	5028	13384	5206	13978	4322	21044	4474	21452	-4.19	-21.22	14.05
p7_1	**2316**	1575	3236	1678	2833	2914	3455	3052	2641	4795	3002	4997	-14.03	-22.32	6.78
p8_1	**1865**	3859	3487	3932	3251	4515	3690	4675	3029	7510	3198	7804	-62.41	-74.31	6.83
p9_1	5153	7406	3902	7894	3639	8200	4259	8546	**3145**	15251	3537	15842	38.96	29.39	13.56
p10_1	**4040**	11396	5670	16459	4966	13031	5829	13452	4692	20544	5247	21073	-16.13	-22.92	5.52
p11_1	3611	1339	2652	2015	2496	2077	2659	2195	**2244**	3825	2390	3991	37.87	30.89	10.10
p12_1	3822	2237	2898	3657	2466	3271	2986	3358	**2608**	6886	2702	6925	31.78	35.49	-5.75
p13_1	4398	3859	3656	5002	3275	4606	3741	4714	**3034**	9618	3381	10121	31.00	25.53	7.35
p14_1	**2950**	5599	4369	6785	4043	6021	4617	6281	3733	12734	4036	13175	-26.53	-37.04	7.67
p15_1	5044	7894	5067	9514	4425	8854	5502	9026	**4014**	16930	4724	17215	20.43	12.26	9.30
p16_1	6493	9722	5345	10643	5056	13206	5550	13895	**4629**	21152	4905	21642	28.71	22.13	8.44
p17_1	**1928**	2237	3587	2527	2953	2954	3983	3145	2889	4829	3335	5124	-49.79	-53.13	2.18
p18_1	4548	4131	3569	4482	3128	4926	3624	5172	**3054**	7596	3283	7942	32.86	31.24	2.35
p19_1	4297	7143	4442	7259	3753	8595	4640	8676	**3374**	15094	4171	15883	21.47	12.65	10.10
p20_1	**3367**	10998	4646	12721	4303	11955	4837	12361	3956	21567	4278	22234	-17.51	-27.80	8.05

A%= The effectiveness score of elitist GA over GA_LS_2; B%= The effectiveness score of elitist GA over GA_LS_1; C%= The effectiveness score of GA_LS_1 over GA_LS_2

Table 4.6 presents the performance of different algorithms on aggregated basis on the given objective function values and CPU times. The average of best values of objective function for GA, GA_LS_1, and GA_LS_2 is 51104, 49492, and 47163 respectively. As it can be seen from the Table 4.6 that GA_LS_2 outperforms the other two algorithms for all sub-classes except for subclass Id-5. Sub-class Id-5 corresponds to case where total supply exceeds total demand and belongs to the class in which demand of all the nodes lies in the interval [0.3, 0.8] times of the vehicle capacity. between for sub-class Id-5 GA performs better. For the cases in which total pickup equated to total demand no penalty is incurred (Subclass Id's 1,4 and 7). By comparing the average results of Sub-class Id 2 with 3, 5 with 6, and 8 with 9, it is worth noting that all three algorithms perform better for cases where total pickup is less than total delivery (sub-class Id's 3, 6, and 9).

Table 4.6: Aggregated values of the objective function on sub-classes

ID	GA				GA_LS_1				GA_LS_2			
	Best solution		Average values		Best solution		Average values		Best solution		Average values	
	Obj. value	CPU (s)	Obj. value	CPU (s)	Obj. value	CPU (s)	Obj. value	CPU (s)	Obj. value	CPU(s)	Obj. value	CPU(s)
1	3791	5506	3912	6378	3555	6596	4104	6828	**3282**	11944	3601	12338
2	84799	5614	85132	5799	81890	6565	82619	6789	**76263**	11945	76485	12212
3	67212	5632	67545	5830	64777	6558	65326	6764	**60799**	12017	61118	12284
4	3478	5666	3831	5866	3526	6775	3987	6971	**3116**	12015	3472	12291
5	**67198**	5523	67475	5716	68355	7140	68904	7348	70745	11919	71027	12194
6	62355	5490	62708	5672	60180	6718	60608	6929	**56033**	12071	56388	12350
7	4495	5606	4788	5798	4357	6524	4781	6719	**3815**	12167	4183	12443
8	93138	5351	93430	5562	87807	6597	88271	6799	**83681**	12206	83962	12477
9	73473	5536	73765	5738	70979	6678	71402	6906	**66732**	11818	67100	12113
Average	**51104**	**5547**	**51398**	**5818**	**49492**	**6684**	**50000**	**6895**	**47163**	**12011**	**47482**	**12300**

The convergence fashion of all three algorithms differs uniquely. As shown in Figure 4.5, GA_LS_2 converges more sharply than the other two algorithms. The minimum fitness function of instance p1_1 over different generation for all three algorithms is plotted in Figure 4.5. Although GA_LS_1 does not show any substantiality in terms of providing

good solutions with respect to elitist GA. However, it takes a lesser number of generations to reach near-optimal solution.

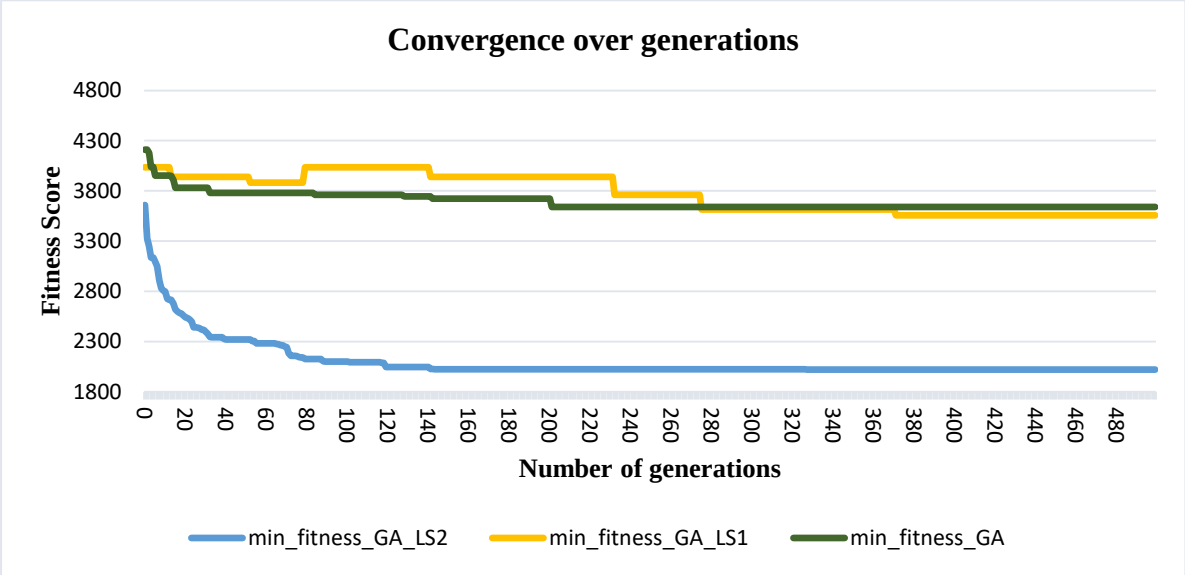


Figure 4.5: Convergence diagram of GA, GA_LS1, and GA_LS_2 algorithms

To ensure statistical resilience of the proposed hybrid GA algorithms (GA_LS_1 and GA_LS_2), statistical testing has been done. Since, our results contain 3 groups corresponding to GA, GA_LS_1 and GA_LS_2 respectively, we have used one-way analysis of variance (ANOVA) to prove the statistical relevance of comparing the performance of the three algorithms over their best-reported value. ANOVA is used to prove that at least one or more group mean is significantly different. IBM SPSS statistics version 28 software was used for the analysis of data. The null hypothesis in ANOVA is given below-

Hypothesis
H_0: $\mu_1 = \mu_2 = \mu_3$ (all groups mean is same i.e., all algorithms performs same)
H_A: At least one algorithm is significantly different

The null hypothesis H_0, is accepted if the significance value is greater than 0.05 and rejected if it is lesser than 0.05 for 95% confidence interval. Table 4.7 shows the results of ANOVA applied on the results of instance category where total pickup demand is equal to total delivery demand.

Table 4.7: Results of ANOVA

Test of Homogeneity variances					
Levene's statistic	df1	df2	Significance		
1.060	2	177	0.349		
ANOVA					
	Sum of Squares	**df**	**Mean Square**	**F**	**Significance**
Between Groups	7732755	2	3866378	4.132	0.018
Within Groups	1.656e+8	177	935624.9		
Total	1.733e+8	179			

The value of significance factor was $p = 0.018$ which is less than 0.05. So, we reject the null hypothesis and conclude that all algorithms perform differently.

4.1.4.3. Case study

In this section, we present a case of a volunteer-based surplus food recovery and redistribution unit namely Robin Hood Army (Robin Hood Army, n.d.). The RHA is an international volunteer-based organization currently operating in around 10 countries in Asia, Africa and Latin America. The journey of RHA starts from a very small-scale operation of feeding 50 unfortunate people in the night drives per week in the year 2014. According to current statistics, RHA had served over 28 million unfortunate people across 159 cities (Robin Hood Army, n.d.). Robin Hood Army is modeled according to Portugal's redistribution food program (I. Martins et al., 2011). In India, they have a network over 100 major cities. We contacted the state manager of RHA from Uttar Pradesh, India to get

acquainted with their daily operations and get the desired information for their surplus recovery points, delivery points, demand, surplus food quantity, time window of their operation, and number of vehicles and volunteers and we thankfully received this information for Lucknow city which is the capital of Uttar Pradesh state in India from the RHA officials. The RHA primarily works on a front-end model that is it practices the collection and distribution of ready-to-eat (cooked) food. A complete list of 54 locations with location names is shared with us. Out of these, 25 locations are surplus/leftover recovery points (i.e. pickup locations) from where they get requests for surplus/ leftover food and 29 locations are delivery locations shown where they used to distribute foods on day-to-day basis. According to the insights of executives and ground-level volunteers, their most of the operations are performed in three localities where most of the volunteer resides. Therefore, we have considered these 3 volunteer hub locations as the depots for modeling the problem in multi-depot variant. The geographical location of pickup nodes, delivery nodes and depots are plotted using the ArcGis software and presented in Figure 4.6. The actual distances and travel time of two locations are computed by using geographical coordinates and Geopy module in python programming language.

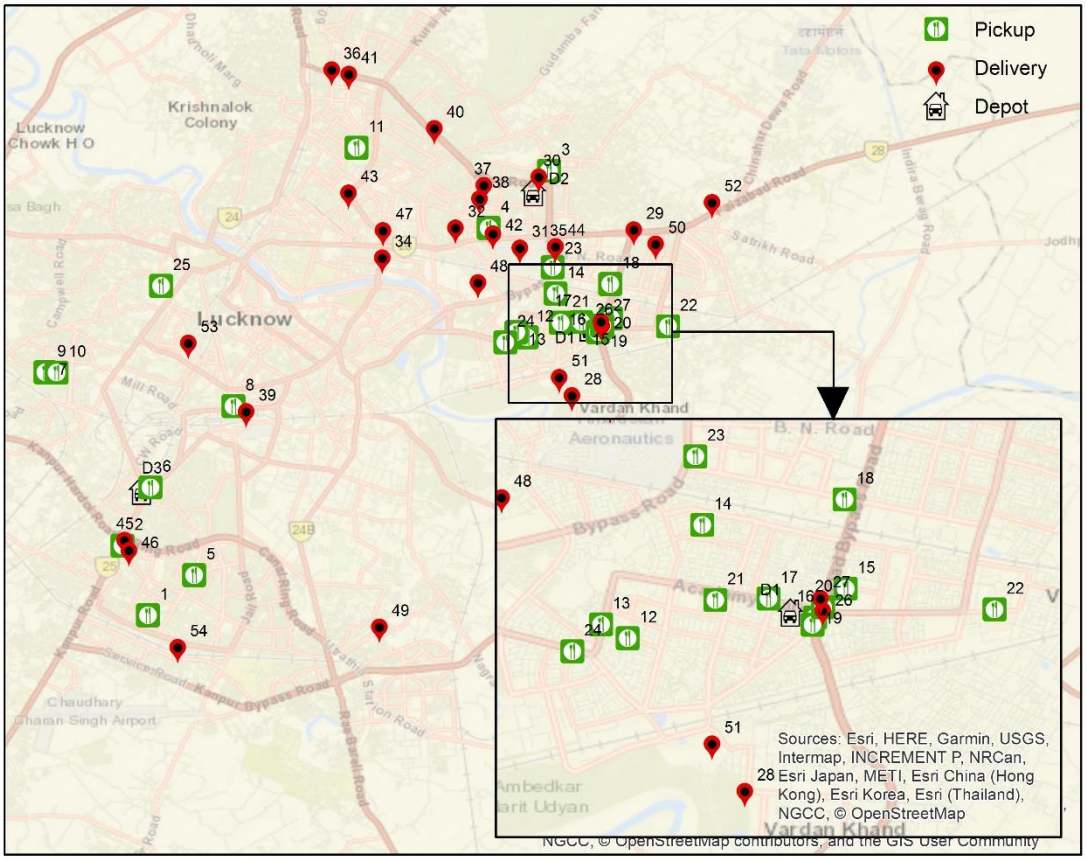


Figure 4.6: Location of depots, pickup, delivery sites on Lucknow Map

Unfortunately, the past record of each individual pickup and delivery made was not maintained with RHA. However, we could receive the data for the total number of meals picked up and distributed per day for a recent month. Further, with the help of ground-level volunteers, we estimated the average figures for surplus/leftover food from each pickup point and demand at the delivery sites. It is observed that most of the delivery sites of RHA are slum locations. The total demand of food at the delivery location is considered to be around 5-20% of the total population of that area. The demand was estimated by considering the past experience and insights from the RHA volunteers. It was also observed that usually, the total delivery demand exceeds the total supply donations. The service time for each location is proportional to the quantity of pickup demand or delivery demand. The latitude, longitude, service time and demand of pickup and delivery locations are given in Appendix - B (Table B-2).

In the Indian scenario, both two-wheeler vehicles (e.g. scooters and motorbikes) and four-wheeler vehicles are commonly used as means of commute. Therefore, at each depot we considered a heterogeneous fleet comprising of both two-wheeler and four-wheeler vehicles. Capacity of these vehicles is assumed to be 50 units (i.e. number of meal units' equivalent) and 100 units for two-wheelers and four-wheeler vehicles, respectively. These vehicles can either be owned/possessed by the volunteers or need to be hired from the transport service providers.

From the past experience of availability of vehicles, we have considered 3 two-wheeler vehicles and 13 four-wheeler vehicles at each depot. It is also shared with us that there are around 500 volunteers registered with RHA. However, the actual number of volunteers available in a day could be as small as 30. Also, with each vehicle going for food drive at least two volunteers are needed. Therefore, the number of volunteers available at each depot is considered as double the number of vehicles at that depot. The fixed cost of two-wheeler vehicles and four-wheeler vehicles are considered as 250 and 500 respectively. Practically, if the vehicles are owned by volunteers, then such cost might seem to be irrelevant. However, there are multiple reasons for considering these fixed costs. If the vehicles are to be hired, then the fixed cost is justifiable. Further, as indicated in the work of (Dubey & Tanksale, 2022), substantial reliance on volunteers is an obstruction for the Indian food banks and they need paid employees for smooth functioning of day-to-day work. Such hiring or outsourcing of collection/distribution part comes with a cost. Also, the objective could be to use the optimal number of volunteers/vehicles to perform the drives efficiently. Consideration of the fixed cost will play a key role in such selection. Next, the penalty cost for each unit of unmet demand (pickup or delivery) of any node is taken as 1000 to demotivate any such occurrence. A time window of 4.5 hours is considered for fulfilling each

pickup or delivery request from the practical viewpoint to avoid wastage of the ready-to-eat food.

It should be noted here that the problem settings described above are representative in nature based on the inputs received from the food bank and they might change significantly. We try to consider such variations in parameters by building different scenarios. The above-mentioned case description is considered as a base case as summarized in Table 4.8.

Table 4.8: Description of supply and demand donation of base scenario

Instance	**Relationship between supply and demand**	**No. of volunteers available on each depo**	**Description of vehicles on each depo**	
			Two- wheeler (Number, capacity, fixed cost per vehicle)	**Four-wheeler (number, capacity, fixed cost per vehicle)**
Base case	Total supply donation is less than the demand	32	(3, 50, 250)	(13, 100, 500)

The proposed algorithms were executed on base case scenario. The results of elitist GA, GA_LS_1 and GA_LS_2 are given in Table 4.9.

Table 4.9: Results of the proposed algorithm on base-case scenario

Algorithm	**Best Obj***	**CPU (s)**	**Number of two-wheeler vehicles**	**Number of four-wheeler vehicles**	**Number of routes formed**	**Utilization of volunteer (%)**	**Total routing cost**
GA	1391596	1094	0	5	5	10.41	2096
GA_LS_1	1391895	553	0	6	6	12.5	1895
GA_LS_2	1391569	2065	1	5	6	12.5	1819

It is observed from the results that both GA and GA_LS_2 produce slightly better results in comparison to GA_LS_1. However, GA_LS_1 is the most efficient in terms of computational time among the three. Overall, GA_LS_2 provides the lowest routing cost and better volunteer and vehicle utilization. Therefore, we focus on the results of GA_LS_2 for further discussion. The total pickup and delivery demand in the base case scenario are 1198 and 2585 units respectively. Therefore, at most 1198 units of delivery demand could be satisfied and remaining 1387 units' delivery demand couldn't be met. Since we have imposed a huge penalty cost on each unit of unmet demand, therefore our model tries to maximize as much delivery demand as possible. In this case, the algorithm tries to minimize the routing cost and number of vehicles while trying to maximize the pickup demand to avoid the penalty. It is observed that all the pickup demands available are respected to meet the maximum portion of the delivery demands. Moreover, the utilization of two wheeler vehicles is low in comparison to the four-wheelers regardless of their higher fixed costs. One possible reason for this could be the limited capacity of two-wheelers vehicles that leads to too many splitting of demands and resulting in multiple visits to the same nodes. This in turn would lead to increased routing costs.

We now present the details of the solution obtained by GA_LS_2 for the base case scenario. The total transportation cost incurred is 1819 units and total fixed cost is 2750 for five 4-wheeler and one 2-wheeler vehicle. The total penalty cost incurred is 1387000 which is proportional to the 1387 units of the excess delivery demand. A total of 6 routes are formed, two from each depot, whose details are as given in Table 4.10. To visualize the results a bit further, the first route from depot 3 using vehicle 16 is shown on the Open Street map of Lucknow using ArcGis software in Figure 4.7.

Table 4.10: The routes formed by GA_LS_2 on current existing scenario S-1

V-16 route	D3	22	17	51	5	15	42	24	31	6	37	23	D3
A		92	40	-129	74	31	-59	72	-147	39	13	-115	
B		92	40	-129	74	31	-59	72	-147	39	13	-115	
C		92	8	-100	74	26	-59	59	-100	39	13	-52	
D		0	32	-29	0	5	0	13	-47	0	0	-63	

V-14 route	D2	19	49	7	52	27	13	50	4	39	12	35	D2
A		14	-98	65	-29	-82	35	-79	67	-43	89	-143	
B		14	-98	65	-29	-82	35	-79	67	-43	89	-143	
C		14	-14	65	-29	-36	35	-35	67	-43	76	-76	
D		0	-84	0	0	-46	0	-44	0	0	13	-67	

V-8 route	D3	3	36	20	47	8	1	40	14	54	D3
A		11	-110	38	-100	36	55	-49	80	-73	
B		11	-110	38	-100	36	55	-49	80	-73	
C		11	-11	38	-38	36	55	-49	31	-73	
D		0	-99	0	-62	0	0	0	49	0	

V-16 route	D1	10	16	44	25	30	2	33	18	9	32	21	29	11	28	D2
A		29	73	-82	61	-10	10	-94	73	11	-130	13	-134	77	-18	
B		29	73	-82	61	-10	10	-94	73	11	-130	13	-134	77	-18	
C		29	71	-82	61	-10	10	-79	73	11	-84	13	-13	18	-18	
D		0	2	0	0	0	0	-15	0	0	-46	0	-121	59	0	

V-1 route	D2	17	15	24	51	44	36	D2
A		40	31	72	-129	-110	40	
B		32	5	13	-29	-99	32	
C		32	5	13	-29	-42	32	
D		0	0	0	0	-57	0	

V-4 route	D1	16	26	21	14	35	12	27	11	43	25	53	9	45	2	46	D1
A		73	-53	13	80	-143	89	-82	77	-132	61	-105	11	-62	10	-138	
B		2	-53	13	49	-119	13	-46	59	-132	61	-105	11	-62	10	-138	
C		2	-2	13	49	-62	13	-13	59	-59	61	-61	11	-11	10	-10	
D		0	-51	0	0	-57	0	-33	0	-73	0	-44	0	-55	0	-128	

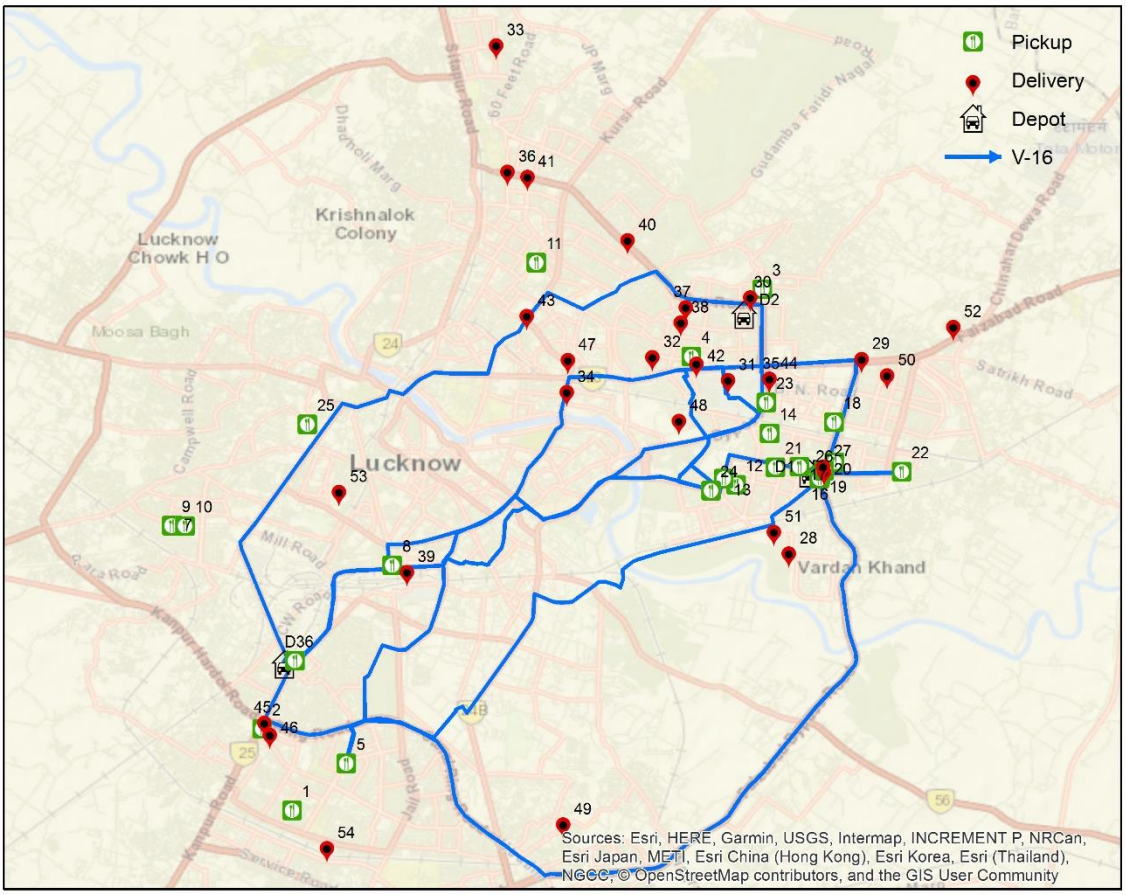


Figure 4.7: Plot of the first route from depot 3 using vehicle number 16

Out of 25 pickup nodes and 29 delivery nodes, the number of nodes visited multiple times due to splitting of pickup or delivery demand splitting are 7 and 5, respectively. This means that around 28% of the pickup demand and 17% of delivery demand are getting split or met by more than one vehicle. The reason for this fragmentation on the demand nodes could be either due to pickup/delivery requests exceeding the vehicle capacity or due to limited capacity/commodity remaining on the serving vehicle en route.

Food bank phenomena involve uncertainties related to amount of donation, demand of food and availability of volunteers (Brock & Davis, 2015). For example, during festivals and holidays, amount of food donations increases. Similarly, cases arise where delivery demand diminishes for some areas. To take into account the reverberations of these uncertainties, we have constructed various scenarios on top of the base case scenario. In particular, 5

scenarios are generated following the variations in supply, demand and number of volunteers available. The description of these variation along with number of two and four-wheeler vehicles and number of volunteers on each depot, its capacity and fixed cost is given in Table 4.11.

Table 4.11: Description of variation on generated scenarios

Scenario type A: Keeping the delivery demand fixed and varying supply donation				
Instance	Description of scenarios	No. of volunteers available on each depo	Vehicles description on each depo	
			Two- wheeler (number, capacity, fixed cost per vehicle)	Four-wheeler (number, capacity, fixed cost per vehicle)
S-1	Total supply donation is equal to demand	32	(3,50,250)	(13,100,500)
S-2	Total supply donation exceeds the demand	32	(3,50,250)	(13,100,500)
Scenario type B: Keeping the supply donation fixed and varying delivery demand				
S-3	Total supply donation is equal to demand	32	(3,50,250)	(13,100,500)
S-4	Total supply donation exceeds the demand	32	(3,50,250)	(13,100,500)
Scenario type C: Varying the number of volunteers				
S-5	Total supply donations are less than the demand (similar to base case scenario)	8	(2, 50, 250)	(2, 100, 500)

In scenario type A, we fixed the delivery demand and induced variations on the supply donation as follows.

- Supply donation escalates to a degree that it becomes equal to the total delivery demand (S-1).
- Supply donation ascents in such a way that it becomes greater that the total delivery demand (S-2).

Similarly, in scenario type B, we have fixed the supply donation to explore the effects of variation on the delivery demand (S-3 and S-4). Scenario type C deals with the unavailability of volunteers to carry out the collection and distribution drives. Thus, it significantly hampers the performance of food banks and therefore we supplement the current existing scenario by inducing variation on the number of volunteers available (S-5).

The results of elitist GA, GA_LS_1 and GA_LS_2 on the supplemented scenarios is given in Table 4.12, 4.13 and 4.14, respectively.

Table 4.12: Results of elitist GA on the generated scenarios

Type	Objective function Value using elitist GA	Number of two-wheeler vehicle used	Number of Four-wheeler vehicle used	Number of routes formed	Utilization of volunteers (%)	Total Routing cost	Time taken (in seconds)
S-1	4832	1	5	6	12.5	2082	1274
S-2	70773	0	5	5	10.4	2273	1285
S-3	5118	0	6	6	12.5	2118	1013
S-4	216799	1	3	4	8.33	2049	1070
S-5	3276889	1	4	5	10.4	1639	577

Table 4.13: Results of GA_LS_1 on the generated scenarios

Type	Objective function Value using GA_LS_1	Number of two-wheeler vehicle used	Number of Four wheeler vehicle used	Number of routes formed	Utilization of volunteers (%)	Total Routing cost	Time taken (in seconds)
S-1	4553	0	6	6	12.5	1553	703
S-2	70393	1	4	5	10.4	2143	617
S-3	4545	0	5	5	10.4	2045	593
S-4	217885	2	4	6	12.5	2385	515
S-5	3061398	0	5	5	10.4	1898	684

Table 4.14: Results of GA_LS_2 on the generated scenarios

Type	Objective function Value using GA_LS_2	Number of two-wheeler vehicle used	Number of Four wheeler vehicle used	Number of routes formed	Utilization of volunteers (%)	Total Routing cost	Time taken (in seconds)
S-1	4374	0	6	6	12.5	1374	2725
S-2	70911	1	5	6	12.5	2161	2368
S-3	4223	0	5	5	10.4	1723	2865
S-4	215971	0	3	3	6.25	1471	2666
S-5	2885026	1	4	5	41.67	1776	987

As we can see from Table 4.12, 4.13 and 4.14 that each scenario obtained from current existing scenario performs differently. For scenario S-1 all the three algorithm tries to satisfy as much delivery demand as the available total pickup demand henceforth incurring zero penalty for this case. Analyzing the routing cost, we can contemplate that GA_LS_2 performs best in nearly all scenarios. In addition, S-2 encounters most number of splitting of pickup and delivery demands in comparison to the other generated scenarios. Generally, when an adequate number of volunteers are available only a small percentage of total volunteers available are needed to perform the task of collection and distribution due to the small scale operation of RHA. Furthermore, four-wheeler vehicles are found more suitable and economical for collection and distribution with capacity constraints. The value of objective function using elitist GA for S-4 is 57.5% more than that of base case. Similarly, the value of objective function using GA_LS_1 and GA_LS_2 for the S-4 case is 54.5% and 51.8% more than the base case respectively. S-5 got developed using the base case scenario. All the specific details such as pickup demand of a node, delivery demand of the node, associated time windows of nodes of instance of base case remain same in S-5, only the number of volunteers available are diminished. In S-5 the utilization of volunteers increases to perform the task of collection and distribution (41.6%). In addition, due to the

limited availability of volunteers in S-5 (only 8 on each depot), the penalty cost due to unmet demand got increased.

4.1.5. Conclusion

This paper introduces a variant of the Vehicle Routing Problem (VRP) termed MDVRP-TW-SP-SD, specifically tailored to address the problems of surplus or leftover food collection and redistribution within the domain of food banks. Despite the extensive body of literature dedicated to VRP and its various derivatives, this particular class of problem has been subject to limited exploration. Notably, the integration of split pickup and split delivery with associated time windows in the context of multi-depot VRP remains uncharted in existing literature. Therefore, this paper not only contributes to the discourse on food banks but also enriches the VRP literature by presenting a novel problem formulation specifically, a mixed-integer problem formulation for MDVRP-TW-SP-SD. To tackle the computational complexity inherent in this problem, we propose both an elitist Genetic Algorithm (GA) and a hybrid GA coupled with local search methodologies, providing efficient and effective solutions to the outlined problem. This paper introduces two variants of a hybrid Genetic Algorithm (GA) - GA_LS_1 and GA_LS_2. In the course of computational experiments, a novel dataset comprising 180 instances, derived from benchmark instances for multi-depot VRP with time windows, is proposed. Statistical analysis employing ANOVA reveals a noteworthy difference in the performances of the both the algorithms. The classical elitist GA is observed to exhibit slow convergence, particularly for larger problem instances. Results from numerical experiments indicate that GA_LS_2 excels in terms of solution quality, while GA_LS_1 demonstrates computational efficiency in relation to CPU time. It is worth noting that the proposed hybridized genetic algorithms outperform the capabilities of the state-of-the-art solver GUROBI, which fails to produce feasible solutions for medium to large-sized instances.

Moreover, we offer a real-time case study focusing on the operations of the Robin Hood Army in Lucknow, an Indian food recovery and distribution unit. The findings of the case study demonstrate that approximately 28% of pickup nodes and 17% of delivery nodes experience fragmentation, signifying the splitting of pickup and delivery demand in the base case scenario. Additionally, the analysis indicates a higher utilization of four-wheeler vehicles as opposed to two-wheeler vehicles, attributed to the latter's limited capacity. Notably, scenario S-6 demonstrates the maximum utilization of volunteers among all the scenarios.

The MDVRP-TW-SP-SD offers promising avenues for future research to address various other realistic scenarios that has not been explored in the current study. Firstly, the model can be further enhanced by incorporating the complexities associated with the collection and distribution of multiple commodities and incorporating multiple time window slots for pickup and delivery. Secondly, while our current study primarily focuses on maximizing delivery requests, a subsequent research endeavor will explore the consideration of equitable distribution from a social perspective. This exploration will be presented in a distinct research article, featuring a different solution methodology and perspective. Thirdly, given the inherent uncertainty surrounding the supply and demand of donated food necessitates the endeavor of suitable modeling techniques such as two-stage stochastic optimization or robust optimization. These challenges will be systematically addressed in future research endeavors.

4.2. Multi-Objective Optimisation for Surplus Food Recovery and Redistribution

4.2.1. Introduction

In contrast to profit-driven businesses and firms, the primary objectives of food banks extend beyond mere profit maximization. Adivar et al. (2010) introduced the concept of the "social welfare chain" to encapsulate the activities supporting social programs within the supply chain. Furthermore, (Davis et al., 2016) highlighted that the operations of food banks align with the principles of a humanitarian supply chain, distinct from commercial supply chains, with a focus on achieving social objectives. Food banks strive to attain three primary objectives: effectiveness, efficiency and equity (Sengul Orgut et al., 2016). Recently, Hasnain et al. (2021) proposed a mechanism for eliciting preferences among efficiency, effectiveness, and equity, while Alkaabneh et al. (2021) introduced a dynamic programming approach to ensure these objectives in food bank operations. Liu et al. (2021) introduced a robust model for achieving these objectives in humanitarian relief operations during disasters. The literature also addresses the logistic challenges of food banks concerning equitable resource allocation (Eisenhandler & Tzur, 2019a, 2019b). This study endeavors to address these significant objectives of food banks within a multi-objective framework, aiming to optimize operational decisions within Indian food banks while considering the aforementioned multiple objectives.

The operational framework of Indian food banks distinguishes itself from counterparts due to distinctive operational characteristics. These food banks function through both front-end and back-end models, contingent on the specific distribution processes, the nature of donated food, and its shelf life. Front-end model food banks engage in the operations of food recovery and redistribution centers. Typically, these entities swiftly distribute ready-to-eat food, characterized by a limited shelf life, immediately after recovery. This work

extends the research conducted earlier in the Chapter 4.1 and fills the gap pertaining to inclusion of social objectives into the proposed model. The original work established a framework for addressing the operational challenges faced by Indian food banks through the lens of a single-objective multi-depot vehicle routing problem with time windows and split pickup and delivery (MDVRP-TW-SP-SD). This paper, however, expands the scope by delving into a multi-objective optimization perspective, introducing a richer problem formulation - the Multi-Objective MDVRP with Time Windows, Split Pickup, and Split Delivery (MO-MDVRP-TW-SP-SD). The transition from a single-objective to a multi-objective approach is pivotal, as it allows for a more comprehensive exploration of the intricacies within the operational landscape of Indian food banks. This shift to a multi-objective paradigm not only broadens the applicability of the model but also aligns with the real-world complexities and goals of food banks. By tackling multiple dimensions simultaneously, this paper enhances the decision-making framework for Indian food banks, contributing valuable insights. In Chapter 4.1, we addressed a single-objective MDVRP-TW-SP-SD problem within the context of Indian food banks. They applied a genetic algorithm and three opt-local search algorithms to solve the mentioned problem. In chapter 3, we highlighted the inadequacies in planning (strategic, tactical, and operational) within Indian food banks. It is essential to note that the scope of this paper is confined to addressing the operational decision planning aspects of food banks.

While this chapter builds upon the groundwork laid by our previous work pertaining to MDVRP, its contributions are manifold. This study addresses the three primary objectives of food banks effectiveness, efficiency, equity, as depicted in Figure 4.8. A multi-objective optimization problem is formulated to solve a multi-depot Vehicle Routing Problem (MDVRP) with demand splitting and time windows (MO-MDVRP-TW-SP-SD). Efficiency, within this context, is defined as the minimization of total transportation costs,

encompassing both total traveling costs and the total vehicle fixed costs incurred when selecting vehicles for distribution tasks. Effectiveness is quantified by the extent of total shortages present within the network, with a higher total shortage indicating a less effective system. The total shortage value corresponds to the sum of shortages available in all the pickup and delivery demands. Equity is defined by the equitable distribution of food, aiming to minimize the maximum shortage among all nodes in the network. In a system designed for greater equity, the objective is to minimize the largest unmet demand value among all delivery nodes. A network characterized by equity is one in which the maximum shortage (unmet demand) among all delivery nodes is kept to a minimum. Equity is achieved through a min-max approach, leveraging methodologies outlined by Sakiani et al. (2021). This study explores several exact approaches to solve the multi-objective problem. Using these exact methods, a small illustrative problem is solved through the state-of-the-art solver Gurobi. Additionally, a case study focusing on Indian food banks is presented, with the problem addressed using the Non-Dominated Sorting Genetic Algorithm II (NSGA-II).

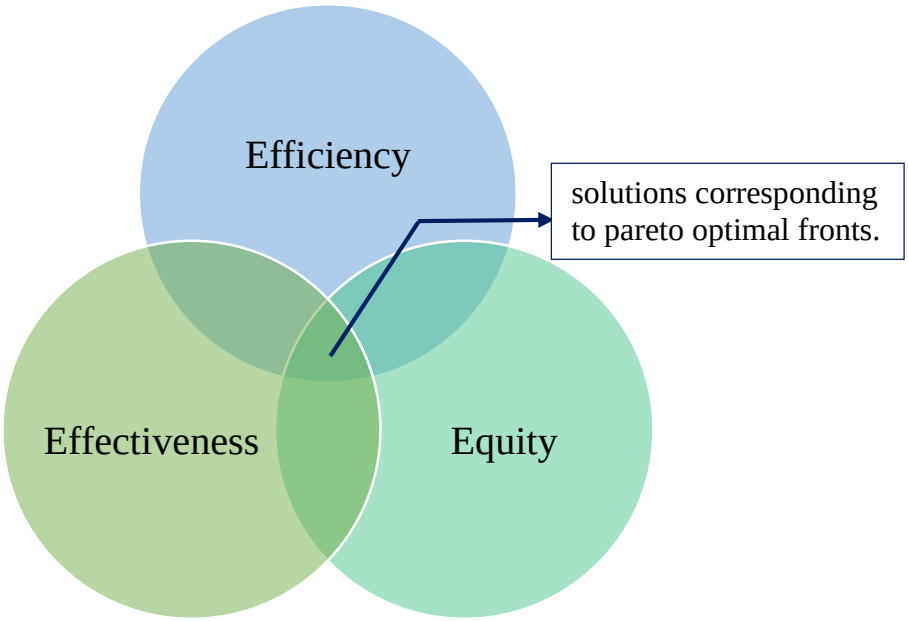


Figure 4.8: Multiple objectives of food banks

Taking into consideration the fact that there is significant scope to study MDVRP problems for food banks especially in MO framework, this study proposes following research objectives –

- Objective 1: To tackle the operational problem faced by food banks in terms of collection and distribution of surplus/leftover food.
- Objective 2: To tackle the issue of perishability of the donated food in food banks.
- Objective 3: To observe the trade-offs between different objectives of food bank operations (efficiency, effectiveness and equity)

Objective 4: To assess the performance of the developed MO mathematical model and proposed metaheuristic model against the case problem of an Indian food bank.

4.2.2. Model Formulation

The proposed Multi-Objective (MO) model represents a significant advancement in addressing the core objectives of food banks: efficiency, effectiveness, and equity. These objectives, quantified in terms of total transportation cost, total shortage (unmet demand), and equitable distribution among beneficiaries, respectively, serve as key performance indicators for the operational success of food banks. The MO model introduces a comprehensive framework capable of assessing the inherent trade-offs among these objectives, providing valuable insights applicable across diverse food bank scenarios. Given the non-profit nature of food banks, coupled with their inherent constraints of limited resources and manpower, operational planning becomes crucial for daily collection and distribution operations. The logistical challenges of efficiently catering to as much demand as possible under these constraints are modeled effectively as a Vehicle Routing Problem (VRP) with pickup-delivery. Considering the perishable nature of some donated items, particularly cooked food, a time window variant of VRP, as established in prior work presented in the Chapter 4.1, emerges as the most accurate representation for this study. To

further optimize the distribution process, the paper considers the nuanced case of splitting pickup and delivery demands, a practical consideration driven by the limited supply of donated food and constrained transportation capacities. In contrast to classical VRP scenarios where depots act as central warehouses (Birasnav et al., 2020; Bowden & Ragsdale, 2020), the unique context of food banks designates volunteer locations as the depots. This approach aligns with the decentralized nature of food bank operations, particularly acknowledging the existence of multiple volunteer hub locations. The adoption of a Multi-Depot variant for vehicle routing problems is deemed more fitting for food banks, reflecting the diverse volunteer network within their operational framework.

In this section, we propose a MO model corresponding to efficiency, effectiveness, and equity for MDVRP-TW-SP-SD problem as shown in Figure 4.8.

An MDVRP-TW-SP-SD is defined on a multi-graph with $G = (N, S, A)$ where N, S and A are vertex sets for customers, depots and arcs respectively. The vertex sets for customers, N consists of pickup nodes, P and delivery nodes, D *i.e.*, $N = P \cup D$. The total nodes in the network is represented by set $V = N \cup S$. Each node in set N have three parameters – demand ($\pm\theta_i$), service time (τ_i), time window ($[a_i, b_i]$). Demand of ay node in N can be given as- $Demand(i) = \begin{cases} \theta_i > 0, & i \in P \\ \theta_i < 0, & i \in D \end{cases}$. Each (i, j) pair in arc set has two associated parameters- travelling time, T_{ij} and cost of traversing path, C_{ij}. A uniform fleet of vehicles of size $|K| = m$ with capacity C_{ks} is present at each depot S. Additionally, splitting of demand (pickup/demand) is allowed in the network which triggers multiple visits to some node to minimize shortage. Using the above problem description, the approach is to devise optimal routes for food banks which ensures minimum travelling cost, minimum vehicle fixed cost, minimum shortage in demand as well as ensure equitable distribution. Using the same notation given in Chapter 4.1, we formulate the current problem as a MO mixed-

integer linear programming problem (MILP). The sets, indices, and parameters used in the model is given in Table 4.1.

4.2.2.1. Decision variables

In addition to the decision variables given in section 4.1.2.3, following auxiliary variables are needed for ensuring efficiency, effectiveness, and equity.

$\lambda =$ maximum shortage in the network

γ = variable needed for augmented weighted Tchebycheff method

S_2, S_3= variable needed for augmented $\in$- constraint method

4.2.2.2. Formulation

In addition to the constraints given in section 4.1.2.4., we need additional constraint for minimizing the maximum shortage. The value of shortage (unmet demand) for each node in the network should proportionate to its demand. We are not considering a perfect equitable case for this model. Sengul Orgut et al. (2016) pointed out that significant increment could be observed in total distribution by allowing smaller deviations from a perfectly equitable distribution. However, in the proposed work we utilize the min-max approach to ensure the equitability. In this approach we try to minimize the maximum shortage value that could arise on the delivery nodes of the network. Constraint (32) ensures that the variable λ takes the largest shortage value among all the delivery nodes and we put minimization of λ as one of objectives as given in equation (34).

$$\lambda \geq \sigma_j \qquad \forall j \in D \qquad (32)$$

4.2.2.3. Objectives

In this study, the trade-offs between three objective functions are analysed. These three objective functions correspond to efficiency, effectiveness and equity. Food banks are not-for-profit organizations having limited budget and therefore the total cost incurred in

collection and distribution drives should be minimal to sustain the daily activities of food banks economically. Equation (33) corresponds to the efficiency of objective function. The first term in equation (33) deals with total transportation cost and second terms deals with the fixed cost of selecting/hiring/renting a vehicle. The second essential objective of food banks are to minimize food wastage by minimizing the shortages (unmet demand) in the system. Equation (34) corresponds to the objective function which ensures effectiveness in the proposed model. Equation (34) describes the sum of unmet demand of pickup and delivery nodes respectively. Food banks are not-for-profit organizations and fair allocation of food to its beneficiaries are necessary to ensure envy-free distribution of food. Equation (35) takes care of the equity within all delivery nodes in the system by minimizing the maximum shortage among all the delivery nodes in the system.

$$\min Z_1 = \sum_{s\in S}\sum_{k\in K}\sum_{(i,j)\in A} C_{ij}x_{ijks} + \sum_{s\in S}\sum_{k\in K} f_{ks} * w_{ks} \quad (33)$$

$$\min Z_2 = \sum_{i\in P} \pi_i + \sum_{j\in D} \sigma_j \quad (34)$$

$$\min Z_3 = \lambda \quad (35)$$

4.2.3. Methodology

In this section, we endeavor several exact and metaheuristics needed to solve the MO multi-depot VRP with split pickup, split delivery and time windows. It is possible to define any MO optimization problem as a minimization problem without losing generality using the general notation from literature (Collette & Siarry, 2004).

$$\boldsymbol{min}\ \boldsymbol{f}(\boldsymbol{x}) = \begin{cases} \boldsymbol{f_1}(\boldsymbol{x}), \\ \boldsymbol{f_2}(\boldsymbol{x}), \\ \quad : \\ \quad : \\ \boldsymbol{f_n}(\boldsymbol{x}) \end{cases} \quad (36)$$

Subject to;

$$\boldsymbol{g_i}(\boldsymbol{x}) \leq 0 \qquad \forall i = 1,2,\dots,p \quad (37)$$

$$\boldsymbol{h_j}(\boldsymbol{x}) \leq 0 \qquad \forall j = 1,2,\dots,q \quad (38)$$

where $x \in \chi$ is the solution vector and the solution space of the problem having n objective functions is given by χ. A solution $\hat{x} \in \chi$ is said to be pareto optimal in the solution space, if there exists no $x \in \chi$ such that $f_k(x) \leq f_k(\hat{x})$ for $k = \{1,2, \dots, p\}$ and $f_i(x) < f_i(\hat{x})$ for some $i = \{1,2, \dots, p\}$. In other words, pareto optimal solution (non-dominated solution) is impossible to improve one objective function without compromising another. The trade-offs between conflicting objectives can be visualized from pareto-optimal front which consist of all non-dominated solutions in the objective space. There are several scalarization methods adopted in literature to solve MO problems are discussed in the section below.

4.2.3.1. Exact Methods

In this study we are going to endeavor three scalarization methods-

- Weighted sum (Miettinen, 1998),
- Augmented ∈-constraint (Mavrotas & Florios, 2013) and,
- Augmented weighted Tchebycheff (Nurjanni et al., 2017; Steuer, 1989).

In these scalarization techniques MO is converted into single objective and some associated parameters. Weighted sum (WS) is a well-known method in which weights are assigned to different objective functions and then a composite function is created and treated as a single-objective function as given in equation (39).

Weighted-sum method

$$\min w_1 Z_1 + w_2 Z_2 + w_3 Z_3 \quad (39)$$

Subject to;

Constraint (2) – (32)

The assigned weights follows $w_1 \geq 0$, $w_2 \geq 0$ and, $w_3 \geq 0$ such that $w_1 + w_2 + w_3 = 1$ and Z_1, Z_2 and, Z_3 are the three objectives of the proposed problem. The solutions obtained

using WS method are weakly pareto optimal and on the convex Pareto front, the algorithm can only give extreme solutions.

The idea behind ∈-constraint method is that, at a time one objective is treated as main objective function and rest of the objectives of the model are transformed into constraints. In Augmented ∈-constraint method, non-extreme solutions can be obtained. The equations for Augmented ∈-constraint method is given below-

Augmented ∈-constraint method (Augmecon)

$$\min \; Z_1 + \delta(S_2/R_2 + S_3/R_3) \tag{40}$$

Subject to;

$$Z_k + S_k = e_k \qquad \forall k = \{2,3\} \tag{41}$$

Constraint (2) – (32)

where $\delta \in [10^{-3}, 10^{-6}]$ and S_2 and S_3 are each constraints' surplus variable. The values R_2 and R_3 are the ranges of the objective functions Z_2 and Z_3. The values e_2 and e_3 are the RHS parameters drawn from grid points for the specific iteration of the objective functions Z_2 and Z_3.

The weakly pareto optimal solution can be avoided using the augmented weighted Tchebycheff approach as given below-

Augmented weighted Tchebycheff method

$$\min \; \gamma + \rho * \sum_{k=1}^{3} (Z_k - z_k^{**}) \tag{42}$$

Subject to;

$$w_k(Z_k - z_k^{**}) \leq \gamma \qquad \forall k \in \{1,2,3\} \tag{43}$$

Constraint (2) – (32)

The scalar $\rho > 0$ is a small number. The values z_k^{**} represents the best value of the objective function Z_k for $k \in \{1,2,3\}$. The description of all other notations in augmented weighted Tchebycheff is similar to weighted sum method.

4.2.3.2. Metaheuristics

In this section, a well-documented meta-heuristic is modified according to the proposed problem to solve the larger instances of the problem efficiently. Due to wide applicability, efficient methodology and high diversity in terms of variety in problem domains, NSGA-II is selected for solving the MO-MDVRP-TW-SP-SD problem. The procedure starts with initializing the population and performing the genetic-operators like mutation and crossover to generate a new population to ensure elitism. The typical process for NSGA-II algorithm is shown in Figure 4.9.

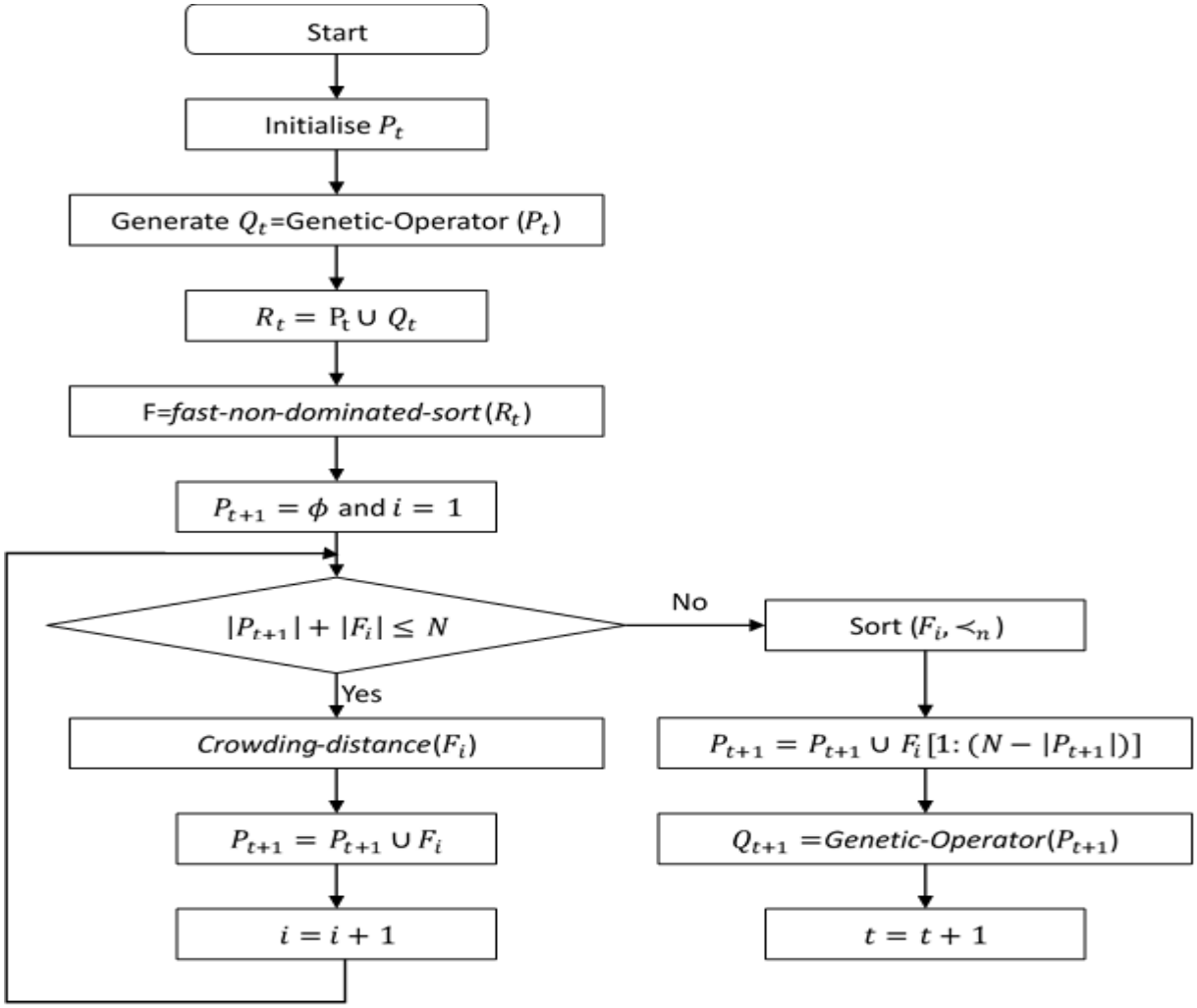


Figure 4.9: Flowchart for NSGA-II algorithm

The offspring for the first generation (Q_t) of the population is obtained using binary tournament selection, crossover and mutation operator on parent population (P_t). After the first generation, the combined population R_t is subjected to fast-non-dominated-sort (Deb et al., 2000). The non-dominated solutions are arranged in different fronts given by composite set, $F = f_{1,}, f_2, \ldots f_{last}$. f_1 is the best non-dominated set, f_2 is the second best non-dominated set and so on. After this, crowding distance $I(d_j)$ is obtained for all the individuals j of each front f_i using following Algorithm 3.

Algorithm 3: Crowding Distance (F_i)

Input: set $\boldsymbol{f_i}$

Output: crowding distance $\boldsymbol{I(d_j)}$ for each individual $\boldsymbol{j}$ in the input front

Initialize: $\boldsymbol{I(d_j) = 0}$ for each individual $\boldsymbol{j}$ in the input front;$\boldsymbol{|f_i| = n}$

for each objective function **z**

  $\boldsymbol{I = sort(f_i, z)}$

  $\boldsymbol{I(d_1) = \infty}$ and $\boldsymbol{I(n) = \infty}$

  for $\boldsymbol{j = 2}$ to $\boldsymbol{n - 1}$

    $$\boldsymbol{I(d_j) := I(d_j) + \frac{I(d_{j+1}).z - I(d_{j-1}).z}{z_{max} - z_{min}}}$$

  end for

end for

The term $I(d_j).z$ refers to the value of objective function z for j[th] individual of front f_i.We continue to add the individuals to the new population P_{t+1} according to their non-domination rank and crowding distance until the size of P_{t+1} reaches *N*. The generation counter is updated at the end and this whole procedure runs iteratively for the given generation size.

4.2.4. Computational Experiments

In this section, we present the outcomes derived from employing the three scalarization methods elucidated earlier, accompanied by an illustrative example. Furthermore, we present the results of given conflicting objectives within a case study involving the Robin Hood Army. Our computational experiments conducted to extract the following insightful analyses:

- To scrutinize the intricate trade-off behavior inherent in multiple objectives, specifically addressing efficiency, effectiveness, and equity.
- To evaluate the efficacy of diverse scalarization techniques in resolving the stipulated multi-objective problem, leveraging a smaller instance for a comprehensive understanding.
- To assess the performance of the Non-dominated Sorting Genetic Algorithm II (NSGA-II) when applied to larger networks, specifically in the context of the Multi-Objective Multi-Depot Vehicle Routing Problem with Split Pickup and Delivery (MO-MD-VRP-SP-SD).
- To undertake a comprehensive validation of the proposed model tailored for the Multi-Objective Multi-Depot Vehicle Routing Problem with Time Windows and Split Pickup and Delivery (MO-MD-VRP-TW-SP-SD).

These computational experiments are designed to yield invaluable insights into the nuanced behavior and effectiveness of the proposed methodologies across a spectrum of scenarios, contributing to the robustness and applicability of the developed model.

The computational experiment is structured into two distinct categories, each contributing to a comprehensive analysis of the proposed methodologies Firstly, we address the MO-MD-VRP-TW-SP-S) for a smaller instance. The specification of this instance are outlined in Table 4.15, derived through modifications to the first instance of the benchmark dataset

originally presented by Cordeau et al. (2001) for MD-VRP-TW problems. Secondly, we embark on the development of a case study centered around the operations of the Robin Hood Army and utilising the Non-dominated Sorting Genetic Algorithm II (NSGA-II) for its solution. The data integral to this case study is sourced from the contributions of Chapter 4.1, ensuring a robust and relevant foundation for the experimental analysis. This dual-pronged approach ensures a holistic investigation, combining instances for the exploration with a real-world case study, thereby enhancing the applicability and robustness of the proposed methodologies.

Table 4.15: Description of the smaller size problem

Nodes	Demand	Service Time	x-coordinate	y-coordinate
depo1	0	0	4.163	13.559
depo2	0	0	21.387	17.105
depo3	0	0	-36.118	49.097
depo4	0	0	-31.201	0.235
1	-25	2	-29.73	64.136
2	-11	7	-30.664	5.463
3	100	21	51.642	5.469
4	-45	24	-13.171	69.336
5	-23	1	-67.413	68.323
6	-56	17	48.907	6.274
7	-5	6	5.243	22.26
8	21	5	-65.002	77.234
9	-67	7	-4.175	-1.569
10	27	1	23.029	11.639

The network of smaller size problem consists of 14 nodes, out of which, 4 nodes are depots, 3 nodes are pickup (highlighted in bold in Table 2), and rest are delivery nodes. For ease of experimentation, we have taken a common time window for each of these nodes as [0, 1000]. Each depot has a homogenous fleet of vehicles of size 4 and capacity 200 units. The fixed cost of hiring/renting a vehicle, f_{ks} is taken as 500 units. The traversal time for each node-pair $(i, j) \in A$ is generated randomly between [1,10]. The cost of traversal between two nodes is taken as the Euclidean distance between the two nodes.

For the case study, the size of the overall network is 57 out of which there are 3 depot nodes, 25 pickup nodes and 29 delivery nodes. There are 5 vehicles at each depots of capacity 200 units. The fixed cost in this case is taken as 500 units which is motivated from the actual cost of renting a vehicle. The cost of traversal between two nodes is obtained by computing the real-time distances between the locations using geopy module of python and BING Api. The time-window here corresponds to the time within which the recovered/surplus food is distributed such that the perishability of the food can be avoided. The time window for delivering the food is taken as 4.5 hours.

We have conducted all the experimentation on the compute nodes of Param Shivay supercomputer facility. There are 212 compute nodes and each compute node have 384 GB RAM. Param Shivay supercomputer has 2*Intel Xeon SKL G-6148 processor having memory speed of 25GB/sec and processing speed of 2.4 GHz. State-of-the-art-solver Gurobi has been used to solve the MO problem using weighted sum, Augmecon, and Tchebycheff method. DEAP framework has been utilised for developing NSGA-II algorithm for solving the case problem.

4.2.4.1. Result of weighted sum method on smaller size network

For weighted sum methods 3 sets of weights (w_1, w_2, w_3) of 21 elements each are generated such that $w_1^i + w_2^j + w_3^k = 1$ and $i = j = k$ are the indices of the elements of the set w_1, w_2, w_3 respectively, as given in Table 4.16. These 3 sets of weights can be used in 6 ways $\{(w_1^i, w_2^j, w_3^k) : i = j = k \ \& \ (i, j, k)$ are the indices of $(w_1, w_2, w_3)\}$ for the given objectives. So, the overall MO model is run for 21*6 = 126 times for all the 126 combinations of weights.

Table 4.16: Different weights used for weighted sum method

$\boldsymbol{w_1}$	0.01	0.02	0.03	0.04	0.05	0.06	0.07	0.08	0.1	0.1	0.11
	0.12	0.13	0.14	0.15	0.16	0.17	0.18	0.19	0.2	0.45	

w_2	0.98	0.96	0.95	0.92	0.9	0.88	0.86	0.84	0.82	0.8	0.85
	0.61	0.64	0.67	0.7	0.73	0.76	0.79	0.48	0.52	0.3	
w_3	0.02	0.03	0.02	0.04	0.05	0.06	0.07	0.08	0.08	0.1	0.31
	0.27	0.23	0.19	0.15	0.11	0.07	0.03	0.33	0.46	0.25	

These 126 combinations are run for the proposed model by giving a time limit of 1 hour for each run on Gurobi. Out of the 126 results, only 6 non-dominated solutions can be obtained i.e. only 5% distinct solutions are obtained using weighted sum approach. The values of all three objective functions as well as their optimality gap are reported in Table 4.17. The column weight combination of Table 4.17 represents the order of weights used for the multiple objectives. For example, weight321_1 refers to the first elements of the sets w_3,w_2, and w_1, respectively.

Table 4.17: Non-dominated solution using weighted sum approach

Solution no	**Run no.**	**Weight combination**	Z_1	Z_2	Z_3	**Optimality gap (%)**
1	2	weight123_1	812.59	84	13.6	0
2	17	weight123_16	806.1	84	25	0
3	18	weight123_17	791.57	84	67	0
4	19	weight123_18	634.06	126	45	0
5	21	weight123_20	0	380	67	0
6	107	weight321_1	798.13	84	45	0

The first, second and third objective corresponds to efficiency (Z_1), effectiveness (Z_2), and equitable distribution (Z_3). As we can see from Table 4.17, the extreme solution is [0,380,67] and [812.59, 84, 13.6]. In the first extreme solution [0,380,67], all the pickup and delivery demands are treated as shortages and therefore cost incurred is 0. In this case solution obtained is most efficient. However, this solution is worst in terms of effectiveness. The second extreme solution [812.59,84,13.6], where we get most effective and equitable solution, however we receive worst solution in terms of efficiency.

4.2.4.2. Results of Augmented ∈-constraint method on smaller size problem

For Augmecon method, firstly a payoff table has been developed by using the traditional lexicographic approach to solve the MO problems (Cococcioni et al., 2020; Veeramani & Duraisamy, 2012; Zhong et al., 2022). From the payoff table given in Table 4.18, we calculated the ranges, R_2 and R_3 of Objective 2 and Objective 3 as 296 and 53.4 and these ranges are divides into 9 equal intervals using 32.89 and 5.93 as steps for generating values of e_2 and e_3 in Table 4.19.

Table 4.18: Payoff table

	$\boldsymbol{Z_1}$	$\boldsymbol{Z_2}$	$\boldsymbol{Z_3}$
min $\boldsymbol{Z_1}$	0	380	67
min $\boldsymbol{Z_2}$	791.5643127	84	13.6
min $\boldsymbol{Z_3}$	812.5835379	84	13.6

As we can see from equation (41), there are 10 values for changing the parameter e_k for each objective function $k \in \{2,3\}$. So, there will be a total of 100 runs (10*10). The values of e_k for each objective function $k \in \{2,3\}$ and for each run is given in Table 4.19.

Table 4.19: Value of parameters e_2 and e_3 for Augmecon method

Grid	0	1	2	3	4	5	6	7	8	9
e_2	84	116.89	149.78	182.67	215.56	248.45	281.34	314.23	347.12	380
e_3	13.6	19.54	25.47	31.4	37.34	43.27	49.2	55.14	61.07	67

In Augmecon method, out of 100 runs we got 19 distinct non-dominated solutions i.e., 19% distinct non-dominated solutions have been obtained using Augmecon method. The number of run, grid specification, values of all three objective functions and optimality gap of the obtained solution is given in Table 4.20. The grid specification corresponds to the interval value of parameters e_2 and e_3 taken in the constraint. For example, grid0_2 corresponds to the grid 0 and 2 for the values e_2 and e_3 respectively.

Table 4.20: Non-dominated solution using Augmecon method

Solution no.	**Run no.**	**Grid specification**	Z_1	Z_2	Z_3	**Optimality gap (%)**
1	1	grid0_0	812.59	84	13.6	0
2	3	grid0_2	806.1	84	25	0
3	7	grid0_6	798.13	84	45	0
4	10	grid0_9	791.57	84	67	0
5	13	grid1_2	803.17	92	25	0
6	17	grid1_6	797.68	92	45	0
7	22	grid2_1	803.65	126	17.8	0
8	23	grid2_2	725.59	126	25	0
9	24	grid2_3	725.35	126	29	0
10	27	grid2_6	622.02	134	45	0
11	33	grid3_2	723.03	180	25	0
12	57	grid5_6	620.59	180.01	45	0
13	59	grid5_8	620.34	246	56	0
14	70	grid6_9	564.85	268	67	0
15	87	grid8_6	616.33	326	48	0
16	89	grid8_8	566.48	326	56	0
17	90	grid8_9	561.71	326.01	67	0
18	98	grid9_7	616.33	326.01	48	23.1
19	100	grid9_9	0	380	67	0

Augmecon provides some unique non-dominated solutions along with all the solution obtained by weighted sum method. Solution no. 1 to 4 have the same value for Z_2. As we can see from solution 1 and 2, Z_1 reduces by 0.8% and Z_3 increases by 83.8%. Similarly, from solution no. 2 & 3, it can be seen that Z_1 reduces by 0.9% and Z_3 increases by 80%. It can be noted that a very small increment in efficiency (minimization of total cost) leads to a significant decrease in equitability (max shortage).

4.2.4.3. Result of Augmented weighted Tchebycheff method on smaller size problem

The combination of weights in the augmented weighted Tchebycheff method is same as that of weighted sum method as given in Table 4.16. The number of runs in both methods are same. The constraint (43) for the augmented weighted Tchebycheff method uses the best values z_k^{**} for $k \in \{1,2,3\}$ from the payoff table given in Table 4.18. The value of ρ used in the equation (42) is taken as 0.001 in all the 126 runs. Out of 126 runs, 19 distinct non-dominated solutions are obtained as shown in Table 4.21. Augmented weighted Tchebycheff performs better than the weighted sum methods in terms of ability to produce distinct non-dominated solution. Using this method, 15% non-dominated distinct solutions has been obtained. The specification of weight combination column is same as that of given in section 4.2.4.2.

Table 4.21: Non-dominated solution using Augmented weighted Tchebycheff method

Solution no.	**Run no.**	**Weight combination**	Z_1	Z_2	Z_3	**Optimality gap (%)**
1	**1**	weight123_0	791.5643127	84	67	0
2	**4**	weight123_3	787.4470391	118	67	0
3	**5**	weight123_4	634.0569785	126	45	0
4	**7**	weight123_6	622.0170589	134	45	0
5	**11**	weight123_10	620.5812459	180	45	0
6	**19**	weight123_18	564.841128	268	67	0
7	**20**	weight123_19	797.6794421	92	45	0
8	**21**	weight123_20	0	380	67	0
9	**22**	weight132_0	803.6416188	126	17.8	0
10	**24**	weight132_2	723.0266744	180	25	0
11	**25**	weight132_3	622.0170589	134	45	25.7
12	**26**	weight132_4	620.5812459	180	45	17.7
13	**27**	weight132_5	616.3236533	326	48	22.3
14	**28**	weight132_6	566.4779355	326	56	0

15	**41**	weight132_19	803.1606866	92	25	0
16	**66**	weight231_2	725.589206	126	25	0
17	**116**	weight321_10	543.3702357	370	67	0
18	**117**	weight321_11	561.710254	326	67	0
19	**119**	weight321_13	598.7118739	258	67	0

In the augmented weighted Tchebycheff method, some runs could not produce optimal solutions and a very large optimality gap is obtained. The solution no. 11, 12, and 13 are the best bound obtained with time limit of 1 hour and optimality gap of 25.7%, 17.7%, and 22.3%, respectively. Several intermediate solutions are obtained using this method. However, the behavior persists similar to one explained in Augmecon method. To reduce the total cost by a small degree, the value of maximum shortage in the system increases significantly. To better visualize the different non-dominated solutions obtained from exact methods, pareto solutions are plotted in a pair-wise function for each objective functions. As, it can be seen from Figure 4.10, very few solutions are obtained using weighted sum method and high trade-off can be observed between total cost and shortage.

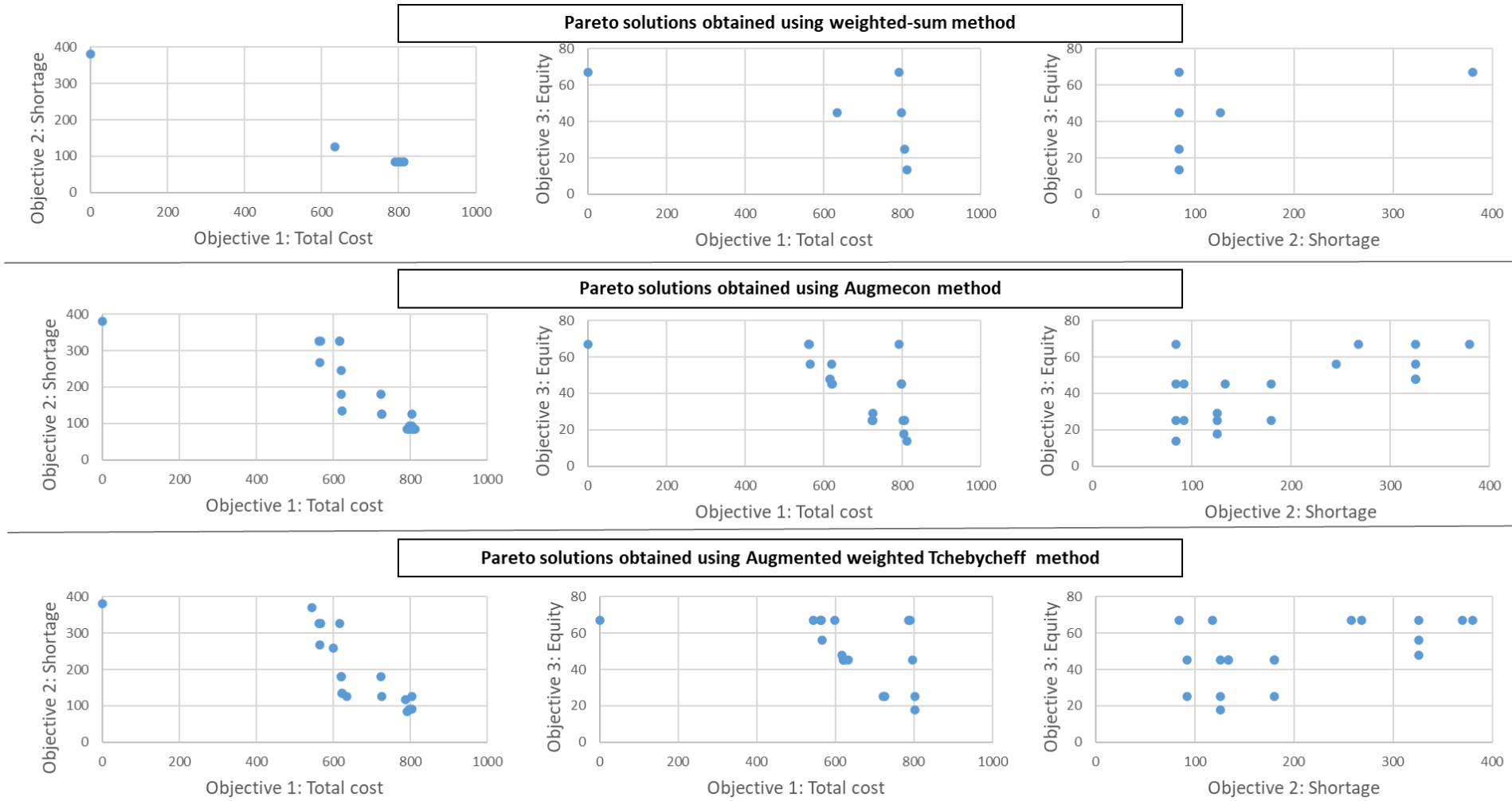


Figure 4.10: Non-dominated solutions obtained by exact method

4.2.4.4. Results of NSGA-II on case study

NSGA-II methodology has been adopted to solve the case problem. The multiple objectives corresponding to equation (33) – (35) in the case problem. For the NSGA-II algorithm population size is taken as 200, maximum number of generations are 500, crossover and mutation probabilities as 0.7 and 0.25 respectively. The CPU time to solve the problem of 54 nodes and 3 depots using NSGA-II algorithm is 76.2 seconds. The average number of vehicles used ranges from 2 to 5. The pareto front for pair-wise objectives obtained using NSGA-II algorithm is shown in Figure 4.11.

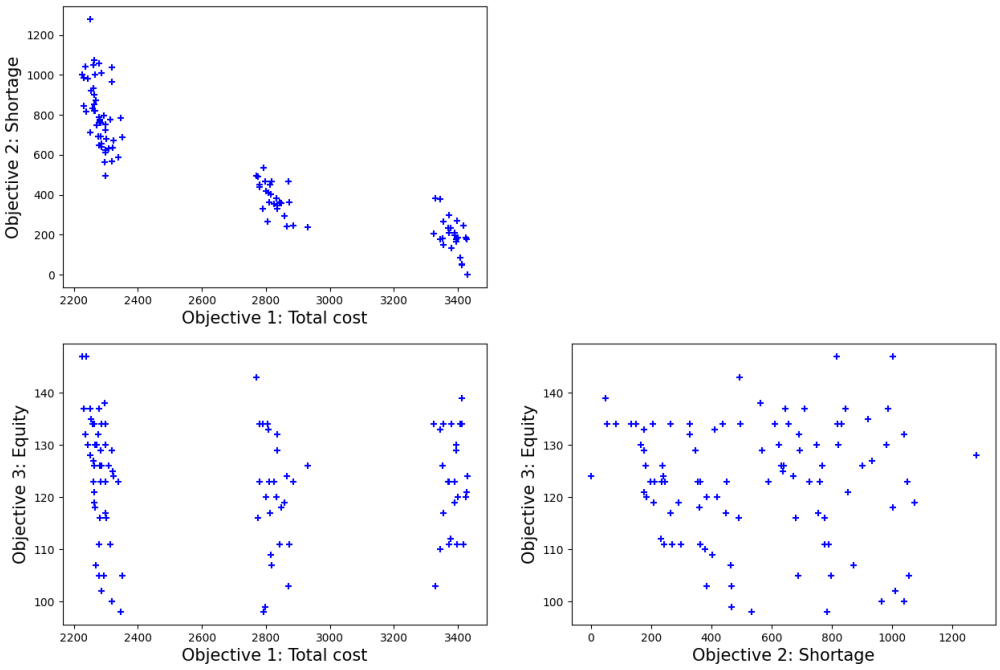


Figure 4.11: Pareto-front obtained for the multi-objective model

The first sub-plot of Figure 4.11 specifies the relation between objective-1 (Efficiency-Total transportation cost) and objective-2 (Effectiveness- Total shortage). A trade-off behavior is

evident in the first subplot in which more effective system incurs higher transportation cost while a lesser effective system incurs lesser transportation cost. There could be several reasons for this type of behavior. A more effective system tries to minimize the unmet demands of all the nodes (pickup and delivery). Therefore, it requires more vehicles to fulfill the demand and hence higher vehicle costs are incurred. Also, an increase in splitting the demands of the nodes in the network could be a prominent reason for increase in the travelling cost. The second sub-plot in Figure 4.11 shows pareto front of objective-1 (Efficiency-Total transportation cost) and objective-3 (Equity-maximum unmet demand among all the delivery nodes). Although clear trade-off is not evident from this subplot but as system tries to decrease the total cost the performance of objective-3 degrades with some minor fluctuations. In the third sub-plot, pareto fronts are plotted for objective-2 (Effectiveness- Total shortage) and objective-3 (Equity-maximum unmet demand among all the delivery nodes).

As it can be observed from Figure 4.11, there exists three clusters of non-dominated solutions for objective-1 and objective-2. However, their respective behaviour with increase and decrease can not be directly shown. To better analyse the non-dominated solutions, the pareto efficient solutions for all the three objectives are plotted on a 3-D graph in Figure 4.12.

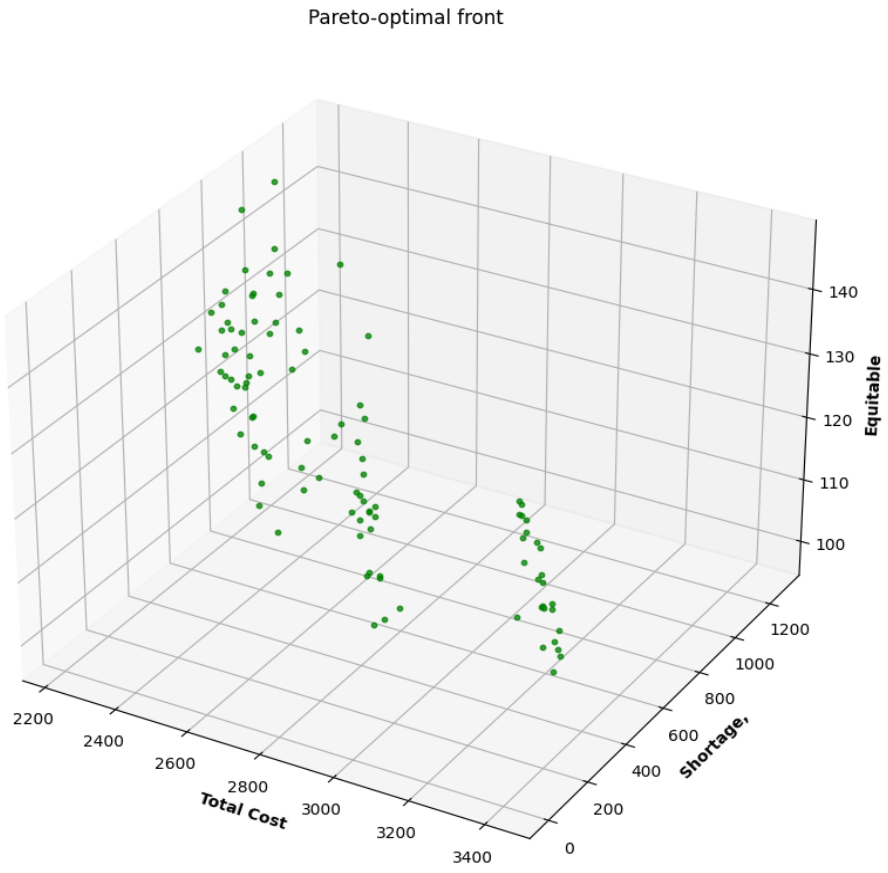


Figure 4.12: Pareto efficient solution in 3-D space

The value of total shortage increases when the model tries to minimize the maximum unmet demand among all the beneficiary nodes. In order to analyze the trade-off behavior of the different objective function, a heat map is plotted for all the non-dominated solutions according to their correlation values as shown in Figure 4.13.

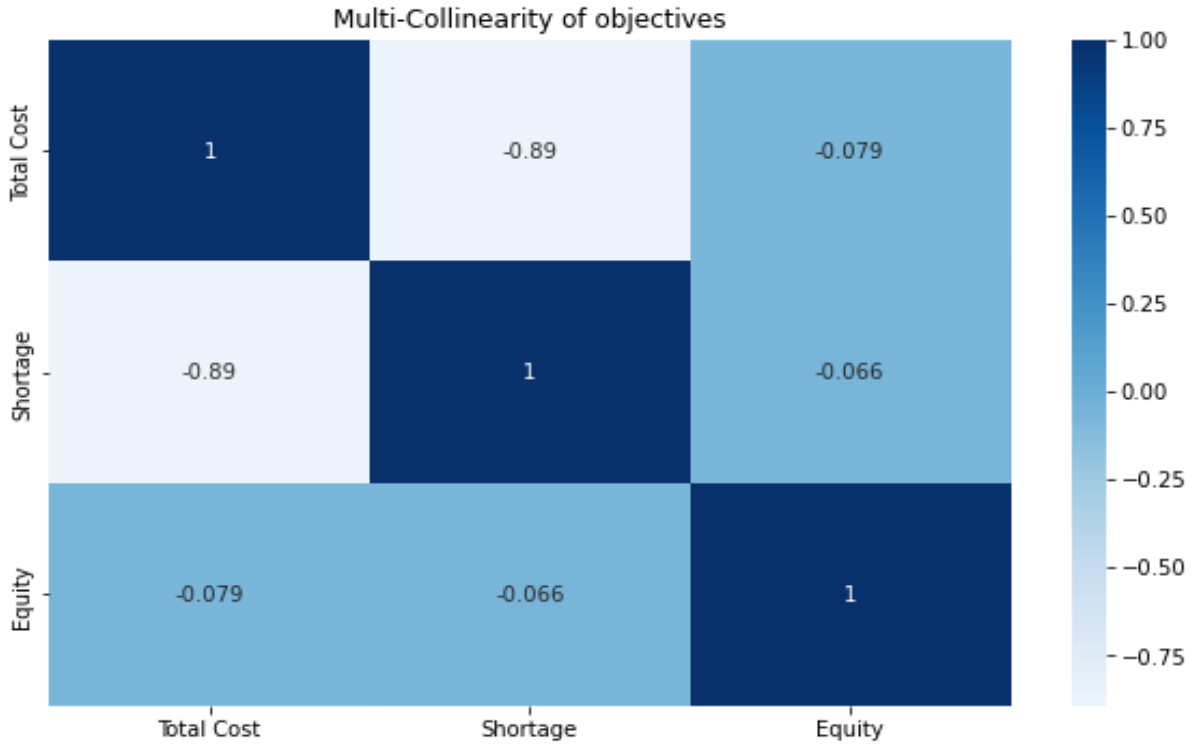


Figure 4.13: Heat-map for correlation among different objectives

Positive correlation refers to linear relation between the two variables under study and negative correlation refers to inverse linear relation between the two variables under study. There is high negative correlation between total cost and total shortage (-0.89). It means, as total shortage decreases (Effectiveness increases), the value of total cost increases (Efficiency decreases) to nearly same extent in the system. However, there is relatively lesser negative correlation between total cost and equity (-0.079). It means as system tries to minimize the maximum shortage among all delivery node (Equity increases), the total shortage (Effectiveness decreases) deteriorates. This deterioration is relatively less as compared to that of total cost and equity. Similarly, negative correlation exists for effectiveness and equity (-0.066).

4.2.5. Conclusion

This paper undertakes endeavors multi-objective modeling framework for surplus food recovery and redistribution in India, thereby enriching the single-objective traditional Vehicle Routing Problem (VRP) generally used to solve that problem. Extending the groundwork laid in the Chapter 4.1, this study endeavors to address the limitation of our earlier study and gap in the existing literature by introducing equity as one of the objective functions. The incorporation of equity as a measure contributes significantly to the promotion of fair distribution practices within food banks, employing a min-max approach to ensure even-handed allocation of donated food among beneficiaries. This research not only fills an important void in the existing body of literature but also makes a noteworthy contribution to the field of food banks by delving into the context of multi-objective optimization. The proposed model, formulated as a multi-objective mixed-integer linear programming problem, serves as a practical and comprehensive solution to the daily operational challenges faced by Indian food banks. Moreover, the study advances the solution methodology by introducing a modified Non-dominated Sorting Genetic Algorithm II (NSGA-II) specifically tailored to handle large-scale problems, thus enhancing the scalability and efficiency of the proposed approach. This collective contribution positions the research as a pivotal advancement in the domain of food bank operations. This study endeavors to identify a set of non-dominated solutions for multiple objectives related to efficiency, effectiveness, and equity. The efficiency of the proposed problem is gauged through the total transportation cost. Effectiveness is measured by the total shortage among all nodes in the network, while the maximum shortage value (unmet demand) among all delivery nodes is considered a metric for equity within the system. The research introduces a mixed-integer linear programming model to formulate the

operational challenges faced by Indian food banks as a Vehicle Routing Problem (VRP), enriched with real-time intricacies such as time windows, split demand, and multiple depots, all within a multi-objective framework. To the best of our knowledge, this is the first study that considers Indian food banks in a multi-objective scenario for the daily operational problem of collection and distribution. Computational experiments are conducted across two categories of the problem: a smaller size network, exemplified through an illustrative example, and a larger size network, as demonstrated in the case study. The smaller network comprises 10 nodes for customers and 4 nodes for depots, while the larger network encompasses 54 nodes for customers and 3 nodes for depots, providing a comprehensive assessment of the proposed model's applicability and scalability.

Exact method for solving the Multi-Objective (MO) problem, such as scalarization techniques weighted sum method, augmented $\in$-constraint method, and augmented weighted Tchebycheff method, have been deployed to address to solve the problem consisting of smaller network. The outcomes demonstrate that the weighted sum method, augmented $\in$-constraint method, and augmented weighted Tchebycheff method yield 5%, 19%, and 15% distinct non-dominated solutions, respectively. It is noteworthy that even with extensive computational capabilities of the Param Shivay Supercomputer, the exact methods failed to provide optimal solutions for specific combinations within a one-hour runtime. Consequently, to tackle the intricacies of a larger-sized network, the Non-Dominated Sorting Genetic Algorithm II (NSGA-II) is proposed. NSGA-II remarkably produces near-optimal solutions within 76 seconds for the larger network. The results outline a noteworthy cost incurred in ensuring both effectiveness and equity within the system. Future enrichments to the proposed problem could involve more intricate computations, and a comprehensive performance analysis of alternative

metaheuristics for solving the proposed MO model could be conducted. This study operates under the assumption of the deterministic nature of supply donation and demand. However, in reality, donations and demand exist with uncertainty, a facet not considered in the present study. Recognizing this limitation, future research endeavors could focus on developing a robust model that encompasses the uncertain behavior of supply donation and demand, thereby addressing a crucial research gap in this domain.