COMPARING METAHEURISTIC ALGORITHMS FOR SOLVING CROWDSHIPPING PROBLEMS

YÜKSEL MERT CANKUŞ

Master's Thesis

Graduate School
Izmir University of Economics
Izmir
2022

COMPARING METAHEURISTIC ALGORITHMS FOR SOLVING CROWDSHIPPING PROBLEMS

YÜKSEL MERT CANKUŞ

A Thesis Submitted to
The Graduate School of Izmir University of Economics
Master's Program in Computer Engineering

Izmir
2022

ABSTRACT

COMPARING METAHEURISTIC ALGORITHMS FOR SOLVING CROWDSHIPPING PROBLEMS

Cankuş, Yüksel Mert

Master's Program in Computer Engineering

Advisor: Asst. Prof. Dr. Kutluhan EROL

October, 2022

This thesis focuses on crowdsourced delivery systems and refers to its operational decision problem as a crowdshipping problem formulates as an offline optimization problem. In order to solve the crowdshipping problem, several metaheuristic algorithms and heuristic operations are proposed. An experimental setup is designed to assess the performance of proposed solution techniques. Results of conducted experiments in this thesis are presented and analyzed in a comparative manner. Results indicated that algorithms with less randomization outperform more randomized algorithms with statistical significance. Less randomized outperforming algorithms provide statistically similar results to each other.

Keywords: Crowdshipping, Crowdsourced Delivery, Last-Mile Delivery, Crowd Logistics, Metaheuristics, Sharing Economy.

ÖZET

METASEZGİSEL ALGORİTMALARIN KİTLE DESTEKLİ NAKLİYE PROBLEMİ İÇİN KARŞILAŞTIRILMASI

Cankuş, Yüksel Mert

Bilgisayar Mühendisliği Yüksek Lisans Programı

Tez Danışmanı: Dr. Öğr. Üyesi Kutluhan EROL

Ekim, 2022

Bu çalışma, kitle destekli dağıtım sistemlerine odaklanmakta ve operasyonel karar problemini bir çevrimdışı optimizasyon problemi olarak ele alıp kitle destekli nakliye problemi olarak atıfta bulunmaktadır. Kite destekli nakliye problemini çözmek için çeşitli metasezgisel algoritmalar ve sezgisel işlemler önerilmiştir. Önerilen çözüm tekniklerinin performansını değerlendirmek için bir deney düzeneği tasarlanmıştır. Bu tezde yapılan deneylerin sonuçları karşılaştırılmalı bir şekilde sunulmakta ve analiz edilmektedir. Bu çalışmalardaki sonuçlar, daha az rastgeleliğe sahip algoritmaların, istatistiksel olarak daha rastgele algoritmalardan daha iyi performans gösterdiğini göstermiştir. Daha az rastgele, daha iyi performans gösteren algoritmalar, istatistiksel olarak birbirine benzer sonuçlar vermiştir.

Anahtar Kelimeler: Kitle Destekli Nakliye, Kitle Destekli Rotalama, Son Mil Dağıtımı, Kitle Destekli Lojistik, Metasezgizeller, Paylaşım Ekonomisi.

ACKNOWLEDGEMENTS

Araştırma boyunca sabrı, yardımları ve yol göstericiliği için tez danışmanım Kutluhan Erol’a en içten teşekkürlerimi ve dileklerimi sunmak isterim. Bu süreçte bana desteklerini esirgemeyen aileme teşekkürlerimi sunarım.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1: INTRODUCTION

1.1 Motivation

Last-mile delivery and other related operational activities such as delivery operations of shipments from transportation hubs to private customer households in urban areas is an important area of research as described in Boysen et al. (2021). However, many existing last-mile delivery concepts and methodology do not address all the given macro-scale world problems that presented as follows:

- The COVID-19 pandemic and its global effects are undoubtedly important for the logistics sector as described by Raj et al. (2022).
- Inflationary effects on global prices caused by ongoing Russian invasion of Ukraine following the pandemic hindered the purchasing power of households. Current Last-Mile delivery operations do not provide an abundant source of part-time jobs opportunities.
- Climate change and related environmental issues are requiring more ecofriendly solutions in Last-Mile Delivery.

Although there are separate individual solutions related to the presented problems, they do not solve the combination of all the mentioned problems and issues in a unified frame. For example, literature covered by Ghaderi et al. (2022), aims to solve environmental problems in isolation without addressing the other issues. Even though, such approach separates a specific problem and its solution from others, addressing all mentioned issues with minimal increase of complexity is preferable to integrate several existing solution techniques with each other. Another example is that most of traditional vehicle routing literature consider vehicles & couriers as full time employees. Since described approach is highly classical, alternate constraints which enables part-time job offers or constraints that represents the acceptance criterions of couriers regarding to transportation offers are not extensively studied in the existing vehicle routing problem literature involved in current Last-Mile delivery studies. In order to solve the combination of all the described caveats of the existing literature, a crowdsourced delivery system is presented in my thesis.

Crowdsourced delivery systems, which is surveyed by Alnaggar et al. (2021), applies "crowdshipping" idea of enabling crowds to participate in Last-Mile Delivery operations for compensation. Crowdshipping problem can be described as follows. A set of payloads with different sizes are requested to be transported where each request involves a pickup location and a delivery location. Those requests can be a flower delivery for an anniversary, an urgent delivery of a document or daily preplanned delivery operations of cargos and medical equipments. Requests may be available for pickup

and/or delivery for specific time windows. Customers may or may not know the price and acceptance information of their requests but they share their price expectations they are willing to pay. Crowdsourced delivery system may charge different each customer requests with different prices which can be declined by customer if charged price is inconsistent with price expectation of each customer transporation request.

In order to satisfy the transportation requests, prospective individuals who are also called "crowdshippers" would like to participate in the pickup and delivery process of the mentioned requests in return of compensation. Parcitipating crowdshippers include a university student who would like to earn extra money on their way to campus, a taxi driver would like to earn more while returning from a successful taxi trip or even a pedestarian using public transportation can be a crowdshipper. Each crowdshipper share their origin and destination location of their original route, time availability described in two time windows for origin and destination locations described with earliest arrival and latest departure, capacity information and expected compensation per extra km for their detour. Crowdsourced delivery system sends an offer relevant with their compensation expectations to each crowdshipper. Crowdshippers can accept or reject offers which is may or may not known to crowdsourced delivery systems before offering. However, acceptance and rejection behaviour of each crowdshipper must be relevant with their provided compensation expectation. A crowdsourced delivery system aims to match given requests with crowdshippers, route and schedule pickup and delivery task of crowdshippers with pricing and compensation decisions constrained by the price and compensation expectations of both crowdshippers and customers.

Crowdsourced delivery systems enable crowdshippers to have part-time income which provides an opportunity to stabilize inflationary shocks. Crowdsourced delivery systems are also beneficial for Last-Mile delivery operations, since the COVID-19 pandemic revealed that the combined positive increase in demand with negative supply shocks stresses the capabilities of Last-Mile delivery operations. Crowdsourced delivery systems can provide more cost efficient and ecofriendly operations via enabling routes that were not available to the system.

1.2 Research Issues

In addition to the difficulty of modeling and solving existing more traditional Last-Mile Delivery concepts, crowdshipping-based problems have their own unique collected set of issues.

1.2.1 Generalization

There are already many crowdshipping-based studies in the literature, but the modeling of these studies is both partially related or unrelated to each other and very disconnected from other models that can be counted as predecessors. In this thesis, described problem require more generalized methodology in order to capture methodological aspects of both crowdshipping-based studies and existing more traditional Last-Mile delivery studies.

1.2.2 Pricing and Compensation

Unlike traditional routing problems, crowdshipping problem involves pricing and compensation decisions which are subjected to expectations of different types of customers and crowdshippers. Such pricing and compensation decisions requires and enables adjustments on supply of the crowdshippers via compensation decisions and demand of the delivery requests via pricing decisions. Adjustments done via pricing and compensation mechanisms directly affects the complexity and profits of crowdshipping problem because the crowdshipping problem is highly constrained via expectations of both crowdshippers and requests. For example expected compensation of a university professor may be higher than a university student for same exact route. A minimal adjustment on compensation decision may satisfy the compensation expectations of an university student which may not satisfy the compensation expectation an university professor. Described adjustments on compensation may or may not change the profitability of the operation, availability of crowdshippers, demand for delivery requests and complexity of operational scenarios.

1.2.3 Heterogeneous and Asymmetric Constraints

Proceeding with pricing and compensation decisions, crowdshipping-based problems have to address several types of decisions and constraints. Such heterogeneity of decisions and constraints are impactful on many different aspects of crowdshipping-based problems. For example:

- University students might like to visit the same university as a destination point at different times.
- Some crowdshippers may not have sufficient capacity for all packets with close proximity and profitability.
- Different crowdshippers may start at same origin location with different destination locations.
- University students may have different compensation expectations than university

professors.

All the problems mentioned and not limited by this list implies that even a minor unsatisfied constraint affects the availability and abundance of scenarios.

1.3 Approach

This work treats crowdsourced delivery systems and its operational decisions as an offline optimization problem referred as crowdshipping problem. In order to solve proposed crowdshipping problem, metaheuristic algorithms are utilized to tackle intractability aspect of the proposed formulation. Proposed metaheuristics are implemented for benchmarking purposes and obtained results from experiments are presented with analysis and discussion.

1.3.1 Problem Formalization

An offline optimization problem formulation is proposed in this thesis, based on a formulation proposed by Le et al. (2021) which integrates several components of crowdshipping problem formulations in the literature. The formulation proposed by my thesis inspired by more traditional optimization problem formulation such as PDPTW by Ropke and Cordeau (2009) and other works focuses on crowdshipping problems. As a result, my formulation allows the adaptation of solution methods that has been proposed for both traditional and crowdshipping-based problems.

1.3.2 Solution Methodology

The formulation proposed in this work is adapted from Le et al. (2021). However, authors of the mentioned work do not provide an explicit solution approach to their formulation. I adapt several heuristic operations and metaheuristic optimization algorithms from other more traditional optimization problems for solving proposed crowdshipping problem.

1.3.3 Experimentation and Analysis

In order to test the performance of mentioned solution metaheuristics, an experimental procedure is presented in detail which involves instance generation, parameter settings for proposed algorithms and experimental specifications. Results of the related experiments are reported and analyzed in comparative manner in order to assess the performance of metaheuristic algorithms.

1.4 Organization

General overview of this thesis and its approach to crowdshipping problem presented as follows. Section 2 provides necessary related works about the formulation, solution and bencmarking of crowdshipping problem. Section 3 describes more formal problem description and formulation for crowdshipping. Section 4 describes technical details for implementation of solution methodlogy. Section 5 describes experimental setup. Section 6 presents the results of experiments described in Section 5. Section 7 provides analysis and discussion about the provided results from Section 6. Finally, Section 8 provides a conclusion section.

CHAPTER 2: RELATED WORK

This section provides relevant literature about both formulation and solution works about the crowdshipping problem provided in this thesis. Survey works by Alnaggar et al. (2021); Laporte et al. (2015); Mourad et al. (2019) can be visited for more detailed and through treatment of the literature. Formulation approaches for crowdshipping problems that exists in the literature heavily relies on the more traditional routing problems such as Traveling Salesperson Problem(TSP), Vehicle Routing Problem(VRP) and Pickup and Delivery Problem(PDP). To give a general definition of the problems, the TSP problem can be considered as a route optimization problem where a travelling salesperson, who will start and finish his route in the same city, visits all cities with the lowest total distance. Continuing from the previous example, we can think of the VRP problem as the multi-salesperson case of the TSP problem where each city is visited by at least one salesperson, and the PDP problem as the VRP problem variation with the pickup and delivery operations. Especially formulations regarding to PDP literature, for example Dumas et al. (1991), shows resemblance to most of the related crowdshipping-based formulations in the literature.

A notable example for formulating crowdshipping is Arslan et al. (2019), which provides solid framework for crowdsourced delivery operations via providing a dynamic pickup and delivery problem formulated as a matching problem such as in Stiglic et al. (2015). Even though the mentioned work is exceptional, the authors of the mentioned work did not treat pricing and compensation aspects in their mathematical formulation for crowdsourced delivery problem. Including pricing and compensation aspects in formulation generalizes real world scenarios involves acceptance criterias of both customers and crowdshippers. Solution methodology presented in Arslan et al. (2019) involves solution of routing subproblems formulated as Traveling Salesperson with Time Windows and Precedence constraints (TSP-TWPC) as in Mingozzi et al. (1997). Solution methodology and problem formulations are supplemented via theoretical insight from Stiglic et al. (2015) which are formulated as formal observations.

Another approach to formulate crowdsourced delivery problem is to introduce occasional drivers(OD) to more traditional and well studied optimization problems. Introducing occasional drivers(OD) to formulate crowdshipping problems is a highly accepted approach among the literature. Examples for introducing occasional drivers(OD) to mathematical formulation of crowdshipping include but not limited to related works such as Ghaderi et al. (2022); Archetti et al. (2016); Macrina et al. (2017); Santini et al. (2022); Voigt and Kuhn (2022); Dahle et al. (2017); Dahle et al. (2019); Torres et al. (2022); Yıldız (2021); Gdowska et al. (2018). In general, in the aforementioned studies, occasional drivers are modeled as an extension of other drivers, and they are

put forward as occasional to be used optionally. The main reason behind not following same formulation in this work is that generalization of both classical and occasional drivers can be achieved by more generalized formulations, parameter fixations and instance designs.

The mathematical program for crowdshipping problem that is proposed in this thesis is based on modifications done to formulation proposed by Le et al. (2021). Modifications to mentioned original work is based on changes to objective function and elimination of decision variables by fixating those decision variables to given parameters, which are related to compensation and pricing procedure of crowdshipping system. The modification done to objective value to reformulate the compensation part of the objective value to include only extra detours taken from original route crowdshipper are included in compensation process for convenience of the notation. The other modification is to eliminate decision variables which are designed to represent pricing and compensation decision in order to reduce complexity of mathematical formulation.

Because most of the crowdshippind-based formulation are not presented with solution techniques, several metaheuristic solution algorithms which were proposed for other more traditional optimization problems are adapted in this work to tackle crowdshipping problem. The Simulated Annealing algorithm is based on its simplest form which a prospective reader can refer to Delahaye et al. (2019). The General Variable Neighborhood Search (GVNS) algorithm proposed in this work based on Alcaraz et al. (2019) where adaptation of GVNS from de Armas and Melián-Batista (2015);De Armas et al. (2015). The Adaptive Large Neighborhood Algorithm proposed in this work is adapted from Ropke and Pisinger (2006). The reason for choosing the mentioned metaheuristic algorithms is that they can be coded structurally similar to each other and they utilize the proposed heuristic operations very similarly. In addition, all of them are trajectory-based algorithms which implies that population-based algorithms are not used.

Set of heuristics are adapted from different works for proposed metaheuristics. Insertion and Deletion heuristics are adapted from Ropke and Pisinger (2006). Inter&Intra Relocate and Exchange heuristics are adapted from Ma et al. (2021). The main reason for choosing these heuristic operations is that those heuristics and adaptations of those heuristics have been extensively used in the literature. Additionally, those heuristics are applicable to other traditional optimization problems.

CHAPTER 3: PROBLEM FORMULATION

Let n be the number of requests and m be the number of crowdshippers. The crowdshipping problem is defined on a graph $G = (V, E)$ with set of nodes $V = P \cup D \cup \{\tau_1, .., \tau_m\} \cup \{\bar{\tau}_1, .., \bar{\tau}_m\}$ and set of edges $E \subseteq V \times V$ where $P = \{1, .., n\}$ is the set of all pickup nodes, $D = \{n + 1, .., 2n\}$ is the set of all delivery nodes, $M = \{1, .., m\}$ is the set of all crowdshippers and for each crowdshipper $k \in M$ origin and destination nodes of crowdshipper defined as $\tau_k = 2n + k$ and $\bar{\tau}_k = 2n + m + k$. Q_k is the capacity of crowdshipper $k \in M$. For each node $i \in V$ a service time s_i, a time window $tw_i = [ea_i, ld_i]$ where ea_i is the earliest arrival and ld_i is the latest departure, a load of packages l_i at node $i \in P \cup D$ where l_i is positive for pickup nodes and negative for delivery nodes are defined. ETP_i^k is the expected payment per distance for crowdshipper $k \in M$ for each extra distance taken by the crowdshipper. WTP_i is the maximum amount of price for packet $i \in P$, travel distance d_{ij} where $(i, j) \in E$.

There are four types of decision variables, where $x_{i,j}^k$ is a binary variable that represents traversal of crowdshipper $k \in M$ at edge $(i, j) \in E$, S_i^k is a non-negative integer that represents arrival time of crowdshipper $k \in M$ at node $i \in V$, L_i^k is a non-negative integer that represents the load of crowdshipper $k \in M$ at node $i \in V$, z_i is a binary variable that represents status of packet $i \in P$ and $z_i = 1$ if it is placed in the request bank which signifies that the request will not be served by any crowdshipper.

Table 1. Formulation Overview

Sets:	
P	set of all pickup nodes, $P = \{1, 2, ..., n\}$
D	set of all delivery nodes, $D = \{n+1, n+2, ..., 2n\}$
M	set of all crowdshippers, $\|M\| = m$
Indices:	
i	Index of the request by its pickup node, $i \in P$
k	Index of the crowdshipper, $k \in M$
Parameters:	
n	Number of requests
m	Number of crowdshippers
$\tau_k, \bar{\tau_k}$	Start and end stop of crowdshipper k, $\tau_k = 2n+k$ and $\bar{\tau_k} = 2n+m+k$
ea_i, ld_i	Time window associated with each request where ea_i is for earliest arrival and ld_i is for latest departure.
l_i	load of package at node i which is positive for pickup node $i \in P$ and negative for delivery node $i \in D$
WTP_i	Maximum price that a sender is willing to pay (WTP) $\forall i \in P$.
ETP_i	Minimum compensation that crowdshipper k expects to be paid (ETP) per km, $\forall k \in M$
Q_k	Capacity of crowdshipper k
Variables:	
$x_{i,j}^k$	Binary variable for traversal from node $i \in N$ to node $j \in N$ depicted as an edge travesal as $(i, j) \in E$ of crowdshipper $k \in M$
z_i^k	Binary variable for unassigned requests. $z_i = 1$ for unassigned requests resides in request bank, $z_i = 0$ otherwise.
S_i^k	Non-negative integer variable representing the time that crowdshipper k starts at node i.
L_i^k	Non-negative integer variable representing the upper bound for the load of packages that crowdshipper k carrying after serving node i

The mathematical model is depicted below:

Objective:

$$\max_{x_{i,j}^k, S_i^k, L_i^k, z_i} \quad \sum_{i \in P} WTP_i(1 - z_i) - \sum_{k \in K}(\sum_{(i,j) \in E} d_{ij} x_{ij}^k - d_{\tau_k \bar{\tau}_k}) ETP_k \tag{1}$$

s.t.

$$\sum_{k \in M} \sum_{j \in V} x_{i,j}^k + z_i = 1, \forall i \in P \tag{2}$$

$$\sum_{j \in V} x_{i,j}^k - \sum_{j \in V} x_{j,n+i}^k = 0, \forall i \in P, \forall k \in M \tag{3}$$

$$\sum_{j \in P \cup \bar{\tau}_k} x_{\tau_k,j}^k = 1, \forall k \in M \tag{4}$$

$$\sum_{i \in D \cup \tau_k} x_{i,\bar{\tau}_k}^k = 1, \forall k \in M \tag{5}$$

$$\sum_{i \in V} x_{i,j}^k - \sum_{i \in V} x_{j,i}^k = 0, \forall i \in D, \forall k \in M \tag{6}$$

$$x_{i,j}^k = 1 \rightarrow S_i^k + s_i + t_{i,j} \leq S_j^k, \forall (i,j) \in E, \forall k \in M \tag{7}$$

$$ea_i \leq S_i^k \leq ld_i, \forall i \in V, \forall k \in M \tag{8}$$

$$S_i^k \leq S_{n+i}^k, \forall i \in V, \forall k \in M \tag{9}$$

$$x_{i,j}^k = 1 \rightarrow L_i^k + l_i \leq L_j^k, \forall (i,j) \in E, \forall k \in M \tag{10}$$

$$L_i^k \leq Q^k, \forall i \in V, \forall k \in M \tag{11}$$

$$L_i^{\tau_k} = L_i^{\bar{\tau}_k} = 0, \forall i \in V, \forall k \in M \tag{12}$$

$$x_{i,j}^k \in \{0,1\}, \forall (i,j) \in E, \forall k \in M \tag{13}$$

$$z_i \in \{0,1\}, \forall i \in P \tag{14}$$

$$S_i^k \geq 0, \forall i \in P, \forall k \in M \tag{15}$$

$$L_i^k \geq 0, \forall i \in P, \forall k \in M \tag{16}$$

$$Q^k \geq 0, \forall k \in M \tag{17}$$

The objective function maximizes the total profit of the system. Constraint (2) ensure that each request either visited by a carrier or placed in the request bank. Constraint (3) ensures that if a carrier visited a pickup node, that carrier has to visit the delivery node of visited pickup node. Constraints (4) and (5) ensure that carrier starts at origin and ends at destination stop. Constraint (6) generates a route between origin and destination stop. Constraint (7) ensure that each node has to be visited within its time window. Constraint (8) ensure that pickup node of a packet has to be visited before its delivery node. Constraints (9), (10), (11) ensure the consistency of load

along the route. Constraint (12) disables excess load at origin and destination stops of the crowdshippers. Constraints (13), (14), (15), (16) and (17) are for describing non-negative integer and binary values of the decision variables.

Note that provided formulation has capability to generalize traditional routing optimization problems such as TSP, VRP and PDP given as an example. For all provided traditional generalized problems, objective function term of $WTP_i = 0, \forall i \in P$ and decision variable $z_i = 0, \forall i \in P$.

- **TSP:** TSP case described as where number of crowdshippers equal to $|m| = 1$, sole crowdshipper starts and end at same location $\tau_0 = \bar{\tau_0}$ and all pickup nodes are also delivery nodes such as $d_{i,n+i} = 0, \forall i \in P$
- **VRP:** VRP case can be summarized as where all crowdshippers start and end at same location $\tau_k = \bar{\tau_k}, \forall k \in M$ where all pickup nodes are also delivery nodes such as $d_{i,n+i} = 0, \forall i \in P$.
- **PDP:** PDP is almost the same problem with crowdshipping where all crowdshippers start and end at same location $\tau_k = \bar{\tau_k}, \forall k \in M$.

CHAPTER 4: SOLUTION METHODOLOGY

The formulation presented in previous section generalizes several NP-complete problems which implies that crowdshipping problem formulated in this thesis is also in NP-complete. Such computational decision problems makes exact methods to be implausible in real-life scenarios because of the combinatorial explosion. Approximation algorithms can also be utilized for the formulation proposed in this thesis but any particular approximation scheme may not apply to other generalized classical routing problems. Hence, metaheuristics based on route modification and construction heuristics are utilized in this thesis.

Metaheuristics algorithms that are utilized in thesis accepts initial solutions as an input and iteratively improves on given initial solution. Each solution is represented as a set of routes. Each route is represented as a sequence of nodes. Improvements to the initial solutions are conducted by heuristic operations that manipulate sequence order of the routes and location changes of the nodes.

4.1 Heuristics

This section describes three removal, three insertion, four exchange and four relocation heuristics. **Insertion** and **Deletion** heuristics treats the route of crowdshipper as a sequence of locations and insertion/deletion operations related with route sequence. **Exchange** heuristics involves swap operations and **Relocate** heuristics involve combined insertion & deletion heuristic application crowdshippers route where **Inter** and **Intra** terminology refers to the number of crowdshippers that involved in process. **Inter** operations describes that two crowdshippers are involved in operation and **Intra** operations states that only one crowdshippers route is modified in heuristic application. All **Insertion** and **Removal** heuristics are **Intra** operations which is not explicitly stated in order to provide terminological relevance to original reference to Ropke and Pisinger (2006).

All **Intra** heuristics selects and evaluates all possible modifications according to a specific measure. Evaluation results of possible modifications are serialized and sorted in an abstract data such as list which is utilized in this work and referred as L. Sorting metric is provided in description of the given **Intra** heuristic but all sortings done via best-improving to worst-improving order. Final stage of the **Intra** heuristics involve the selection of a modification from the constructed list L. A random number $y \in [0, 1]$ is calculated for selection process and $L[y^p|L|]$ represents the selection rule of modification in the list L where $p \leq 1$ is the randomization parameter which higher values of p results in more greedy selections and lower values are associated with more random actions at $p = 1$ gives most randomization in selection process. Inter

operations differs from **Intra** operations by selecting of two crowdshippers instead of one crowdshipper.

Inter heuristics select a target crowdshipper and a candidate crowdshipper randomly. An anchor request is selected randomly from target crowdshipper. The remaining procedure is identical to **Intra** heuristics. Every single exchange and relocate operations between anchor request are serialized and sorted in a list L and selected with $L[y^p|L|]$ rule.

4.1.1 Removal Heuristics

Removal heuristics randomly selects a crowd-shipping route and selects q number of requests to be removed from the route of selected crowdshipper and returns removed requests as output. If the q value is bigger than the length of the crowdshippers route, all requests in crowdshippers route are removed. The randomization parameter p is utilized in removal heuristics and described previously. Removal heuristics are sequential.

Algorithm 1 Removal Heuristic($S \in solutions$, $q \in N^+$, $p \in R^+$)

1: Initialize removed request bank R for output
2: **while** q > 0 **do**
3: All request candidates for removal stored in array L and sorted by defined metric
4: choose a random number $y \in [0, 1)$
5: select to be removed request such that $r = L[y^p|L|]$
6: remove request r from solution S and add it to request bank R
7: q = q - 1
8: **end while**
9: return (S, R)

Shaw Removal Heuristic As proposed by Shaw (1997),Shaw (1998) and adopted by Ropke and Pisinger (2006), *Shaw Removal Heuristic* probabilistically inclined to select more similar requests in order to obtain more diverse and perhaps more profitable solutions. Because general intuition suggests that removing very different requests are most likely to be inserted in positions such that results in similar or worse objective values. In order to measure relatedness, same measure as in Ropke and Pisinger (2006) is utilized. Relatedness measure is a weighted sum of distance , time and capacity terms with weights ϕ,χ and Ψ. It is calculated as:

$$R(i, j) = \phi(d_{A(i),A(j)} + d_{B(i),B(j)}) + \chi(|T_{A(i)} - T_{A(j)}| - |T_{B(i)} - T_{B(j)}|) + \Psi(l_i - l_j)$$

where $A(i)$, $B(i)$ and T_i denotes the pickup location, delivery location and arrival time of request i, and l_i is the load of crowdshipper at location i.

Worst Removal Heuristic The *Worst Removal Heuristic* removes requests with the least decrease in profits. Each possible candidate for removal operation is evaluated by the minimum decrease in profits.

Random Removal Heuristic The random removal heuristic randomly removes a requests from a randomly selected crowdshippers route. As described in Ropke and Pisinger (2006), setting $p = 1$ of *Worst Removal Heuristic* or *Shaw Removal Heuristic* is one way to implement *Random Removal Heuristic* which can be done in a more computationally efficient manner.

4.1.2 Insertion Heuristics

Insertion heuristics take set of requests U and inserts these requests in possible candidate crowdshipper routes. For each request in U all candidates are evaluted. Insertion Heuristics are sequential which implies that an insertion heuristic sequentially selects a crowdshipper route and inserts a single request to the selected crowdshippers route one by one. A drawback of sequential insertion is that, after inserting a request from request bank U may prevents other requests to be inserted in routes of other crowdshippers. This complication will give rise to infinite loops if the Insertion Heuristic tries to insert every request in U without backtracking, hurting computational performance of Insertion Heuristic and still possibility of infinite loop is not prevented. In order to overcome this complication, max number of insertion trials is given as an input t.

Algorithm 2 Insertion Heuristic($S \in solutions$, $U \in requests$,$q \in N^+$,$p \in R^+$)

1: **for** $req \in U$ **do**
2: sort feasible vehicle candidates on array L for insertion with defined metric
3: choose a random number $y \in [0, 1)$
4: select request to be inserted such that $r = L[y^p|L|]$
5: insert request r to solution S and remove it from U
6: **end for**

Random Insertion Heuristic The *Random Removal Heuristic* randomly inserts given requests in randomly selected crowdshippers routes. Similar to *Random Removal Heuristic*, *Random Insertion Heuristic* is the special case of other insertion heuristics where $p = 1$.

Greedy Insertion Heuristic The greedy insertion heuristic evaluates and considers every insertion candidate crowdshipper for given requests in most profitable positions

of candidate crowdshipper routes.

Regret Insertion Heuristic The regret insertion heuristic calculates the profit difference between inserting at most profitable location with k'th most profitable location. If *Regret Insertion Heuristic* value of k is defined as $k = 2$, evaluation metric of insertion heuristic is the profit difference between the most profitable location and the second most profitable location.

4.1.3 Exchange Heuristics

Exchange heuristics swaps the places of pickup and delivery stops of requests within a single crowdshipper's route or between two distinct crowdshippers' routes. To be more specific, given an anchor request R_A with pickup stop P_A and D_A and a candidate request R_C with pickup stop P_C and D_C, locations of P_A with P_C and locations with D_A and D_C are exchanged.

Algorithm 3 Exchange Heuristic($S \in solutions$, $p \in R^+$)

1: randomly select vehicle(s)
2: randomly select an anchor request r from vehicles
3: sort each possible exchange operation on array L via defined metric
4: choose a random number $y \in [0, 1)$
5: select request to be exchanged such that $c = L[y^p|L|]$
6: exchange request c with anchor request r

Random Intra-Exchange The *Random Intra-Exchange* heuristic randomly selects two requests from a randomly selected crowdshipper's route and swaps them.

Greedy Intra-Exchange The *Random Intra-Exchange* heuristic randomly selects a request from a randomly selected crowdshipper's route and swaps the selected request with another request of selected crowdshipper that results in higher profit.

Random Inter-Exchange The *Random Intra-Exchange Heuristic* randomly selects a pair of crowdshippers and then randomly selects, an anchor request assigned to the first crowdshipper. After the selection of crowdshippers and the anchor request, *Random Intra-Exchange Heuristic* swaps the anchor request with a randomly selected request assigned to the second crowdshipper.

Greedy Inter-Exchange The *Greedy Intra-Exchange Heuristic* randomly selects a pair of crowdshippers, and then randomly selects an anchor request assigned to the first crowdshipper. After the selection of crowdshippers and the anchor request, *Greedy Intra-Exchange Heuristic* considers the anchor request and each of the requests assigned

to the second crowdshipper and calculates the profits for each potential swap operation. Finally, *Greedy Intra-Exchange Heuristic* swaps the two requests with the highest increase in profits.

4.1.4 Relocate Heuristics

Relocation heuristics remove a request from a randomly selected target crowdshipper and inserts that request to a selected candidate crowdshipper's route. For Intra-Relocate heuristics, the target and the candidate crowdshippers are identical. For Inter-Relocate heuristics, the target and the candidate crowdshippers are different. Relocate Heuristics can easily be implemented via utilizing insertion heuristics.

Algorithm 4 Relocate Heuristic($S \in solutions$, $p \in R^+$)

1: randomly select vehicle(s)
2: randomly select an anchor request r from vehicles
3: remove selected request r from solution S
4: sort each possible reinsertion location pair on array L via defined metric
5: choose a random number $y \in [0, 1)$
6: select location pair from L such that $(p, d) = L[y^p|L|]$
7: reinsert request r at selected location (p, d)

Random Intra-Relocate The *Random Intra-Relocate* heuristic randomly selects a request and changes the order in its crowdshipper's route.

Greedy Intra-Relocate The *Random Intra-Relocate Heuristic* randomly selects a request and changes the order in its crowdshipper's route to the most profitable location.

Random Inter-Relocate The *Random Inter-Relocate Heuristic* removes a randomly request in a randomly selected crowdshipper and randomly reinserts the selected request in another randomly selected crowdshipper.

Greedy Inter-Relocate The *Random Inter-Relocate Heuristic* removes a random request in a randomly selected crowdshipper and reinserts the selected request in another randomly selected crowdshipper in the most profitable location of the randomly selected crowdshipper's route.

4.2 Solution Methods

This subsection introduce solution methods used in experiments which are Local Search (LS), Simulated Annealing(SA), Adaptive Large Neighborhood Search(ALNS) and General Variable Neighborhood Search(GVNS). All the algorithms mentioned were chosen because they are structurally similar to each other and can be converted to

each other with minor changes. Similar structural similarities can be demonstrated by examining the pseudocode. For example, while the temperature mechanism is the same in both ALNS and SA algorithms, their heuristic selection mechanism are different. All proposed solution methods require an initial feasible solution, which is constructed via *Greedy Insertion Heuristic* that takes all unassigned requests as an input U.

4.2.1 Local Search(LS)

The simple Local Search algorithm utilizes insertion and removal heuristics in a greedy fashion with given randomization parameter p. The number of requests that will be removed is denoted by q and determined via random selection between 4 and $max(5, \xi n)$, where ξ determines the maximum percentage amount of the n number of requests can be taken into account for each reinsertion iteration.

Algorithm 5 Local Search($S \in solutions$, $p \in R^+$)

$\bar{S} = S$
while time limit not exceed **do**
    q = a random number selected between [4,max(5, ξn)]
    U = remove q requests from $\bar{S}$ via *Worst Removal Heuristic* with parameter p
    reinsert removed requests U in $\bar{S}$ via *Greedy Insertion Heuristic* with parameter p
    if $obj(\bar{S}) > obj(S)$ **then**
        $S = \bar{S}$
    end if
end while
return S

4.2.2 Simulated Annealing(SA)

The Simulated Annealing algorithm randomly removes and inserts requests in every iteration via *Random Removal Heuristic* and *Random Insertion Heuristic*. As suggested in original SA, a temperature value T is utilized for accepting less profitable solutions. Each non-improving solution is accepted with probability $e^{-\frac{f(s)-f(\bar{s}}{T}}$. The initial temperature value T_{init} and cooling factor c are provided as input parameters to simulated annealing algorithm. For each iteration temperature value is updated as $T = T \cdot c$.

Algorithm 6 Simulated Annealing($s \in solutions$, $c \in [0, 1]$, $T > 0$)

$\bar{S} = S$
while time limit not exceed **do**
  q = a random number selected between [4,max(5, ξn)]
  U = remove q requests from $\bar{S}$ via *Random Removal Heuristic*
  reinsert removed requests U in $\bar{S}$ via *Random Insertion Heuristic*
  if $obj(\bar{S}) > obj(S)$ **then**
    $S = \bar{S}$
  else with probability $e^{-\frac{obj(S)-obj(\bar{S})}{T}}$
    $S = \bar{S}$
  end if
  $T = T \cdot c$
end while
return S

4.2.3 Adaptive Large Neighborhood Search(ALNS, No Temp ALNS)

The adaptive large neighborhood search is adapted from Ropke and Pisinger (2006). Unlike other proposed algorithms, ALNS heavily relies on large moves rather than iterating over small moves with combined randomized perturbations. Such large neighborhood moves are conducted via applying destroy and repair operations, based on ruin and recreate method proposed by Schrimpf et al. (2000). The ALNS algorithm utilizes all proposed removal and insertion heuristics, removal heuristics are regarded as destroy methods and insertion heuristics are regarded as repair methods. The primary reason behind treating removal and inserting heuristics as large neighborhood operation is that removal and insertion heuristics modify more requests than other heuristics that has been proposed in this work. The ALNS algorithm takes an initial solution as an input and improves the given solution via utilizing set of insertion and deletion heuristics via deleting and inserting randomly selected $q \in [4, max(5, n\xi]$ number of requests for each iteration where n is the number of requests and $\xi \in [0, 1]$ is an input parameter to limit maximum number of requests will be taken into account for each iteration. Insertion and deletion heuristics independently has two set of weights for the roulette wheel selection. For each type heuristic, given the k number of specific heuristic type, weights of given type of heuristic represented as

$$\frac{w_j}{\sum_{i=k}^{k} w_i}$$

Adjustment procedure of the weights is performed in each iteration and divided into segments for entire search procedure where each segment represents 100 iterations. For each segment, weight update rule is dependent on weights of heuristic i at segment j which is w_{ij}, the score parameter π_i, the counter parameter θ_i to keep the track of

the number of times heuristic i attempted during last segment and a reaction factor r as step size of the weight adjustment. Given parameters, weight update rule is

$$w_{i,j+1} = w_{i,j}(1 - r) + r\frac{\pi_i}{\theta_i}$$

Adjustment procedure of the weights w_j relies on π_i which is calculated via three parameters ω_1, ω_2 and ω_3. Each parameter is associated with score values that will be added to π_i in order to measure and update the performance of selected heuristics. The first parameter ω_1 associated for heuristic was able to find a new global best solution. Second parameter ω_2 associates with unexplored solutions with improvements which resides to a heuristic that was able to find a solution which is not accepted before with better profits than the current solution. Third parameter ω_3 is associated with unexplored and accepted solutions given that an accepting mechanism(which will be introduced in later) accepts a solution which is not accepted before with less profits.

Two variants of ALNS mentioned in this work differs from each other with their accepting mechanisms. The first variant of ALNS utilizes a temperature mechanism with initial temperature T_{init}, current temperature T and cooling factor $c \in [0, 1]$ where temperature mechanism can accept worse solution $\bar{S}$ to more profitable current solution S with probability $e^{-\frac{f(S)-f(\bar{S}}{T}}$. The current value of temperature is updated via cooling mechanism $T = Tc$ for each iteration. The second variant of ALNS(No Temp ALNS) simply follows the same procedure without the temperature mechanism.

Algorithm 7 Adaptive Large Neighborhood Search($s \in solutions$, r, π, ω, $c \in [0, 1]$, $T > 0$)

$\bar{S} = S$
$\tilde{S} = S$
Initialize weights w
Initialize counters θ, scores π, iterations $iter$ and parameters ω
Allocate memory for visited solutions as vs
while time limit not exceed **do**
  q = a random number selected between [4,max(5, ξn)]
  apply *Removal Heuristic* and *Insertion Heuristic* to $\bar{S}$ via roulette wheel selection
  if $obj(\bar{S}) > obj(S)$ **then**
    $S = \bar{S}$
    if $\bar{S} \notin vs$ **then**
      $\tilde{S} = \bar{S}$
      update scores π with ω_1
    else
      update scores π with ω_2
    end if
    add $\bar{S}$ to visited solutions vs
  else with probability $e^{-\frac{obj(S)-obj(\bar{S})}{T}}$
    $S = \bar{S}$
    if $\bar{S} \notin vs$ **then**
      $\tilde{S} = \bar{S}$
      update scores π with ω_3
    end if
    add $\bar{S}$ to visited solutions vs
  end if
  if $iter\%100$ **then**
    update weights w
    reset counters θ and scores π
  end if
  $T = c \cdot T$
end while
return S

4.2.4 General Variable Neighborhood Search (GVNS)

The General Variable Neighborhood Search adapted from (Armas and Melian-Batista). The GVNS algorithm takes an initial feasible solution as an input and iteratively applies Shaking process followed by the Variable Neighborhood Descent (VND) that improves the solution obtained from the Shaking process. For each iteration of GVNS, an input parameter h_{max} controls the maximum number of consequent non-improving iterations which is also implemented in VND process as a nested strategy.

Algorithm 8 General Variable Neighborhood Search($s \in solutions$,$h_{max} \in N$)

```
1: while time limit not exceed do
2:     while h < h_max do
3:         apply Shaking to the solution S
4:         apply VND to S and obtain S̄
5:         if obj(S̄) > obj(S) then
6:             S = S̄
7:             h = 1
8:         else
9:             h = h + 1
10:        end if
11:    end while
12: end while
13: return S
```

Shaking Process The Shaking process applies heuristics as a random perturbation, in order to escape local optimal solutions. This process randomly selects two random heuristics from the randomized heuristic set of *Random Intra-Exchange*, *Random Intra-Relocate*, *Random Inter-Relocate*, *Random Inter-Exchange* and combined *Random Insertion Heuristic* with *Random Removal Heuristic*. The selected heuristics will be applied to the current solution S to obtain $\bar{S}$ as the output of Shaking procedure.

Algorithm 9 Shaking($s \in solutions$)

```
1: randomly select two heuristics from the heuristic set
2: apply selected heuristics to s to obtain s̄
3: return s̄
```

Variable Neighborhood Descent The General Variable Neighborhood Descent process is the exploitation procedure of the GVNS. The VND process iteratively selects and applies a randomly selected heuristic from the predefined set of greedy heuristics which consists of Greedy Intra-Exchange, Greedy Intra-Relocate, Greedy Inter-Relocate, Greedy Inter-Exchange and combined Worst Removal Heuristic & Greedy Insertion Heuristic. The maximum iteration parameter h_{max} will be passed from GVNS as an input to the VND. For each iteration an iteration counter h will be kept. Each improving solution will reset the counter h or otherwise increments it by 1 for non-improving solutions until counter h reaches the maximum iteration h_{max}.

Algorithm 10 Variable Neighborhood Descent($s \in solutions$,$h_{max} \in N$)

while time limit not exceed **do**
    while $h < h_{max}$ **do**
        apply Shaking to the solution S
        apply VND to S and obtain $\bar{S}$
        if $obj(\bar{S}) > obj(S)$ **then**
            $S = \bar{S}$
            $h = 1$
        else
            $h = h + 1$
        end if
    end while
end while
return S

CHAPTER 5: EXPERIMENTAL SETUP

5.1 Instance Generation & Format:

Experiments are conducted with three set of instances to measure the performance of proposed solution technique for proposed crowdshipping problem formulation. Each set of instances concerns coupled number of requests and crowdshippers as instance size which are 50&50,75&75,100&100. Each set of instances consists of 100 randomly generated instances.

Instance Generation: Coordinates of pickup and delivery locations of requests and origin and destination locations of crowdshippers are randomly generated between latitude interval of $[38°N, 38°5'N]$ and longtitude interval of $[27°W, 27°5'W]$ which covers the city of Izmir, Turkey. Distances between coordinates calculated via Haversine Formula. ETP and WTP values are based on Le and Ukkusuri (2019a), Le and Ukkusuri (2019b), Le et al. (2021) where ETP values are randomly generated between \$0.5 and \$1 and WTP values are randomly generated between \$5 and \$10. All time windows are randomly generated between $[0, 86400]$ time interval which represents 24-hour by seconds. Load values of requests are randomly generated between $[10, 100]$ and capacity values of crowdshippers are randomly generated between $[100, 1000]$. All randomly generation process of instances are based on uniform distributions.For each instance in all instance sets, an initial solution is generated and provided to metaheuristic algorithms. Iterations of all algorithms are timeboxed as 100, 150 and 300 seconds based on respective instance sizes of 50 & 50, 75 & 75, 100 & 100.

Table 2. Example Instance File

id	type	x1	y1	x2	y2	ea1	ld1	ea2	ld2	lc	EW
0	C	38.473	27.378	38.364	27.243	873	6589	18346	21488	707	0.8
1	C	38.443	27.290	38.391	27.315	3096	6085	14996	15069	109	0.6
0	R	38.026	27.462	38.421	27.046	3287	3290	10069	10410	92	8.0
1	R	38.295	27.160	38146	27.334	459	1201	9267	9512	57	5.0

Instance Format: Each instance contains id, row type ,coordinate info, time window, ETP & WTP and load & capacity related information.The first column in the file specifies the id of the crowdshipper and the request. Crowdshipper's id information are serialized first and then request's id information serialized. Columns "x1" and "y1" specify origin coordinates for crowdshippers and pickup coordinates for requests. Columns "x2" and "y2" specify destination coordinates for crowdshippers and delivery coordinates for the requests. Earliest arrival column "ea1" and latest departure column "ld1" specify the time windows associated with origin locations for the crowdshippers

and time windows associated with pickup locations for the requests. Earliest arrival column "ea2" and latest departure column "ld2" specify the time windows associated with destination location for the crowdshippers and time windows associated with delivery location for the requests. The "lc" column represents capacity information for crowdshippers and load information for requests. The "EW" column represents "expected to pay" parameter for crowdshippers and "willing to pay" information for requests which are presented in Section 3.

5.2 Algorithm Parameters:

Initial solutions constructed via *Greedy Insertion Heuristic* with $p = 2$. Local Search algorithm takes initial q value as $q = 5$ with q update value of $\epsilon = 0.4$ and all greedy heuristics takes p value as $p = 2$. All of ALNS algorithm parameters except cooling rate $c = 0.99$ and initial temperature value $T_{init} = 10$ are copied from Ropke and Pisinger (2006) where segment number defined as 100 iterations, p value of heuristics are $p = 2$, *Shaw Removal Heuristic* values are $\phi = 0.33$, $\chi = 0.33$, $\Psi = 0.33$, weight update probability value is $r = 0.1$, scores value of π_i calculated via $\omega_1 = 33$, $\omega_2 = 9$, $\omega_3 = 13$, epsilon value of q selection is $\epsilon = 0.4$ and regret value of *Regret Insertion Heuristic* is 2. All parameters of the second ALNS version without temperature and cooling mechanism are exactly same with first version of ALNS. Simulated Annealing algorithm utilizes same cooling rate, initial temperature and ϵ values with ALNS which are $c = 0.99$, $T_{init} = 10$ and $\epsilon = 0.4$. The GVNS algorithm utilizes $h_{max} = 10$ and $\epsilon = 0.4$ and VND process of GVNS utilizes $p = 2$.

CHAPTER 6: RESULTS

This section provides the results of described and conducted experiments in Section 5. In Table 3 depicts mean and standart deviation values of the solution's objective values by each algorithm for every instance set. In Figure 1 provides a box plot in order to observe differences among all algorithms for each instance set. In Table 4 presents one-way ANOVA comparison test results. For further evaluation, two type of result algorithms are introduced:

Chameleon: This is a hypothetical algorithm that mimicks the objective value of best performing algorithm for each trial conducted on each experiment set. The purpose of constructing this hypothetical algorithm is to see how much worse other algorithms can perform than a hypothetical best algorithm.

Init: Constructs initial solutions via iterative use of *Greedy Insertion Heuristic*. This is used to see how well other algorithms are making progress compared to the greedy construction algorithm.

Table 3. Results

Instances	**Algorithm**	**Mean**	**Std**
50&50	Chameleon	54.95	17.08
	GVNS	54.19	16.98
	LS	54.09	16.79
	No Temp ALNS	53.86	16.64
	ALNS	50.90	16.69
	SA	49.07	15.46
	Init	47.73	16.03
75&75	Chameleon	103.98	21.17
	LS	101.60	20.52
	No Temp ALNS	101.31	21.41
	GVNS	101.27	21.08
	ALNS	94.80	20.86
	SA	88.91	19.55
	Init	87.65	19.92
100&100	Chameleon	157.53	25.4
	LS	152.79	24.96
	GVNS	152.67	26.28
	No Temp ALNS	151.04	25.42
	ALNS	143.27	27.12
	SA	133.13	24.53
	Init	132.33	24.81

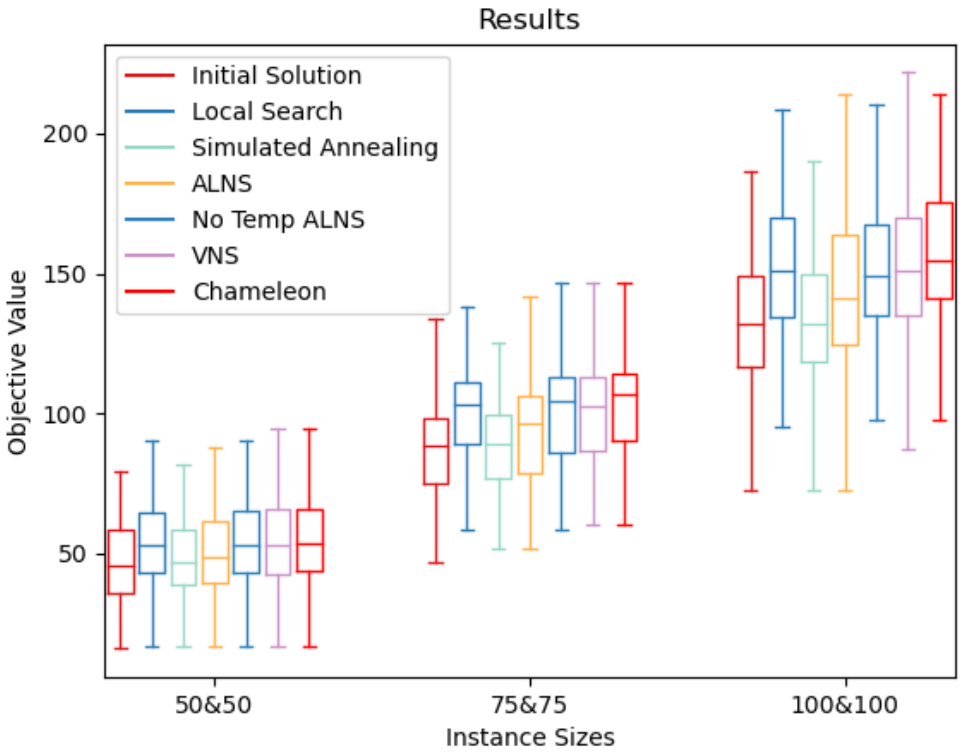

Figure 1. Box Plot of Results

As presented in Table 3 and Figure 1, GVNS performed best in first experiment set and Local Search algorithm performed best in second and third experiment sets. Overall objective values of algorithms increase as size of experiment sets increase. In addition to increase in objective values, standart deviation of all algorithms increase with the instance size. Mean objective values of the results produced by Simulated Annealing and No Temp ALNS algorithm were almost identical with the **Init** algorithm in all experiment sets. For all experiment sets, mean objective values of Local Search, No Temp ALNS and GVNS algorithms are close to each other and to the **Chameleon** algorithm.

Table 4. Comparison Tests

Instances	**Algorithms**	**Init**		**Chameleon**	
		f-score	**p-value**	**f-score**	**p-value**
50&50	GVNS	7.574	0.006	0.0966	0.756
	LS	7.428	0.007	0.126	0.723
	No Temp ALNS	6.966	0.009	0.204	0.651
	ALNS	1.865	0.173	2.827	0.094
	SA	0.359	0.549	6.425	0.012
	All Algorithms	1.948	0.101		
75&75	LS	23.331	2.742e-06	0.632	0.427
	GVNS	21.626	6.081e-06	0.801	0.371
	No Temp ALNS	21.402	6.756e-06	0.763	0.383
	ALNS	6.027	0.015	9.335	0.002
	SA	0.200	0.655	26.788	5.598e-07
	All Algorithms	7.249	1.121e-05		
100&100	LS	33.284	3.047e-08	1.747	0.187
	No Temp ALNS	27.342	4.332e-07	3.209	0.074
	GVNS	31.204	7.645e-08	1.741	0.188
	ALNS	8.770	0.003	14.584	0.0002
	SA	0.053	0.817	47.246	7.963e-11
	All Algorithms	10.742	2.353e-08		

Conducted comparison tests include results of "All Algorithms" rows to represent comparison test conducted with all proposed algorithms. Comparison tests based on "All Algorithms" rows reveals that for 50&50 instance set, performance of all algorithms are statistically similar where claim of statistically significant similarity are not applicable for 75&75 and 100&100 instance sets.

In order to test the validity of previous claims, set of statistical comparison tests based on one-way ANOVA is conducted to prove whether the performance differences of algorithms are statistically significant. Based on conducted comparison tests, neither Simulated Annealing nor ALNS algorithms show statistically significant differences from the **Init**. Remaining algorithms produced statistically significant improvements over the **Init** algorithm.

CHAPTER 7: DISCUSSION

Provided in results section, several key findings are observed:

- Top three best performing algorithms (LS,GVNS,No Temp ALNS) show no statistically significant differences from one and another.

The main reason for this finding may be that the algorithms mentioned show less randomness compared to other algorithms that shows worse performance. The mentioned algorithms are different from each other in terms of both heuristic selection, availability of the operations and exploration mechanism. The only common aspect of working principles in these algorithms is the possibility that they have chosen the greedy exploitative heuristics in a consistent and repetitive way as in the Local Search algorithm with controlling this selection with the q parameter.

- Top three best performing algorithms (LS,GVNS,No Temp ALNS) are as good as the hypothetical Chameleon algorithm.

This finding shows that the best working algorithm works as well as a hypothetically possible Chameleon algorithm described in Results section. Main reason behind such observation is that there is not much significant improvement in working principles by switching between algorithms. This observation shows that well designed heuristics are more important than the algorithms for producing much better and consistent results.

- The two worst performing algorithms (SA, ALNS) do not provide statistically significant improvement on the hypothetical Init algorithm.

This finding can be interpreted with the difference "lack of temperature mechanism" of ALNS and No Temp ALNS algorithms. This interpretation also shows that the ALNS algorithm, which is more randomized than the ALNS algorithm, performs worse, and the SA and ALNS algorithm gives worse results due to excessive randomization and prevents it from giving better results than a greedy solution construction algorithm that only repeats the greedy insertion operation. The main reason for this is that no matter how big the cooling factor is, when it randomly selects a worse solution in the initial steps, it is more difficult to return to a improving solution rather than to even declining to a worse solution.

Given key findings, best performing algorithms LS, GVNS and No Temp ALNS were able to improve initial solutions compared to worst performing algorithms SA and ALNS. Such results are attributed to lack of temperature mechanism in best performing algorithms because lack of temperature mechanism was able to increase performance of ALNS algorithm without changing any algorithmic aspect of ALNS. Not limited to previous discussion SA algorithm were not able outperform ALNS in second and

third instance sets which implies that the algorithmic differences between SA and ALNS do provide an increase in improvement capability of ALNS algorithm which do exists in other version of ALNS with no temperature mechanism. This implies that algorithmic difference between ALNS and SA which is included in no temperature version of ALNS and not attributed to other best performing algorithms clearly indicates that other algorithmic aspects do not provide such importance more than temperature mechanism.

Mentioned findings can support further evaluations with instance sets with more crowd-shippers and requests since increasing the size of instance sets strengthens previously mentioned key findings. More concrete discussion of the results is not possible since unavailability of handcrafted instances with optimal solutions. Randomized nature of instance generation procedure in this thesis may generate instances with caveats. Randomly generated instances can have highly symmetric formation of solution space where abundance of feasible solution with similar quality or highly constrained search space formation.

CHAPTER 8: CONCLUSION

The main focus of this thesis was on crowdsourced delivery systems where partial or total request deliveries are outsourced to crowdshippers in return of compensation for prospective crowdshippers extra effort. Such systems enable previously unavailable routes to traditional delivery operations, which may provide more cost efficient operational scenarios. In order to design operational decision problems of crowdsourced delivery systems, an offline optimzation problem referred as crowdshipping problem is formulated in this thesis. The formulation proposed in this thesis is a modified version of integrated formulation of crowdshipping problem done by Le and Ukkusuri (2019b). Modification involves elimination of two decision variables and reformulation of objective value. Unavailability of solution techniques and heuristics for the formulation proposed in this work, several metaheuristic algorithms and heuristic operations are provided to solve crowdshipping problem formulated in this thesis. In order to measure the relative solving quality of metaheuristics, an experimental setup is utilized and discussed. Experimental setup involves instance generation procedure, parameter fixation and design specifications of experiments conducted in this thesis. Results obtained from the experiments are presented with statistical comparison tests to comparative treatment of proposed algorithms. Given key findings, best performing algorithms LS, GVNS and No Temp ALNS on the average performed at similar levels and to the hypothetical **Chameleon** algorithm that mimics the results of most successful algorithm for each trial. Worst performing algorithms SA and ALNS were not able improve initial solutions provided by the **Init**. Conclusion drawn from the differences between worst performing and best performing algorithms were the temperature mechanism utilized in SA and ALNS was not beneficial and other algorithms with less randomization provide similar results were able to improve initial solutions. This indicates that designing handcrafted heuristics might be promising as future work.

REFERENCES

Alcaraz, J. J., Caballero-Arnaldos, L. and Vales-Alonso, J. (2019), *Rich vehicle routing problem with last-mile outsourcing decisions*, Transportation Research Part E: Logistics and Transportation Review 129, pp. 263–286.

Alnaggar, A., Gzara, F. and Bookbinder, J. H. (2021), *Crowdsourced delivery: A review of platforms and academic literature*, *Omega* 98, pp. 102139.

Archetti, C., Savelsbergh, M. and Speranza, M. G. (2016), *The vehicle routing problem with occasional drivers*, European Journal of Operational Research 254(2), pp. 472– 480.

Arslan, A. M., Agatz, N., Kroon, L. and Zuidwijk, R. (2019), *Crowdsourced delivery—a dynamic pickup and delivery problem with ad hoc drivers*, Transportation Science 53(1), pp. 222–235.

Boysen, N., Fedtke, S. and Schwerdfeger, S. (2021), *Last-mile delivery concepts: a survey from an operational research perspective*, Or Spectrum 43(1), pp. 1–58.

Dahle, L., Andersson, H. and Christiansen, M. (2017), *The vehicle routing problem with dynamic occasional drivers*, International conference on computational logistics, Springer, pp. 49–63.

Dahle, L., Andersson, H., Christiansen, M. and Speranza, M. G. (2019), *The pickup and delivery problem with time windows and occasional drivers*, Computers & Operations Research 109, pp. 122–133.

De Armas, J. and Melián-Batista, B. (2015), *Variable neighborhood search for a dynamic rich vehicle routing problem with time windows*, Computers & Industrial Engineering 85, pp. 120–131.

De Armas, J., Melian-Batista, B., Moreno-Perez, J. A. and Brito, J. (2015), *Gvns for a real-world rich vehicle routing problem with time windows*, Engineering Applications of Artificial Intelligence 42, pp. 45–56.

Delahaye, D., Chaimatanan, S. and Mongeau, M. (2019), *Simulated annealing: From basics to applications*, Handbook of Metaheuristics, New York: Springer, pp. 1-35.

Dumas, Y., Desrosiers, J. and Soumis, F. (1991), *The pickup and delivery problem*

with time windows, European journal of operational research 54(1), pp. 7–22.

Gdowska, K., Viana, A. and Pedroso, J. P. (2018), *Stochastic last-mile delivery with crowdshipping*, Transportation research procedia 30, pp. 90–100.

Ghaderi, H., Tsai, P.-W., Zhang, L., Moayedikia, A. (2022), *An integrated crowdshipping framework for green last mile delivery,* Sustainable Cities and Society 78, pp. 103552.

Laporte, G., Meunier, F. and Wolfler Calvo, R. (2015), *Shared mobility systems*, 4or 13(4), pp. 341–360.

Le, T. V. and Ukkusuri, S. V. (2019*a*), *Influencing factors that determine the usage of the crowd-shipping services*, Transportation Research Record 2673(7), pp. 550–566.

Le, T. V. and Ukkusuri, S. V. (2019*b*), *Modeling the willingness to work as crowd-shippers and travel time tolerance in emerging logistics services*, Travel Behaviour and Society 15, pp. 123–132.

Le, T. V., Ukkusuri, S. V., Xue, J. and Van Woensel, T. (2021), D*esigning pricing and compensation schemes by integrating matching and routing models for crowdshipping systems*, Transportation Research Part E: Logistics and Transportation Review 149, pp. 102209.

Ma, Y., Hao, X., Hao, J., Lu, J., Liu, X., Xialiang, T., Yuan, M., Li, Z., Tang, J., Meng, Z. (2021), *A hierarchical reinforcement learning based optimization framework for large-scale dynamic pickup and delivery problems*, Advances in Neural Information Processing Systems 34, pp. 23609–23620.

Macrina, G., Di Puglia Pugliese, L., Guerriero, F., Laganà, D. (2017), *The vehicle routing problem with occasional drivers and time windows*, International conference on optimization and decision science, Springer, pp. 577–587.

Mingozzi, A., Bianco, L. and Ricciardelli, S. (1997), *Dynamic programming strategies for the traveling salesman problem with time window and precedence constraints*, Operations research 45(3), pp. 365–377.

Mourad, A., Puchinger, J. and Chu, C. (2019), *A survey of models and algorithms for optimizing shared mobility*, Transportation Research Part B: Methodological 123, pp. 323–346.

Raj, A., Mukherjee, A. A., de Sousa Jabbour, A. B. L., Srivastava, S. K. (2022), *Supply chain management during and post-covid-19 pandemic: Mitigation strategies and practical lessons learned*, Journal of business research 142, pp. 1125–1139.

Ropke, S. and Cordeau, J.-F. (2009), *Branch and cut and price for the pickup and delivery problem with time windows*, Transportation Science 43(3), pp. 267–286.

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

Santini, A., Viana, A., Klimentova, X. and Pedroso, J. P. (2022), *The probabilistic travelling salesman problem with crowdsourcing*, Computers & Operations Research 142, 105722.

Schrimpf, G., Schneider, J., Stamm-Wilbrandt, H., Dueck, G. (2000), *Record breaking optimization results using the ruin and recreate principle*, Journal of Computational Physics 159(2), pp. 139–171.

Shaw, P. (1997), *A new local search algorithm providing high quality solutions to vehicle routing problems*, APES Group, Dept of Computer Science, University of Strathclyde, Glasgow, Scotland, UK 46.

Shaw, P. (1998), *Using constraint programming and local search methods to solve vehicle routing problems*, International conference on principles and practice of constraint programming, Springer, pp. 417–431.

Stiglic, M., Agatz, N., Savelsbergh, M., Gradisar, M. (2015), *The benefits of meeting points in ride-sharing systems*, Transportation Research Part B: Methodological 82, pp. 36–53.

Torres, F., Gendreau, M. and Rei, W. (2022), *Crowdshipping: An open vrp variant with stochastic destinations*, Transportation Research Part C: Emerging Technologies 140, 103677.

Voigt, S. and Kuhn, H. (2022), *Crowdsourced logistics: The pickup and delivery problem with transshipments and occasional drivers*, Networks 79(3), pp. 403–426.

Yıldız, B. (2021), *Express package routing problem with occasional couriers*, Transportation Research Part C: Emerging Technologies 123, 102994.