# DATA-DRIVEN LOCAL SEARCH HEURISTICS FOR BILEVEL NETWORK DESIGN PROBLEMS

by

İsmail Sevim

M.S., Industrial and Systems Engineering, İstanbul Şehir University, 2016

B.S., Industrial Engineering, Yıldız Technical University, 2013

Submitted to the Institute for Graduate Studies in
Science and Engineering in partial fulfillment of
the requirements for the degree of
Doctor of Philosophy

Graduate Program in Industrial Engineering

Boğaziçi University

2022

# ACKNOWLEDGEMENTS

This study was partially supported by Boğaziçi University Scientific Research Project under the Grant number: BAP 18461.

# ABSTRACT

## DATA-DRIVEN LOCAL SEARCH HEURISTICS FOR BILEVEL NETWORK DESIGN PROBLEMS

In the Network Design Problem (NDP), one aims to design the configuration of a network by installing links between a set of given nodes and determine the flow of a set of commodities over these installed links. In this thesis, we work on two bilevel NDPs where the sequential process of decision making approach is inherited. In the first bilevel NDP we model the strategic flight NDP of a small airline carrier as a network interdiction problem to analyse the maximum possible disruption in its flight network in the wake of virtual attacks performed by a competitor. We call this problem the $r$-Interdiction Network Design Problem with Lost Demand (RI-NDPLD). In the second problem, namely Bilevel Optimization Model for the Reconfiguration of refugee camp network (BOpt-RRC), the readjustment of configurations of refugee camp network are studied under the case of new refugee flows and possible variations in the supply of public service providers. We implement a set of generic local search matheuristics to solve both problems. In the Tabu Search (TS) proposed for the RI-NDPLD, we enhance the generic implementation with bound based pruning and regression based candidate solution set generation procedures to reduce the computational burden of explicit evaluation of all neighboring solutions, and hence, enjoy better diversification. We also implement a generic TS to solve the BOpt-RRC and devise an adaptive neighborhood selection procedure to incorporate into this implementation. In addition to the generic TS, we also implement a Variable Neighborhood Search (VNS) matheuristic and devise an association rule based injection procedure to incorporate *good* solution components to initial solutions obtained by usual random shaking. Experimental studies reveal promising results for the proposed methods.

# ÖZET

## İKİ SEVİYELİ AĞ TASARIM PROBLEMLERİ İÇİN VERİ GÜDÜMLÜ YEREL ARAMA SEZGİSELLERİ

Ağ Tasarım Problemi'nde (ATP), verili düğümler arasına bağlantılar kurularak ve bu bağlantılar üzerindeki akışlara karar verilerek ağ yapısının tasarlanması amacı güdülmektedir. Bu tezde, sıralı karar verme süreçlerini temel alan iki farklı iki seviyeli ATP üzerinde çalışılmıştır. Önerilen ilk iki seviyeli ATP'de, stratejik uçuş ağı tasarımı problemi, yerleşik bir rakibin edimsiz saldırıları sonucunda küçük bir havayolu şirketinin noktadan noktaya ağ yapılı uçuş ağında meydana gelebilecek olası en ciddi aksaklığı incelemek amacıyla bir ağ saldırılı problem olarak modellenmiştir. Bu problem, $r$-Saldırılı ve Talep Kayıplı Ağ Tasarım Problemi (RSTK-ATP) olarak adlandırılmaktadır. Mülteci Kampları Ağının Yeniden Kurulumu için İki Seviyeli Optimizasyon Problemi (MKYK-İSO) olarak adlandırılan ikinci problemde, yeni mülteci akışları ve kamplara sağlanan kamusal hizmetlerde değişiklikler olması durumunda, mülteci kampları ağının yeniden yapılandırılması incelenmektedir. RSTK-ATP için kodlanan Tabu Arama (TA), sınır tabanlı budama ve regresyon tabanlı aday çözüm kümesi türetme izlekleri ile iyileştirilerek tüm komşu çözümlerin tek tek çözülmesinden kaynaklanan işlem yükü azaltılmış ve daha iyi bir çeşitlendirme sağlanmıştır. MKYK-İSO için de bir temel TA kodlanmış, ve bu kod geliştirilen bir uyarlamalı komşuluk seçme izleği ile iyileştirilmiştir. Ayrıca, bir Değişken Komşuluk Arama (DKA) matsezgiseli kodlanmış ve bu kod, karıştırma aşamasında elde edilen başlangıç çözümlerine *iyi* çözüm bileşenlerinin dahil edilmesini sağlayan birliktelik kuralları tabanlı bir izlek ile iyileştirilmiştir. Deneysel sonuçlar, önerilen çözüm yöntemlerinin olumlu katkısını göstermiştir.

# TABLE OF CONTENTS

# LIST OF FIGURES

# LIST OF TABLES

# LIST OF SYMBOLS

| | |
|---|---|
| $A$ | Set of arcs |
| $A_e$ | Amount of increase in the capacity of existing camp $e$ |
| $\bar{a}_e$ | Upper limit on the capacity increase |
| $C$ | Set of candidate locations |
| $\mathcal{C}(s)$ | A subset of $\mathcal{N}(s)$ |
| $c_{ij}^k$ | The cost of unit flow of commodity $k$ on arc $(i, j)$ |
| $c_h$ | Number of physicians located at hospital $h$ |
| $c_h'$ | Updated number of physicians located at hospital $h$ |
| $D(k)$ | The destination of commodity $k$ |
| $d^k$ | The demand of commodity $k$ |
| $d_s$ | Number of refugees emanating from source $s$ |
| $E$ | Set of existing refugee camps |
| $\mathcal{E}$ | Set of edges |
| $F_{sc}$ | Fraction of refugee flow originating at $s$ and destined at $c$ |
| $F_{se}$ | Fraction of refugee flow originating at $s$ and destined at $e$ |
| $f$ | An objective function |
| $\hat{f}$ | Regression model of $f$ |
| $\mathcal{G}$ | An undirected graph |
| $H$ | Set of hospitals |
| $h_{ij}$ | The fixed charge of installing a unit link on arc $(i, j)$ |
| $I$ | An interdiction pattern |
| $\mathcal{I}$ | Set of BOpt-RRC instances |
| $K$ | Set of commodities |
| $M$ | Sufficiently large number |
| $N$ | Set of nodes |
| $\mathcal{N}(s)$ | The set of neighboring solutions of $s$ |
| $O(k)$ | The origin of commodity $k$ |
| $p^k$ | The cost of unit loss of commodity $k$'s demand |

| | |
|---|---|
| $\bar{Q}_e$ | Current capacity of an existing refugee camp $e$ |
| $Q_e$ | Capacity of an existing refugee camp $e$ |
| $q_c$ | Capacity of candidate camp $c$ |
| $q_k^i$ | Action value of the neighborhood $k$ at iteration $i$ |
| $R_{eh}^+$ | An auxiliary variable indicating if $e$ is reassigned to $h$ |
| $R_{eh}^-$ | An auxiliary variable indicating if $e$ is no more served by $h$ |
| $r$ | Interdiction budget of the attacker |
| $r_k^i$ | Reward gained at iteration $i$ by the neighborhood $k$ |
| $S$ | Set of refugee sources |
| $s$ | A solution vector |
| $s_{best}$ | Best solution found |
| $s_\rho$ | A solution vector obtained by rule injection |
| $\bar{T}_e$ | Number of current refugees located at camp $e$ |
| $T_c$ | Total number of refugees located at candidate camp $c$ |
| $T_e$ | Total number of refugees located at existing camp $e$ |
| $T_{ij}$ | Binary variable indicating if arc $(i, j)$ is interdicted |
| $\mathbf{T}$ | The matrix of $T_{ij}$ values |
| $\mathcal{T}$ | The set of interdicted arcs $(i, j)$ |
| $\mathbf{U}$ | The vector of $U^k$ values |
| $U_{ch}$ | Linearization variable for candidate camp $c$ and hospital $h$ |
| $U_{eh}$ | Linearization variable for existing camp $e$ and hospital $h$ |
| $U^k$ | Binary variable indicating if $k$'s demand is lost |
| $\bar{U}^k$ | Optimal values of $U^k$ |
| $\mathcal{V}$ | Set of vertices |
| $w$ | The capacity of a unit link installed on an arc |
| $\bar{X}_{eh}$ | Indicator value for hospital/camp assignment |
| $X_{ch}$ | Binary variable indicating if camp $c$ is assigned to hospital $h$ |
| $X_{eh}$ | Binary variable indicating if camp $e$ is assigned to hospital $h$ |
| $X_{ij}^k$ | The fraction of $d^k$ that flows on arc $(i, j)$ |
| $\bar{X}_{ij}^k$ | Optimal values of $X_{ij}^k$ |
| $\mathbf{X^R}$ | The matrix of $X_{ij}^k$ values |

| | |
|---|---|
| $Y_c$ | Binary variable indicating if candidate camp $c$ is built |
| $Y_{ij}$ | The number of unit links installed on arc $(i, j)$ |
| $\bar{Y}_{ij}$ | Optimal values of $Y_{ij}$ |
| $\mathbf{Y^R}$ | The matrix of $Y_{ij}$ values |
| $z$ | Optimal objective function value of a network design |
| $z^*$ | Optimal objective function value of a bilevel program |
| | |
| $\alpha$ | Linear coefficient of the reward function |
| $\alpha_e$ | Unit cost of capacity increase |
| $\beta$ | Coefficient as the power of the reward function |
| $\Gamma$ | Set of good solutions |
| $\gamma$ | Vector of a good solution |
| $\gamma_{min}$ | Minimum allowed size for the set of good solutions |
| $\Delta$ | Step size |
| $\delta$ | Increment size |
| $\zeta$ | Arc density |
| $\eta$ | Perturbation length |
| $\theta$ | Constant value for the cooling schedule |
| $\kappa$ | Coefficient for neighborhood definition |
| $\Lambda$ | Set of visited solutions |
| $\lambda$ | Adaptation rate |
| $\mu$ | A set of interdiction patterns |
| $\nu$ | Fixed neighborhood size |
| $\xi$ | Cost of establishing candidate camp $c$ |
| $\pi$ | Fraction of neighbors selected |
| $\rho$ | An association rule |
| $\rho_{eh}$ | Penalty of reassigning existing camp $e$ to $h$ |
| $\sigma$ | Minimum support parameter of Apriori algorithm |
| $\sigma_{max}$ | Maximum allowed value for the minimum support |
| $\sigma_{min}$ | Minimum allowed value for the minimum support |
| $\tau$ | Temperature |

| | |
|---|---|
| $\upsilon_c$ | Penalty of building candidate camp $c$ |
| $\phi_{ce}$ | Total cost of refugee flow from source $s$ to candidate camp $c$ |
| $\phi_{se}$ | Total cost of refugee flow from source $s$ to existing camp $e$ |
| $\chi_{ch}$ | Cost of assignment of candidate camp $c$ to hospital $h$ |
| $\chi_{eh}$ | Cost of assignment of existing camp $e$ to hospital $h$ |
| $\psi$ | Maximum number of refugees a physician can take care of |
| $\omega$ | Number of existing refugees |
| $\omega_{new}$ | Average ratio of the number of new refugees to the number of existing refugees |

# LIST OF ACRONYMS/ABBREVIATIONS

| | |
|---|---|
| ANS | Adaptive Neighborhood Selection |
| ARBI | Association Rule Based Injection |
| AS | Ant Systems |
| B&B | Branch-and-Bound |
| B&BC | Branch-and-Benders-Cut |
| BOpt-RRC | Bilevel Optimization Model for the Reconfiguration of Refugee Camp Network |
| BLS | Basic Local Search |
| BMIP | Bilevel Mixed Integer Programming |
| BP | Bilevel Programming |
| BS | Best Solutions |
| COP | Combinatorial Optimization Problem |
| CV | Cross Validation |
| DC | Distribution Center |
| DDS | Data-Driven Sorting |
| DDP | Data-Driven Perturbation |
| DT | Decision Tree |
| EA | Evolutionary Algorithms |
| GA | Genetic Algorithms |
| ILS | Iterated Local Search |
| IP | Interdiction Patterns |
| KKT | Karush-Kuhn-Tucker |
| L-LRP | Location-Location Routing Problem |
| LB | Lower Bound |
| LLP | Lower-Level Problem |
| LS | Local Search |
| MILP | Mixed Integer Linear Programming |
| MINLP | Mixed Integer Nonlinear Programming |

| | |
|---|---|
| ML | Machine Learning |
| MP | Master Problem |
| MOO | Multi-Objective Optimization |
| NDP | Network Design Problem |
| NDPLD | Network Design Problem with Lost Demand |
| NSGA-II | Non-Dominated Sorting Genetic Algorithm-II |
| OR | Operations Research |
| OS | Optimal Solutions |
| PD | Percent Deviation |
| PSO | Particle Swarm Optimization |
| RCND | Refugee Camp Network Design |
| RD | Restart Diversification |
| RF | Random Forests |
| RI-NDPLD | r-Interdiction Network Design Problem |
| RL | Reinforcement Learning |
| RSM | Response Surface Method |
| SA | Simulated Annealing |
| SP | Subproblem |
| SVM | Support Vector Machines |
| TS | Tabu Search |
| TSP | Travelling Salesman Problem |
| UB | Upper Bound |
| ULP | Upper-Level Problem |
| UNHCR | United Nations High Commissioner for Refugees |
| VNS | Variable Neighborhood Search |
| VRP | Vehicle Routing Problem |

# 1. INTRODUCTION

In the Network Design Problem (NDP), one deals with installing links between a set of given nodes of the network and determining the flow of a set of commodities over these installed links to minimize the total cost of design and flow decisions [1]. However, starting from the late 1970s, the term gains a more comprehensive nature in the progress of time and recently refers to the network optimization problems in which the nodes are also subjected to installing decisions. For instance, in the hub NDP [2], a hub-and-spoke configuration is designed in addition to the usual flow decisions and in the wireless NDP [3], location of mobile sinks on a given set of nodes are also decided. Regarding the current broader definition, the problem has a large number of application areas including, but not limited to, planning of airline and freight transportation, telecommunication, clean water supply, and wireless charging stations. An NDP formulation is proposed in [4] to model a set of strategic and tactical planning problems encountered in freight transportation, and an NDP variant with design-balance requirements is introduced in [5] and a mixed-integer linear programming (MILP) formulation to model transportation systems involving consolidation is devised. Besides freight transportation, the NDP is also used for designing wireless charging stations in the context of urban transportation networks [6]. The authors propose an MILP formulation for the problem at hand, and report a case study with urban network data from Chicago, IL. In addition to model business-related problems, the NDP is also used for humanitarian aid planning. Motivated by the rise in the demand for refugee camps, [7] devise a bi-objective NDP formulation to model the water distribution network design problem to assure the maximized accessibility to clean water supplies. In another work from the humanitarian studies, an NDP for refugee camps is proposed to model the decisions of locating refugee camps and the routing of public service providers to serve these camps [8].

As opposed to the studies in the classical single-level operations research (OR) literature, bilevel programming (BP) formulations are employed to model hierarchical

decision processes in which there exists two decision-makers and the decisions taken at a hierarchical level affect the decisions of the other level. The hierarchical structure of BP formulations makes them suitable candidates for modelling real-life situations in which there exists several hierarchical decision levels. One straightforward example is governmental policy-making, since a government agency has the power to dictate a policy to the system users. Once the policy is announced, the system users make their corresponding decisions. The interdiction models, as special cases of BP formulations known as Stackelberg games [9], are used to model the scenarios in which one of the decision-makers aims to attack the other decision-maker maliciously and in these type of models, objective function values of both levels are exactly the same with opposite directions. The interdiction models are used to model cases such as military operations or competitive markets. The readers are referred to Sections 2.1, 3.1 and 3.2 for the details of BPs, and the corresponding literature reviews on bilevel NDPs and interdiction problems.

In this thesis we introduce a couple of bilevel NDPs. In the first bilevel NDP we propose, a competition between two airline carriers (AC) is modelled as an interdiction problem. In the $r$-Interdiction Network Design Problem with Lost Demand (RI-NDPLD), a hypothetical small airline carrier (SAC) aims to enter the airline market in which there exists a set of incumbent ACs. The SAC uses the RI-NDPLD as a strategic flight network design tool to analyse the maximum possible disruption in its point-to-point (PTP) flight network in the wake of virtual attacks performed by a competitor. In the second bilevel NDP proposed here, namely the Bilevel Optimization Model for Reconfiguration of refugee camp network (BOpt-RRC), we propose a BMIP formulation to model the necessary updates on the current configuration of refugee camp network under new refugee flows. The motivations, definitions and formulations of both problems are thoroughly discussed in Chapters 4 and 5, respectively.

The NDP is an NP-hard problem [1]. Since, the NDP is employed in both RI-NDPLD and BOpt-RRC, these problems are also NP-hard. Although, there exists a set of exact solution methods for general BMIP formulations, it is not possible to opti-

mally solve the large-sized RI-NDPLD and BOpt-RRC instances with these methods. Therefore, we propose a set of data-driven local search matheuristics to solve these problems regarding new developments on ML embedded metaheuristics [10]. We implement a generic Tabu Search (TS) matheuristic to solve the RI-NDPLD. However, due to the excessive computational burden of individual objective function evaluations, we improve this implementation with a pruning based and a data-driven candidate list generation procedure. We also propose a data-driven perturbation procedure to devise a restart diversification scheme for the implementation. For the BOpt-RRC, we propose generic TS and Variable Neighborhood Search (VNS) matheuristics. Then, we improve the TS with a data-driven, value function based adaptive neighborhood selection procedure. We also improve the VNS implementation with a data-driven injection procedure to find promising initial solutions for the local search components of the VNS implementation.

The outline of the thesis is as follows. In Chapter 2, we discuss the basics of BP and the details of local search (LS) metaheuristics and ML models employed in this study. Chapter 3 is reserved for the reviews of bilevel NDP, bilevel NDPs in airline flight networks, interdiction problems and data-driven metaheuristics literature. The RI-NDPLD and its BMIP formulation as a network interdiction problem alongside with the solution methods are discussed in Chapter 4. The formal problem definition of the BOpt-RRC, the general BMIP formulation for the problem and the implementation of solution methods are given in Chapter 5. The computational abilities of both problems and a real-life case study of BOpt-RRC are analysed in Chapter 6. Lastly, the contributions of this thesis are summarized and the future work opportunities are discussed in Chapter 7.

# 2. PRELIMINARIES

## 2.1. Bilevel Programming

As opposed to the studies considered in the field of operations research (OR) that consider classical optimization models involving a single decision maker, BP formulations are utilized to model situations where two decision makers, called the leader and the follower, have conflicting objective functions, and the decision of each player influences the other player's decision. The interest in both modeling real-life situations as BP models and solving them exactly or heuristically has significantly increased within the last two decades [11]. This corresponds to a static Stackelberg game between two players [9]. The formulation defined by the expressions

$$\min_{x \in X, y} \quad \phi^u(x, y), \tag{2.1}$$

$$\text{s.t.} \quad f(x, y) \leq 0, \tag{2.2}$$

$$\min_{y \in Y} \quad \phi^l(x, y), \tag{2.3}$$

$$\text{s.t.} \quad g(x, y) \leq 0 \tag{2.4}$$

is a general representation of the BP problems. In this bilevel formulation, the leader's problem is called the upper-level problem (ULP) and given by the objective function (2.1) and constraints (2.2). The follower's problem is referred to as the lower-level problem (LLP) which consists of the objective function (2.3) and constraints (2.4). The variables $x \in X$ are the upper-level decision variables and controlled by the leader, while the variables $y \in Y$ are the lower-level decision variables that are determined by the follower. The functions $\phi^u(x, y)$, $\phi^l(x, y)$, $f(x, y)$, and $g(x, y)$ can be linear or nonlinear. If at least one set of decision variables are restricted to be integer and/or binary, then the bilevel program is called a BMIP.

In a subset of BP formulations, the leader (or the follower) aims to attack a set of resources of a follower (or a leader) to cause the maximum possible disruption in the

other player's operations. Moreover, the objective functions of the leader and follower turn out to be the same in value but opposite in terms of the sense of the optimization. Such BP problems are called *interdiction problems* [12] and the corresponding Stackelberg games are named as *attacker-defender games*. Interdiction models have been proposed in the literature in different contexts. The earliest examples [13, 14] are related to network interdiction problems where the shortest path problem and the maximum flow problem are extended in such a way that an attacker, as the leader of the bilevel model, interdicts (causes disruption in) the arcs and/or nodes of the underlying network with the objective of maximizing the length of the shortest path or minimizing the flow. The system planner, who is the follower in the bilevel model, solves the problems with an objective function opposite to that of the leader (i.e., minimizing the length of the shortest path or maximizing the flow) given that some nodes/arcs of the network are unusable. In fact, solving the interdiction problem with a virtual attacker provides insights about the resilience of the network and the system operator obtains valuable information about which nodes/arcs are critical in the network. Therefore, interdiction models have been instrumental for the analysis of resilience and vulnerability of critical infrastructure networks such as electricity transmission networks, transportation networks and supply chains. In this thesis, the introduced problems are modelled as BPs where the RI-NDPLD is a network interdiction problem and the BOpt-RRC is a general BP problem.

## 2.2. Local Search Heuristics

Finding optimal solutions for many kind of combinatorial optimization problems (COP) of practical size is intractable. Due to this computational challenge, various approximate solution methods are proposed in the COP literature comprising the exact optimality. On the one hand, in approximation algorithms as a subset of approximate methods, a certain bound on the optimality is guaranteed and this guarantee of certain bound has a computational price and necessitates a certain minimum computational effort [15]. On the other hand, heuristic algorithms as approximate methods usually find *good* solutions in a reasonable computational time [16]. This computational ad-

vantage of heuristic algorithms makes them preferable in practice and various heuristic algorithms are proposed for COPs.

There are two types of heuristics. In a problem-specific heuristic, solution procedures are devised for a certain problem, and they are not meant to solve other COPs. Metaheuristics, on the other hand, offer general frameworks (hence the prefix *meta*) to solve virtually all COPs with necessary modifications. In this sense, they offer a search strategy rather than ways of manipulating the inherent structure of a particular problem to find a solution. Regarding the classification in [16], metaheuristics are grouped into two classes as *single-solution* and *population* based methods. In single-solution based metaheuristics such as Tabu Search (TS), Simulated Annealing (SA) and Iterated Local Search (ILS), the quality of a single solution is improved iteratively, and in population based metaheuristics like Genetic Algorithms (GA), Particle Swarm Optimization (PSO) and Ant Systems (AS), a population of solutions are improved by generating new sets of populations iteratively.

In this thesis, we work on a set of single-solution based matheuristics to solve the RI-NDPLD and BOpt-RRC: various implementations of TS and Variable Neighborhood Search (VNS) heuristics. This section is dedicated to briefly describe the frameworks of the generic versions of these metaheuristics.

### 2.2.1. Tabu Search

Let $\mathcal{S}$ be the solution space of an instance of a COP, $s$ be a solution in $\mathcal{S}$ and $f(s)$ be the objective function of the COP to minimized. The neighborhood $\mathcal{N}(s)$ is defined as the set of all solutions can be reached from the solution $s$ with a predefined *move* such that $\mathcal{N}(s) \subseteq \mathcal{S}$. A basic local search (BLS) heuristic, starts with an initial solution $s^0 \in \mathcal{S}$, and iteratively improves this solution by changing the current solution with the best solution at the neighborhood of the current solution until finding a local optima such that $f(s^i) \leq \min_{s \in \mathcal{N}(s^i)} f(s)$. By deliberately exploiting a single basin of attraction to find a local optima, a BLS then, resides at the intensification extreme

of the exploration/exploitation dilemma. To recover such a myopic approach, various diversification procedures are devised and different single-solution based metaheuristics with these procedures are proposed in the literature. TS is introduced in 1989 as one of these single-solution based metaheuristics [17].

The main search mechanism behind TS is similar to that of BLS, i.e., steepest descent. However, it aims to escape from local optima by diversifying the search through unexplored regions by accepting *nonimproving* solutions as current solutions. At an iteration $i$, TS generates all the neighboring solutions at $\mathcal{N}(s)$. Then, it changes the current solution with $s'$ such that $f(s') = \min_{s \in \mathcal{N}(s^i)} f(s)$ although the condition $f(s') < f(s^i)$ in BLS does not hold. Selecting nonimproving solutions as current solutions may cause a cycling behaviour in which same solutions are visited repetitively. To prevent this, *tabu list* is used as a short-term memory to avoid *tabu* moves to be applied. Let $s'$ be a solution at the neighborhood $\mathcal{N}(s)$ and it is obtained by applying a tabu move on $s$. Then, $s'$ is not accepted as the current solution although it is the best solution in the neighborhood. The only exception to this scheme is that if $s'$ satisfies an *aspiration criteria*. Depending on the implementation, an aspiration criteria may be the condition that $s'$ is an improving solution or $f(s') < f(s_{best})$ holds where $s_{best}$ is the best solution found so far.

In addition to the short-term memory usage in TS, also the medium-term and the long-term memories are used for the sake of exploitation and exploration, respectively. The medium-term memory stores the characteristics of *good* solutions to encourage them in the generated solutions, i.e., it exploits the good solutions. The long-term memory stores the attributes of all visited solutions to avoid the characteristics of mostly visited solutions to direct the search into unvisited basins of attraction, i.e., it assures the further exploration of the solution space. Regarding these definitions, a generic TS is outlined in Figure 2.1.

Since its introduction in the late 80s, various TS implementations are proposed to solve various COPs such as travelling salesman problem (TSP) [18, 19], vehicle routing

problem (VRP) [20,21], NDP [22,23], network interdiction problems [24–26] and many more.

**Input:** Initial solution $s_0$
$s \leftarrow s_0$ ▷ Initial solution
**while** stopping_criterion is not satisfied **do**
  Generate $\mathcal{N}(s)$ ▷ Generate all the neighbors of the current solution
  Evaluate all $s' \in \mathcal{N}(s)$ ▷ Evaluate every neighboring solution
  $s \leftarrow \arg\max_{s' \in \mathcal{N}(s)} f(s')$ ▷ Accept the best non tabu or aspired neighbor.
  Update short, medium and long-term memories
  **if** intensification_criterion holds **then**
    Intensification
  **end if**
  **if** diversification_criterion holds **then**
    Diversification
  **end if**
**end while**
**Output:** Best solution found so far.

Figure 2.1. A generic Tabu Search template.

### 2.2.2. Variable Neighborhood Search

In single-solution based metaheuristics such as BLS, TS and SA, geographical metaphors are used for defining the solution space of a problem. From this perspective, a general local search (LS) seems analogous to devising a trajectory through a landscape to find a stationary point, i.e., the lowest point of a valley (or basin) for a minimization problem. The structure of a landscape directly depends on the problem, the instance, and the neighborhood definition. Let $\mathcal{N}^1$ and $\mathcal{N}^2$ be two different neighborhood definitions to be employed in a general LS for the same instance of a particular problem. Then, $\mathcal{N}^1$ and $\mathcal{N}^2$ define two different landscapes, and a local optima $s^* \in \mathcal{N}^1$ is not necessarily a local optima in $\mathcal{N}^2$, and vice versa. Regarding this observation, VNS metaheuristic first proposed in [27], uses a set of variable neighborhood definitions within a single-solution based metaheuristic, i.e., a set of neighborhoods are systematically explored in the hope of finding better local optimal solutions.

The VNS necessitates a general LS procedure (BLS, TS, SA, etc.) and a set of neighborhood definitions $\mathcal{N}^k$ such that $k = 1, 2, ..., k_{max}$ for *shaking*. Although there is not a restriction on the set of neighborhood definitions and their orders, the usual approach is to use nested neighborhoods and and the corresponding order from the simplest to the most complex one [16, 28], i.e., $\mathcal{N}^1 \subset \mathcal{N}^2 \subset ... \subset \mathcal{N}^{k_{max}}$. The general template of the basic VNS is outlined in Figure 2.2.

---

**Input:** $s_0$, $k_{max}$.

1: $s \leftarrow s_0$ ▷ Initial solution
2: **while** stopping_criterion is not satisfied **do**
3: $k \leftarrow 1$
4: **while** $k \leq k_{max}$ **do**
5: $s' \leftarrow$ pick a random neighboring solution from $\mathcal{N}^k(s)$ ▷ Shaking
6: $s'' \leftarrow LocalSearch(s')$ ▷ Apply local search
7: **if** $f(s'') < f(s)$ **then**
8: $s \leftarrow s''$
9: $k \leftarrow 1$
10: **else**
11: $k \leftarrow k + 1$
12: **end if**
13: **end while**
14: **end while**

**Output:** Best solution found.

---

Figure 2.2. Basic Variable Neighborhood Search.

The algorithm is initialized with $s^0$ as the current solution $s$ and $k = 1$. Posterior to picking a random neighboring solution $s'$ from $\mathcal{N}^k(s)$, $s'$ is fed into the LS heuristic as the initial solution to find the corresponding local optima $s''$ in line 6. If the condition $f(s'') < f(s)$ in line 7 holds, $s''$ is accepted as the current solution $s$ and the heuristic returns to the first neighborhood, i.e., $k = 1$. Otherwise, $s$ is not updated and the value of $k$ is increased by 1 in line 11. This inner loop continues until $k > k_{max}$. If the stopping criterion in line 2 do not hold, the inner loop is initialized again with the current $s$ and $k = 1$.

The VNS and its variants are proposed for solving various COPs in the literature such as the recent studies on berth-allocation and quay crane assignment [29], knapsack problem [30], machine scheduling [31], NDP [32], bilevel NDP [33], network interdiction [34, 35] and many more.

## 2.3. Machine Learning

Machine learning (ML) is broadly defined as optimizing a set of performance criteria through sample or historical data to devise a *predictive/descriptive* model, or both [36]. The ML paradigms are classified into *supervised*, *unsupervised* and *reinforcement* learning (RL) [37]. In supervised learning, a data with readily available labels for each data entry is used to train an ML model for extrapolation purposes. In the classification problem as a supervised learning method, features of a set of entries (i.e., inputs) and the corresponding classes (i.e., labels or outputs) are fed into an ML model for training. Then, the trained model handles the prediction of classes of future data entries. On the other hand, the aim of unsupervised learning models is to extract *hidden* patterns in data with unlabelled (or partially labelled) entries. In clustering, for instance, a set of data entries without labels/outputs are grouped into $k$ clusters with respect to the similarities/disparities between the data entries. Note that there is no prior knowledge on the characteristics of clusters and their members. This knowledge (i.e., hidden patterns) is extracted from the data.

In RL as the third paradigm, a learning agent interacts with an environment through its actions and aims to maximize the total reward collected from the environment. Take the example of a robot in a maze [36]. The robot is left at the entrance of the maze, and it is supposed to find the exit. In this setting, the maze is the environment and the robot as the agent aims to reach the exit, i.e., reward is defined as finding the exit. At its current position, the robot takes one of the four actions of going up, down, left or right. By taking these actions in a series of epochs and feeding the collected information into learning, the robot finds its way out of the maze.

In this thesis, we embed various ML models into a set of single-solution based metaheuristics to devise data-driven LS matheuristics to solve the RI-NDPLD and the BOpt-RRC. This section is dedicated to introduce the preliminary knowledge on these ML models. Random forests (RF) as a supervised ML model is presented in Section 2.3.1. Clustering and the association rules as unsupervised models are briefly described in Sections 2.3.2 and 2.3.3.

### 2.3.1. Random Forests

A decision tree (DT) as an ML tool, uses a divide-and-conquer strategy for classification and prediction tasks. It consists of *test* and *leaf* nodes [38]. At each test node a set of outcomes is computed based on the characteristics (i.e., inputs) of data entries, and each outcome represents a branch emanating from the test node and ends up at some other test node or a leaf node. A leaf node holds the label (i.e., output), and hence returns the predicted class/value. DTs are known to be interpretable models. However, other ML methods usually return better prediction accuracy compared to DTs. Regarding this observation the RF model in which a set of diverse DTs are built to yield better prediction accuracy by reducing the variance of the predictions in individual DTs is proposed in [39]. The diversity of the DTs assembled in an RF is at the heart of this approach and the diversity is ensured by exploiting bootstrap aggregation and random selection of a subset of predictors at each split point in the trees. The disadvantage of using an RF model over a DT model is usually the computational time and less interpretability. However, these disadvantages are usually negligible in practice.

### 2.3.2. Clustering

One of the problems encountered in ML literature is the clustering problem. Let $\mathcal{D} = \{d_1, d_2, ...d_{|\mathcal{D}|}\}$ be a set of data entries without labels. The aim of the clustering problem in its simplest form is to partition the points in $\mathcal{D}$ into $k$ distinct clusters. In the $k$-means clustering algorithm initially proposed in [40], centers of the $k$ clusters

and the members of each cluster are computed in an iterative manner. In a solution to the algorithm, the total sum of squared distances between each point in $\mathcal{D}$ and the center of the corresponding clusters are locally minimized. In this sense, the found solution is not guaranteed to be the minimal solution. However, as discussed in [41], finding a minimal solution for larger $|\mathcal{D}|$ and $k$ values are impractical, and hence the $k$-means clustering algorithm is used in practice as it is. In this thesis, the algorithm is used for partitioning a set of visited solutions in an LS scheme with respect to their objective values.

### 2.3.3. Association Rules

Let $\mathcal{I}$ be the set of *items* such that $\mathcal{I} = \{i_1, i_2, ..., i_{|\mathcal{I}|}\}$, $\mathcal{D}$ be the set of *transactions* where each transaction $T \in \mathcal{D}$ corresponds to an *itemset* such that $T \subseteq \mathcal{I}$. ML models to find association rules aim to find the common items in *most* of the transactions where an association rule is a logical expression in the form of $i_m \implies i_n$ such that $i_m, i_n \in \mathcal{I}$ and $m \neq n$ holds [42]. This rule is true in $c\%$ of all transactions $T \in \mathcal{D}$ such that $i_m \in T$ and $c$ is known as the *confidence*. The *support* $s\%$ is defined as the ratio of the number of transactions containing both $i_m$ and $i_n$ to the number of all transactions $T \in \mathcal{D}$. The semantic version of these relations is as follows: In $s\%$ of all transactions, if $i_m$ is in a transaction $T$, then $i_n$ is also in $T$ with confidence $c\%$. The apriori algorithm proposed in [43], aims to find all of the association rules with given minimum confidence and minimum support values. The algorithm is usually referred in marketing context. In this thesis, we use the apriori algorithm to extract association rules to detect the common components in a set of visited solutions in an LS scheme.

# 3. LITERATURE REVIEW

## 3.1. Bilevel Network Design Problems

Initial BP problems date back to the seminal paper of H. von Stackelberg in which the leader-follower games are introduced and modelled [9]. Due to the inherent hierarchical decision process, various real-life applications are modelled as BP formulation in the fields such as military operations [13], social networks analysis [44, 45], renewable energy systems planning [46], electricity market [47], agriculture [48], marketing [49,50] and many more. In this section, we review the BP studies in which the ULPs or the LLPs are the variants of the NDP. For general reviews on BP problems readers are referred to the review papers [11, 51, 52].

In [53], a bilevel discrete NDP model is introduced for transportation network expansion problem and a branch-and-bound (B&B) method is proposed to solve the problem. In the ULP of the problem, the system planner makes the expansion decisions over an existing road network and the users at the LLP make routing decisions with respect to the expansions. The system planner aims to increase the total performance of the system even though the user equilibrium defined by the LLP reduces the efficiency of some individual users. Another BP formulation including a network equilibrium model as the LLP is introduced in [54]. The system planner as the decision-maker of the ULP, aims to optimize the bus frequencies to decrease the total travel time of all public transportation users in Dalian economic & technological development zone. The LLP of the introduced model is a variant of the traffic assignment problem and finds the user equilibrium as a multi-modal transport network equilibrium model. The authors proposed a column generation and a B&B method to solve the given formulation. In [55], maintenance planning problem of a road network is modelled as a dynamic discrete NDP and a BP formulation is proposed for the problem. The government agency at the ULP devises a maintenance plan for the road segments and the bridges and the system users at the LLP aims to optimize their routes respecting the currently

closed arcs due to the planned maintenance. In addition these studies on road networks, a railway NDP is studied in [56]. Following a similar manner with above BP studies, the government agency as the decision-maker in the ULP, aims to design a railway network by adding new lines and segments and the system users at the LLP optimizes their routes. For the other selected BP studies on transportation networks readers are referred to [57–59].

A BP formulation is introduced in [60] to model the hazmat transport NDP. The aim of the problem is to minimize the total hazmat risk in a particular region by forbidding transportation of hazardous materials in some set of road segments. The ULP, then, corresponds to this governmental decision, and the LLP finds the least cost routes for all hazmat carriers avoiding the closed links in the road network. The BP formulation is solved as a single-level model by adding the Karush-Kuhn-Tucker (KKT) optimality conditions of the LLP to the ULP. The introduced model is extended later in [61]. The introduced BP formulation, in a similar manner with the approach of [60], is transformed into a single-level mixed integer linear programming (MILP) and is solved with a multi-cut Benders decomposition method.

A BP formulation for the location of relief distribution centers (DC) in humanitarian logistics is introduced in [62]. An aid agency as the ULP decision-maker aims to locate a set of capacitated DCs to serve the demands of the beneficiaries by minimizing the total cost of establishment and the uncovered demand. The beneficiaries as the LLP decision-makers aim to choose a DC to be served. The authors proposed a set of exact solution methods to find the Pareto optimal solutions to this bi-objective BP formulation. As another study in the context of humanitarian logistics, the food aid modality selection problem is introduced in [63] and modelled as a BP formulation. As the application of the problem, an aid agency intends to design an aid program in a predetermined area, and the beneficiaries decide the combination of basic, tasty and temptation foods to receive with the provided aids. The authors work on a case study in Kenya's Garissa county to illustrate how the introduced problem can be used in practice.

### 3.1.1. Bilevel Programming in Airline Network Design

Above, we discuss the literature on general bilevel NDPs and the studies on BP models of humanitarian logistics. In this section, we review a set of BP studies with applications in the airline industry. In [64], a one-to-one competition between an incumbent airline company exploiting a hub-and-spoke network and an entrant exploiting a low cost point-to-point network is modelled as a BP formulation. The model explicitly considers the number of flights at each link and the prices to find the market shares of both companies. The authors employ a sensitivity analysis approach to solve the BP model. A similar study on the competition between two airline companies employing hub-and-spoke flight networks is reported in [65]. In the ULP of the proposed BP formulation, an existing airline company adjusts its network and a potential entrant designs its flight network in the LLP. Both companies aim profit maximization by increasing market share. A random case study based on the regional market in Taiwan and China is solved heuristically with a GA method.

In [66], the authors study the problem of airline flight scheduling under competition. In the proposed modelling framework, an airline company designs its flight schedule such that the total revenue and resource usage is maximized. The framework, then, explicitly considers the passenger demand and the competition in the market in which a simulation model is used for predicting the passenger behaviour. Regarding the passenger behaviour, another BP formulation for the airline industry is introduced in [67] as a general framework to analyse the effects of possible airline policies on passenger behaviour. In the ULP, the regulative decisions are made by a government agency as the market regulator and the LLP models the passengers' route choice behaviour under the implied policy. The authors consider a case study in which the allocation of international/domestic flights to an urban and a remote airport of the same city is analysed.

To the best of knowledge, the only interdiction study on the airline flight network design is reported in [68]. A BP formulation based on the $p$-hub location problem to

analyse the resiliency of a hub-and-spoke flight network is introduced in the study. The formulation is actually an interdiction problem in which the objective functions at each level are same but in the opposite direction. By using the formulation under a cardinality budget assumption (i.e., $r$-interdiction), the authors aim to find the worst-case scenario out of all possible interdiction patterns.

### 3.2. Interdiction Problems

In the literature, interdiction models have been developed in different research areas. One stream of research focuses on *facility interdiction* problems where the attacker targets facilities in a network that provide service to customers. Since this type of interdiction models is out of the scope of this thesis, we only mention the pioneering work of [69] which sets the basis of a large number of facility interdiction models for both median-type and coverage-type location problems studied later. Within the context of facility interdiction problems, we can also mention problems that involve interdiction of hub facilities that provide some sort of service for demand points over a hub-and-spoke network [70]. There also exist studies that consider the interdiction versions of some well-known problems in the field of OR. [71] and [72] focus on the binary *knapsack interdiction problem* where the attacker tries to determine (interdict) the items that cannot be selected by the defender so as to minimize the profit of the latter who solves the binary knapsack problem to maximize the same objective function. Interdiction models have been employed in graph theoretic problems as well. For example, [73] investigate the problem of edge removal from an undirected graph such that the size of a maximum clique is minimized. A linear binary program is proposed to solve this clique interdiction problem. [74] examine the *matching interdiction problem* where the attacker aims to minimize the weight of the maximum matching, while the defender solves the maximum matching problem.

Contrary to facility interdiction models, in *network interdiction* models the attacks are conducted to damage the arcs rather than the facilities in the network. The pioneer work in network interdiction models is by [13]. In this study, the maximum flow

between an origin and a destination node is minimized by removing a certain number of arcs from a network. [14] considers the same problem under a limited interdiction budget and a limited number of interdicted arcs constraint. The first studies on network interdiction usually focus on security applications. [75] propose two algorithms to find the worst case scenario in a capacitated communication network in terms of reduced arc capacities and increased arc costs under attack. [76] present an algorithm to find the optimal interdiction of a supply network under an attack of an opposing force where the optimal interdiction is defined as the interdiction scenario that minimizes the network flow capacity. [77] investigate the problem of finding critical arcs on a network where critical arcs are defined as the arcs that maximize the length of the shortest path if they are fully interdicted by an adversarial attack. The authors introduce a MILP formulation and propose a decomposition algorithm for its exact solution. [78] deal with a multi-commodity network flow problem, where the arc capacities can be completely or partially decreased. [79] examine a shortest path interdiction problem in which the attacker and the system operator do not share the same information on arc lengths. In addition to these earlier studies focusing on *arc* interdiction, [80] develop BP formulations to analyse the security of electric grids under terrorist threat. [81] generalize this terrorist threat problem and propose a BP framework for the problem. Other BP formulations for possible attacks on electricity distribution networks are suggested in [82], and [83].

There also exist studies on network fortification-interdiction problems whose number is more limited than pure interdiction models. [84] try to determine the best allocation of protection resources, where the attacker aims the unprotected nodes and arcs in a shortest path network. [85] formulate a three-level model to specify the components of an electricity distribution network should be protected. [86] address an operator-attacker-operator problem, in which the operator make fortification decisions for a subset of arcs in the first level and the attacker damages a subset of unprotected arcs in the second level. The operator then solves a TSP in the third level. There also exist few multi-period network fortification-interdiction models in the literature [87,88]. The interested reader can refer to the survey paper of [12] for network interdiction models.

### 3.3. Data-Driven Search Heuristics

Recently, there is an increasing interest in ML-assisted data-driven solution methods for the COPs and various data-driven approaches are proposed as exact and heuristic methods to solve these NP-hard problems by the OR community. In this thesis, we propose a set of data-driven heuristic methods to solve a couple of bilevel NDPs and hence we review the corresponding heuristics literature in this review. The data-driven exact approaches to COPs are out of scope of this study and the interested readers are referred to most recent review paper [89] and the references therein.

According to the taxonomy introduced in [10], the ML methods *"at the service of metaheuristics"* are classified into six groups: Algorithm selection, fitness evaluation, initialization, evolution, parameter setting and cooperation. Our proposed metaheuristic methods contribute to the literature of fitness evaluation and initialization classes. To this extent, the reviewed papers are limited to these two classes.

In the fitness evaluation studies (or surrogate-assisted search), the ML methods are used to predict the fitness values (i.e., objective function values) of given solutions to a problem. The reason behind using predicted fitness values in a metaheuristic implementation is twofold. First, calculating the exact fitness value can be computationally expensive and second, the fitness function cannot be represented in a closed form at all (e.g., quarter car design simulation). All metaheuristic methods need at least one solution to initialize the search and an initial solution can traditionally be obtained in a random, greedy or a hybrid way [16]. Recently, the ML models are proposed to find partial or complete initial solutions to start the metaheuristic search and the initialization class of ML-assisted data-driven search literature discusses such approaches.

Most studies of using surrogate models for fitness evaluation are on evolutionary algorithms (EA) due to the necessity of large numbers of objective function evaluations inherent to the population based metaheuristics. In [90], individuals in a population

are clustered into $k$ groups by the $k$-means clustering algorithm and only the individuals closest to the centers of each clusters are evaluated. By exploiting these actual objective function values, the fitness values of remaining individuals are predicted by using an artificial neural network. A similar clustering approach is used in [91] in multi-objective optimization (MOO) context for detecting individuals close to the Pareto frontier. In [92], a global surrogate model is proposed as an evolutionary MOO method. The authors use the support vector machines (SVM) to predict the actual objective function values of each individual. Then, regarding the predicted objective function values, the actual objective function values of non-promising individuals are not calculated to reduce the total computational time. Another study incorporating a SVM based surrogate-assisted EA is proposed in [93] to devise a computational tool for aerodynamic shape design. The authors particularly work on the initial training data and its effect on the quality of solutions. In addition to the data-driven search studies on EAs, a surrogate-assisted SA method is discussed in [94] in the MOO context. The authors employs a response surface method (RSM) to predict the objective function values of solutions at each SA iteration.

The ML methods embedded into metaheuristics can be trained in an offline or an online fashion. In offline learning, adequately large number of instances of the same problem are optimally solved and the optimal objective values alongside with instance properties are used to train a supervised learning model. Then, the supervised model is incorporated into a search method to solve a new instance of the same problem. Preprocessing of an offline method is time consuming, however, once the trained ML model is obtained, the computational time of solving a new instance is reduced. The offline methods are reported to work properly on the instances possessing similar features. An example of an offline learning approach on EAs can be found in [95]. The authors solve the combinatorial circuit design problem with a GA in which a set of promising individuals extracted from the historical data are added into the initial population alongside with a set of random individuals. On the contrary, the training data is collected during the solution of a problem in the online learning, i.e., the solution and the training (or periodical training) are held simultaneously and the ML model

is not expected to have knowledge on the other instances of the problem. In [96], the gene combinations of elite solutions in a GA implementation are extracted during the solution. Then, these gene combinations are injected into the individuals of the future populations as an initialization strategy. The authors reported promising results for the TSPLIB instances. In [91], the individuals in the population are clustered by the $k$-means clustering algorithm, and the most representative individuals of each clusters are fed into an ILS implementation for the intensification purposes in a non-dominated sorting GA-II (NSGA-II) framework to solve the hub-and-spoke NDP.

In an earlier study on data-driven initialization, the historical dispatching schedules prepared by field experts are subjected to decisions trees and the key scheduling components indicated by the trained DT are incorporated into the initial solutions of a set of dispatching heuristics [97]. In addition to the traditional approach of using historical dispatching rules, association rules are recently used for solving COPs. In the method proposed in [98], set of association rules extracted from historical solutions to job shop scheduling problems are used for generating the initial populations of GA and PSO implementations for the problem.

Most of the studies in the data-driven search literature are on EAs proposed for MOO, and the studies on single-solution based metaheuristics are scarce. In this thesis, we devise a set of TS and VNS matheuristics relying on some ML models for performance increase.

# 4. STRATEGIC FLIGHT NETWORK DESIGN

Most of the major airline passenger carrier companies utilize hub-and-spoke (HS) network topology to benefit from scale economies [99]. For instance, when the expansions of the Middle Eastern based Qatar Airways, Etihad Airways, and Emirates Airways (known as ME3) and USA based United Airlines are examined, the HS network topology is regarded as the main reason for the success of these airline carriers (ACs) [100, 101]. In HS networks, ACs establish a single or a few airports as their hub airports, and aim to interconnect spoke airports through these hubs [102]. On the contrary, point-to-point (PTP) network topology is preferred by those ACs whose main flight network strategy is to establish direct flights between a set of airports. PTP flight networks are generally inherited by low-cost airline companies such as USA based Southwest Airlines and Ireland based Ryanair. The reason behind the preference of establishing PTP flight networks and refusing the economic advantages of HS networks is *strategic positioning* [103]. From this point of view, using PTP networks is about *differentiating* the product and avoiding direct competition with the major ACs in the industry [104]. For a detailed discussion of this topic, readers are referred to the case study of Southwest Airlines' entry to the USA market analysed in [105].

Interdiction models can be used to represent the competition between large and small companies [50] including the one in the airline market in terms of flight network design [65]. Regarding this literature, as the first bilevel NDP we propose here, the $r$-Interdiction Network Design Problem with Lost Demand (RI-NDPLD) focuses on the strategic flight network design of a hypothetical small airline carrier (SAC) adopting the PTP approach which targets to enter an airline market with potential threats from incumbent carriers and aims to analyse the maximum possible disruption in its flight network in the wake of virtual attacks performed by a competitor. In the problem, the disruption is measured by the number of disallowed flights and the loss of the corresponding demand. A bilevel mixed integer program (BMIP) is formulated where the decision maker in the upper hierarchical level (i.e., the upper level problem (ULP)

in a bilevel setting) is the virtual attacker that interdicts some of the links in the flight network of the SAC. The SAC, on the other hand, is the decision maker in the lower hierarchical level (i.e., the lower level problem (LLP) in a bilevel setting) that solves the NDP to determine the best flight network given the information on which links are interdicted, i.e., are disallowed for flight.

## 4.1. Problem Definition and Mathematical Models

This section is dedicated to the formal definitions and a set of arc based network flow formulations for the NDP, NDPLD and RI-NDPLD. Alongside, an illustrative example for the NDP is also given.

### 4.1.1. Multi-Commodity Fixed-Charge Capacitated Network Design Problem

Consider a flight network consisting of airports and potential flights between these airports. In the context of our problem, the airports and potential flights represent nodes and arcs of the flight network, respectively. Also, consider a set of passenger groups representing commodities where each passenger group has an origin airport, a destination airport, and associated demand. The NDP deals with (i) determining whether or not to install a link (i.e., direct flight) on each arc, and (ii) deciding the flow of passenger itineraries over these links with the objective of minimizing a cost function incorporating the total cost of design and flow decisions [106]. For the sake to keep the connection with other types of network we will use the terms node, potential arc, link, and commodity in the sequel.

In terms of designing a capacitated network, the NDP takes a network with no capacity on its arcs as input, and returns as output a network with nonzero arc capacities by installing links over a subset of its arcs. An example of an input network consisting of five nodes and nine potential arcs is illustrated in the leftmost part of Figure 4.1 and a solution to the NDP for the input network is given in rightmost part

of Figure 4.1 with $w$ denoting the capacity of a unit link. As can be seen, links are established or installed on potential arcs $\{(1,2),(1,4),(2,1),(3,1),(4,3),(4,5),(5,4)\}$. They are highlighted as solid lines.

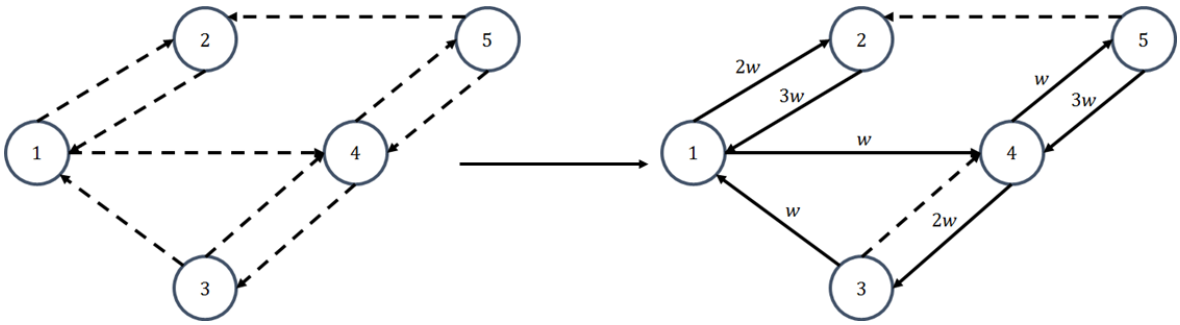


Figure 4.1. Illustration of the NDP.

In the literature, the NDP is modelled using three different formulations [107]: (i) arc based, (ii) path based, and (iii) tree based. In this study, we use the arc based formulation and obtain the model referred to as NDP which is adapted from [108]. Using the definitions of the sets, parameters, and variables given in Table 4.1 the formulation is given as

$$\min \sum_{(i,j)\in A} h_{ij}Y_{ij} + \sum_{k\in K}\sum_{(i,j)\in A} c_{ij}^k d^k X_{ij}^k, \tag{4.1}$$

$$\text{s.t.} \sum_{k\in K} d^k X_{ij}^k \leq wY_{ij} \qquad (i,j)\in A, \tag{4.2}$$

$$\sum_{j\in N\setminus\{i\}} X_{ij}^k - \sum_{j\in N\setminus\{i\}} X_{ji}^k = \begin{cases} 1 \text{ if } i = O(k) \\ 0 \text{ if } i \neq O(k), D(k) \\ -1 \text{ if } i = D(k) \end{cases} \qquad i\in N, k\in K, \tag{4.3}$$

$$0 \leq X_{ij}^k \leq 1 \qquad (i,j)\in A, k\in K, \tag{4.4}$$

$$Y_{ij} \in \mathbb{Z}^+ \cup \{0\} \qquad (i,j)\in A. \tag{4.5}$$

The objective function (4.1) aims to minimize the total cost consisting of two components. The first component represents the fixed cost of link installation, while the

Table 4.1. Sets, parameters and variables of NDP.

| Set | Description |
|---|---|
| $N$ | Set of nodes |
| $A$ | Set of arcs where each arc is represented as $(i, j)$ |
| $K$ | Set of commodities where each commodity $k \in K$ is defined as the triplet $\{Origin, Destination, Demand\}$. |

| Parameter | Description |
|---|---|
| $h_{ij}$ | The fixed charge of installing a unit link on arc $(i, j)$ |
| $c_{ij}^k$ | The cost of unit flow of commodity $k$ on arc $(i, j)$ |
| $d^k$ | The demand of commodity $k$. |
| $w$ | The capacity of a unit link installed on an arc. |

| Variable | Description |
|---|---|
| $X_{ij}^k$ | The fraction of the commodity demand $d^k$ that flows on arc $(i, j)$ |
| $Y_{ij}$ | The number of unit links installed on arc $(i, j)$ |

second one is the cost of commodity flows on installed links. Constraints (4.2) ensure that the flow on arc $(i, j)$ does not exceed the designed capacity of the arc. Constraints (4.3) are the *flow conservation constraints* and guarantee that each commodity flows from its origin to its destination. Constraints (4.4) and (4.5) are restrictions on the decision variables. It is worthwhile to emphasize that the integrality restriction on $Y$ variables allows the model to design a network whose arc capacities are multiples of $w$. Actually, this is what separates the NDP from its *uncapacitated* version [109].

### 4.1.2. Multi-Commodity Fixed-Charge Capacitated Network Design Problem with Lost Demand

In the general setting, the NDP assumes that there exists a decision maker who decides on both the installation of links and the flows of individual commodities over these links. However, in practice, the decision maker may choose not to satisfy the demand of some commodities and exclude them from the problem unless they are economically viable. In addition to this deliberate decision, the decision maker may sometimes be forced to forgo the demand (i.e., demand loss) due to possible disruptions that can occur in the network. For example, airline companies may loose customers because of competition or find themselves in a situation in which certain flights are canceled. To incorporate such decisions into the NDP, we introduce an extended version of the NDP referred to as the NDPLD, where the decision maker also deals with lost demand of a subset of commodities in addition to the design and flow decisions. The arc based MILP formulation of the NDPLD is given below. The set, parameter, and variable definitions of NDPLD are inherited from NDP with two additional definitions: (i) a binary variable $U^k$ that is equal to one if the demand of commodity $k$ is lost, and zero otherwise, and (ii) the parameter $p^k$ that represents the cost of unit loss of commodity $k$'s demand. Using these definitions, the NDPLD is modelled as

$$\min \sum_{(i,j)\in A} h_{ij}Y_{ij} + \sum_{k\in K}\sum_{(i,j)\in A} c_{ij}^k d^k X_{ij}^k + \sum_{k\in K} p^k d^k U^k, \tag{4.6}$$

$$\text{s.t.} \sum_{j\in N\setminus\{i\}} X_{ij}^k - \sum_{j\in N\setminus\{i\}} X_{ji}^k = \begin{cases} 1-U^k & \text{if } i = O(k) \\ 0 & \text{if } i \neq O(k), D(k) \\ U^k - 1 & \text{if } i = D(k) \end{cases} \quad i \in N, k \in K, \tag{4.7}$$

$$U^k \in \{0,1\} \qquad k \in K, \tag{4.8}$$

$$\text{Constraints } (4.2), (4.4), (4.5). \tag{4.9}$$

The objective function (4.6) aims to minimize the total cost of design and flow decisions as well as demand loss. Constraints (4.7) are the *modified* flow conservation

constraints which make sure that if demand of a commodity is lost, then NDPLD does not deal with the flow of that commodity. Mathematically speaking, if $U^m = 1$ such that $m \in K$, then the right-hand side of the corresponding constraint is equal to zero for origin, destination, and all transshipment nodes. Finally, constraints (4.8) ensure that $U^k$ variables take binary values.

### 4.1.3. The $r$-Interdiction Fixed-Charge Capacitated Network Design Problem with Lost Demand

The bilevel mathematical model RI-NDPLD developed in this study tries to answer an important question from the perspective of an airline company that wants to enter an already existing passenger air transport market: which direct flights to operate given that an established airline company in the market will not allow the market entrant to operate flights for some itineraries. To answer this question, a bilevel interdiction model is formulated to mimic a static Stackelberg game between two players in which the leader of the game is the virtual attacker (referred to as the attacker in the sequel) representing the established airline company and the follower is the market entrant company (referred to as the airline operator). The attacker determines a set of adversarial attacks on a subset of potential arcs $A$ to fully interdict the flows on these arcs (i.e., eradicate the airline operator from the market corresponding to the direct flight along the potential arc) provided that the number of attacks does not exceed the available budget. Given the interdicted arcs by the attacker, the airline operator aims to solve the NDPLD in the LLP of the RI-NDPLD. Regarding the additional set, parameter, and decision variable definitions are provided in Table 4.2 in addition to those included in Table 4.1, the BMIP formulation of the RI-NDPLD is given as

$$\max_{\mathbb{T}} \quad \sum_{(i,j)\in A} h_{ij} Y_{ij} + \sum_{k\in K} \sum_{(i,j)\in A} c_{ij}^k d^k X_{ij}^k + \sum_{k\in K} p^k d^k U^k, \tag{4.10}$$

$$\text{s.t.} \quad \sum_{(i,j)\in A} T_{ij} \leq r, \tag{4.11}$$

$$T_{ij} \in \{0,1\}, \quad (i,j) \in A, \tag{4.12}$$

$$\min_{\mathbb{X},\mathbb{Y},\mathbb{U}} \quad \sum_{(i,j)\in A} h_{ij}Y_{ij} + \sum_{k\in K}\sum_{(i,j)\in A} c_{ij}^k d^k X_{ij}^k + \sum_{k\in K} p^k d^k U^k, \tag{4.13}$$

$$\text{s.t.} \quad \sum_{k\in K} d^k X_{ij}^k \leq wY_{ij}, \quad (i,j)\in A, \tag{4.14}$$

$$\sum_{j\in N\setminus\{i\}} X_{ij}^k - \sum_{j\in N\setminus\{i\}} X_{ji}^k = \begin{cases} 1-U^k & \text{if } i = O(k) \\ 0 & \text{if } i \neq O(k), D(k) \\ U^k - 1 & \text{if } i = D(k) \end{cases}, \quad i\in N, k\in K, \tag{4.15}$$

$$Y_{ij} \leq M_{ij}(1-T_{ij}), \quad (i,j)\in A, \tag{4.16}$$

$$0 \leq X_{ij}^k \leq 1, \quad (i,j)\in A, k\in K, \tag{4.17}$$

$$Y_{ij} \in \mathbb{Z}^+ \cup \{0\}, \quad (i,j)\in A, \tag{4.18}$$

$$U^k \in \{0,1\}, \quad k\in K, \tag{4.19}$$

where $\mathbb{T}$ is the matrix of interdiction variables, $\mathbb{X}$ is the matrix of flow variables, $\mathbb{Y}$ is the matrix of design variables and $\mathbb{U}$ is the vector of lost demand variables.

Table 4.2. Sets, parameters and variables of RI-NDPLD-AB.

| Parameter | Description |
|---|---|
| $p^k$ | The cost of unit loss of commodity $k$'s demand |
| $r$ | The maximum number of arcs that can be interdicted by the attacker |
| $M$ | A sufficiently large number |

| Variable | Description |
|---|---|
| $U^k$ | Binary variable which is equal to one if the demand of commodity $k$ is lost, zero otherwise |
| $T_{ij}$ | Binary interdiction variable which is equal to one if arc $(i,j)$ is interdicted by the attacker, zero otherwise |

In RI-NDPLD, the ULP consists of expressions (4.10)–(4.12) while the LLP is represented by expressions (4.13)–(4.19). The objective function of the attacker given in

(4.10) is equivalent to the objective function (4.13) of the airline operator with different sense of optimization. It consists of the total cost of link installation, commodity flow, and lost demand decisions. Constraint (4.11) ensures that the total number of interdictions cannot exceed the threshold value $r$. This constraint makes RI-NDPLD an interdiction problem with a *cardinality-constrained* budget, which is based on [69] where $r$-interdiction median problem is defined. In this seminal paper, the authors considered facility interdiction in a service network and formulated two models from an attacker's viewpoint given that there are $p$ existing facilities serving the customers. The objective is to maximize the demand-weighted total distance by attacking $r$ out of $p$ facilities where the customers of the disrupted facilities have to be reassigned to undamaged facilities to get service. All constraints of the LLP are the same as those existing in the NDPLD formulation with the exception of constraints (4.16). They ensure that if arc $(i, j)$ is interdicted by the attacker, the operator cannot install a link on that arc. The values of the big-$M$ parameters $M_{ij}$ in these constraints are determined by computing the largest amount of flow that can pass over each arc $(i, j)$ based on the commodity demand values $d_k$.

## 4.2. Solution Methods

Reformulating bilevel programs as single-level programs is a typical strategy for optimally solving them. To do so, the KKT optimality conditions of the LLP are added to the ULP as a set of constraints [110]. However, such an approach requires that the LLP be convex [12]. Unfortunately, the LLP of the RI-NDPLD is an MIP, i.e., it is nonconvex, and we cannot exploit such an approach. Elaborated solution techniques such as branch-and-cut algorithm [111] and Benders-decomposition based methods [112] are needed to solve the BMIP formulation of the RI-NDPLD. Hence, we first implement a sampling-based exact (SBE) algorithm [113] to solve the RI-NDPLD. Let

$$\max_{x \in \mathbb{H}^x, y} \quad \phi^u(x, y), \tag{4.20}$$

$$\text{s.t.} \quad g_j^1(x) + h_j^1(y) \leq b_j^1 \qquad j = 1, 2, ..., m_1, \tag{4.21}$$

$$\max_{y \in \mathbb{H}^y} \quad \phi^l(x,y), \tag{4.22}$$

$$\text{s.t.} \quad g_j^2(x) + h_j^2(y) \leq b_j^2 \qquad j = 1,2,...,m_2 \tag{4.23}$$

be an BMIP formulation. Let also $\mathbb{X}(y) = \{x | g_j^1(x) \leq b_j^1 - h_j^1(y), j = 1,\ldots,m_1; x \in \mathbb{H}^x\}$ be the set of feasible solutions to the ULP for a fixed vector $y$ and $\mathbb{Y}(x) = \{y | h_j^2(y) \leq b_j^2 - g_j^2(x), j = 1,\ldots,m_2; y \in \mathbb{H}^y\}$ be the set of feasible solutions to the LLP for a fixed vector $x$. Define $\Omega = \{(x,y) | x \in \mathbb{X}(y), y \in \mathbb{Y}(x)\}$ as the region obtained by relaxing the optimality requirement of the LLP, and define $\Omega(\mathbb{X})$ as the projection of $\Omega$ onto ULP's decision space. Let $\Psi(x) = \arg\max\{\phi^l(x,y) | y \in \mathbb{Y}(x)\}$ be the lower-level problem's rational reaction set for a given vector $x$, i.e., the optimal solution (or set of alternative optimal solutions) of the LLP for a fixed $x$. Finally, define $\mathbb{Y} = \cup_{x \in \Omega(\mathbb{X})} \mathbb{Y}(x)$ be the set of all feasible LLP responses. The BMIP can be restated as

$$z^* = \max_{(x,y)} \{\phi^u(x,y) : x \in \mathbb{X}(y), y \in \Psi(x)\}. \tag{4.24}$$

A single-level problem obtained by relaxing the optimality requirement of the LLP, i.e., the high point problem (HPP), is given as

$$z^{HPP} = \max_{(x,y) \in \Omega} \{\phi^u(x,y)\}, \tag{4.25}$$

and an optimal solution to this problem returns a valid upper bound on $z^*$. In [113], authors formulate the so-called extended high point problem (EHPP) and prove that the EHPP is equivalent to the BMIP given in (4.20)–(4.23). The name of the formulation arises from the idea that the formulation is actually the HPP with an additional set of binary varaibles and a couple of additional set of constraints to ensure the bilevel feasibility. Define $\gamma_{\hat{y}j} = \lfloor b_j^2 - h_j^2(\hat{y}) \rfloor + 1$ for every $\hat{y} \in \mathbb{Y}, j = 1,\ldots,m_2$. Then, define $\mathbb{B}(\hat{y},\mathbb{Y}) = \{(y',q) | \gamma_{y'q} \geq \gamma_{\hat{y}q}, y' \in \mathbb{Y}, q = 1,\ldots,m_2\}$. Also, let $w_{\hat{y}j}$ be a binary variable equals to one if a constraint $j$ blocks solution $\hat{y}$; otherwise a solution may or may not be blocked. Next, let the Big-$M$ values be adequately large numbers. Then,

$$\max_{x,y} \quad \phi^u(x,y), \tag{4.26}$$

$$\text{s.t.} \quad g_j^2(x) \geq -M_j^1 + \sum_{\hat{y} \in \mathbb{Y}} (M_j^1 + \gamma_{\hat{y}j}) w_{\hat{y}j} \qquad j = 1,\ldots,m_2, \tag{4.27}$$

$$\phi^l(x,y) \geq \phi^l(x,\hat{y}) - M_{\hat{y}}^2 \sum_{(y',q)\in\mathbb{B}(\hat{y},\mathbb{Y})} w_{y'q} \qquad \hat{y} \in \mathbb{Y}, \qquad (4.28)$$

$$(x,y) \in \Omega, \qquad (4.29)$$

$$w_{\hat{y}j} \in \{0,1\} \qquad \hat{y} \in \mathbb{Y}, j = 1,\ldots,m_2 \qquad (4.30)$$

is the EHPP formulation. The objective function at (4.26) maximizes the objective function value of the ULP. Constraints (4.27) together with constraints (4.28) ensure the bilevel feasibility. Constraints (4.29) enforce the constraints of both levels. Last, constraints (4.30) are the binary restrictions on $w_{\hat{y}j}$ variables. Solving the EHPP requires the complete enumeration of all solutions to build the set $\mathbb{Y}$. Authors denote that the size of this set could grow exponentially. Therefore, they propose an iterative solution algorithm similar to a column generation procedure. To this end, the relaxed extended high point problem (REHPP) is defined as the EHPP with an exception on $\mathbb{Y}$. Instead of using all solutions, REHPP uses a restricted set of solutions $\hat{\mathbb{Y}}$.

The solution algorithm starts with an initial set $\hat{\mathbb{Y}}^0$. The REHPP with $\hat{\mathbb{Y}}^0$ is optimally solved, and an upper bound on the EHPP (i.e., the BMIP formulation) and an interdiction set is obtained. Given the set, response of the LLP is found by optimally solving the lower-level problem. A solution $\hat{y}^i$ and a lower bound on the EHPP (i.e., BMIP) are obtained, where $i$ is the index of an iteration. Next, restricted set of solutions is updated, i.e., $\hat{\mathbb{Y}}^{i+1} \leftarrow \hat{\mathbb{Y}}^i \cup \hat{y}^i$. The algorithm iterates in this fashion until the REHPP is found to be infeasible or lower and upper bounds are equal. If, the REHPP at an iteration $i$ is infeasible then the EHPP (hence BMIP) is also infeasible, and if the bounds are even then the optimal solution $z^*$ is equal to the lower/upper bound. For the proof of convergence and further discussions, readers are referred to the aforementioned study.

In this study, we first implement this exact algorithm to solve the RI-NDPLD. The implementation details of the algorithm are as follows:

- Initial Solution: The lower-level problem is optimally solved for a random set of $r$-interdictions, and a solution for $y_{ij}$ variables are obtained. Our implementation

starts with $\hat{\mathbb{Y}}$ including only this solution.

- Bilevel Feasibility: There are two issues to point out: (i) Constraints (4.27) of the EHPP refer to the LLP constraints including the ULP decision variables. In our implementation, constraints (4.16) of the RI-NDPLD satisfy this condition, and (ii) Since the RI-NDPLD is a max-min problem and the EHPP is a max-max formulation, we need to modify the constraints (4.28). To do so, we replace it with

$$\phi^f(x,y) \leq \phi^f(x,\hat{y}) + M^2_{\hat{y}} \sum_{(y',q)\in\mathbb{B}(\hat{y},\mathbb{Y})} w_{y'q} \qquad\qquad \hat{y} \in \mathbb{Y}.$$

Preliminary computational experiments on the implementation reveal that the gap between lower/upper bounds are significantly large at earlier iterations and a stagnating behavior is common for the RI-NDPLD. Moreover, the NP-hardness of the LLP and the necessity of LLP solutions at each iteration pose a challenge in terms of computational time. Actually, these issues would cause computational inefficiencies for most of the exact BMIP solution techniques. Therefore, we resort to matheuristic methods to solve the RI-NDPLD. To this end, we implement a generic TS matheuristic for the RI-NDPLD and propose a set of strategies to increase the efficiency of the implementation.

TS is a single-solution based metaheuristic. Since its introduction to the literature in the mid-eighties [114], it is used for solving various COPs including BMIPs. In [24], a matheuristic method incorporating TS and a MILP based exact solution technique to solve a leader-follower game involving facility location-protection-interdiction decisions are proposed . A trilevel $r$-interdiction median model for a facility location problem is introduced in [25] which is solved via a TS heuristic. In a more recent paper, a TS based matheuristic is employed for the $r$-interdiction selective multi-depot VRP [26]. Based on these successful implementations we first opt to develop a generic TS implementation as a matheuristic to solve the RI-NDPLD. Then, we introduce a set of valid inequalities for the LLP to have a stronger formulation and reduce the computational time to optimally solve the neighboring solutions. Then, the performance of the matheuristic

is improved by incorporating various candidate solution set generation procedures to reduce the number of exact solution calls for solving the LLP. Last, we propose a data-driven perturbation operator and devise a restart diversification scheme.

### 4.2.1. A Generic Tabu Search based Matheuristic

In this section, we introduce a generic TS matheuristic and briefly discuss its components. The pseudocode of the matheuristic is provided in Figure 4.2 and a flowchart of the algorithm is given in Figure 4.3. The details of the valid inequalities, candidate solution set generation procedures and the restart diversification scheme are discussed in the remaining sections. The main components of the TS matheuristic are as follows:

- Solution Representation: Each solution $s$ is represented with a $|A|$-bit string with each bit representing a link $(i, j) \in A$. The value of a bit is equal to one if the corresponding link is interdicted, and zero otherwise. For example, $s = [1, 0, 0, 1, 0]$ indicates that links 1 and 4 are interdicted, and the others are operational.
- Neighborhood Structure: The $SWAP$ operator used in the implementation removes the interdiction status of a link and creates disruption in a noninterdicted link, i.e., for $s = [1, 0, 0, 1, 0]$ $SWAP(s)$ generates a neighbor $s' = [0, 1, 0, 1, 0]$ by exchanging the interdiction status of link 1 and link 2.
- Neighborhood Evaluation: The objective value of a neighboring solution is computed by optimally solving the LLP of the RI-NDPLD using an off-the-shelf MILP solver. Further details are described in Section 4.2.2.
- Selection of the Neighbors: At each iteration, TS enlists all solutions in the neighborhood of a current solution $s$, i.e., $\mathcal{N}(s)$. Then, it selects a subset of the neighboring solutions to generate a candidate list of solutions $(s)$, i.e., $\mathcal{C}(s)$ such that $\mathcal{C}(s) \subseteq \mathcal{N}(s)$, and computes the objective value of each $s' \in \mathcal{C}(s)$. The best solution $s^*$ among all evaluated neighbors is chosen as the next solution. In this study, we devise two elimination procedures to remove some of the neighboring

solutions to generate candidate lists. In the section where we present computational results, we provide the analysis on the performance of these procedures.

- Initial Solution: A subset of all possible solutions is constructed systematically, and the best one among them is chosen as the initial solution. Details are given in Section 4.2.4.2.
- Tabu Structure: We use a hash list to avoid cycling.
- Termination Criterion: The algorithm terminates if a time limit is exceeded.

---

**Input:** Initial solution $s_0$, empty hash list of solutions

1: $s \leftarrow s_0$ ▷ Initial solution
2: **while** stopping_criterion is not satisfied **do**
3: Generate $\mathcal{C} \subset \mathcal{N}$ ▷ Generate the candidate list
4: Evaluate $s' \in \mathcal{C}(s)$ ▷ Evaluate every solution in the candidate list
5: $s^* \leftarrow \arg\max_{s' \in \mathcal{C}(s)} f(s')$ ▷ Pick the best neighbor
6: $s \leftarrow s^*$ ▷ Accept the best neighbor as the current solution
7: Update the hash list of solutions ▷ Add currently visited solutions
8: **end while**

**Output:** $s^*$

---

Figure 4.2. Tabu Search.

Let $s$ be the current solution, and $\mathcal{N}(s)$ be the neighborhood of the current solution. If the evaluation (i.e., objective value computation) of each neighbor $s'$ in $\mathcal{N}(s)$ requires a significant computational effort as is the case in solving the LLP of the RI-NDPLD, one may work on reducing the computational time of the evaluation and/or avoiding exhaustive search of the neighborhood $\mathcal{N}(s)$. Regarding this claim, we first devise a set of valid inequalities to reduce the computational time of the neighbor evaluation. To overcome the issue of exhaustive search, we introduce two strategies. First, we implement a pruning procedure to discard a subset of neighboring solutions without optimally computing their objective value. Second, we propose a data-driven neighbor sorting procedure. Both procedures help us to reduce the whole neighborhood $\mathcal{N}(s)$ to $\mathcal{C}(s) \subseteq \mathcal{N}(s)$, and optimally solve only those solutions that are in the candidate list $\mathcal{C}(s)$. We discuss the details of valid inequalities in Section 4.2.2. The details of

the the pruning procedure is discussed in Section 4.2.3, while the data-driven sorting procedure is explained in Section 4.2.4.1.

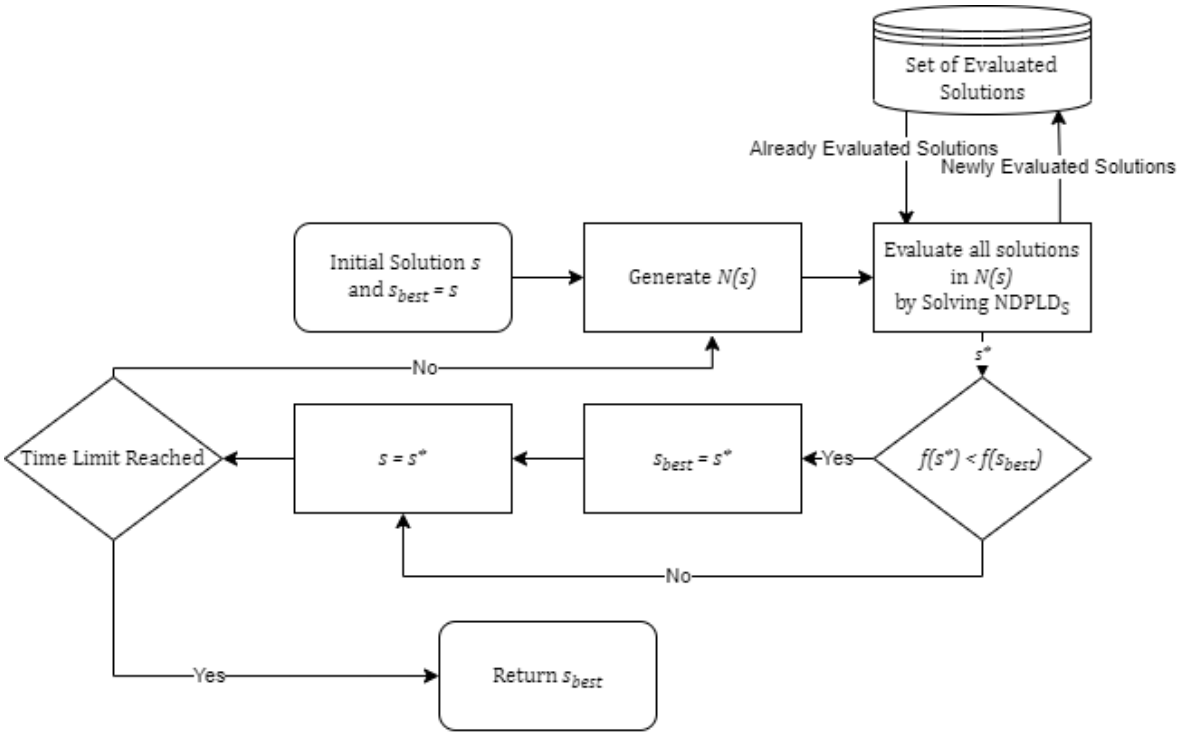


Figure 4.3. Flowchart of TS.

### 4.2.2. Evaluation of Neighboring Solutions

TS based matheuristics we propose in this study for the RI-NDPLD are iterative in nature and need to solve the LLP of the RI-NDPLD with given ULP variable values for neighbor evaluation. This fact, together with the NP-hardness of the LLP for given ULP variable values, makes the solution of the LLP, i.e., neighbor evaluation, the bottleneck operation of our implementation. In light of this observation, we here formulate two strategies of accelerating individual LLP solutions.

Let $T$ be the set of interdicted arcs determined by the values of the ULP variables. Due to the full interdiction assumption, the LLP of the RI-NDPLD is equivalent to the LLP where the constraints (4.16) are interchanged with $\sum_{(i,j)\in T} Y_{ij} = 0$. Instead of

making this transformation, it is possible to find an equivalent problem by defining the NDPLD over the network $G^T = (N, A \setminus T)$. We implement all matheuristic methods in a way that whenever a solution of an LLP with given ULP variable values is needed, it is obtained by solving the NDPLD over $G^T$. The rationale behind this decision is that the NDPLD has a similar structure with the NDP and there is a body of literature on the solution methods of the NDP starting back at early 70s [115]. To accelerate the solution of the LLP, we devise a branch-and-Benders-cut (B&BC) implementation in Section 4.2.2.1 and introduce a set of valid inequalities adapted from this literature in Section 4.2.2.2.

4.2.2.1. Branch-and-Benders-cut implementation. Inspired by the Benders decomposition literature on the NDP, we define the master problem with only design and lost demand variables, i.e., $Y$ and $U$. Let $\theta$ be a nonnegative continuous variable representing the objective value of the subproblem. Then, the master problem (MP) is defined as

$$\min \quad \sum_{(i,j)\in A} h_{ij} Y_{ij} + \sum_{k\in K} p^k d^k U^k + \theta, \tag{4.31}$$

$$\text{s.t.} \quad Y_{ij} \in \mathbb{Z}^+ \cup \{0\} \quad (i,j) \in A, \tag{4.32}$$

$$U^k \in \{0,1\} \quad k \in K, \tag{4.33}$$

$$\theta \geq 0. \tag{4.34}$$

The objective function of the master problem given in Expression (4.31) aims to minimize the sum of design cost, cost of lost demand, and $\theta$. The constraints of the formulation are reserved for the nonnegative integrality restriction on variables $\mathbf{Y}$, binary restriction on variables $\mathbf{U}$, and the nonnegativity restriction on $\theta$. Let $\{\hat{Y}_{ij} : (i,j) \in A\}$ and $\{\hat{U}_k : k \in K\}$ be the optimal values of the decision variables in the master problem. Then, the subproblem (SP) is obtained as

$$\min \quad \sum_{k\in K} \sum_{(i,j)\in A} c_{ij}^k d^k X_{ij}^k, \tag{4.35}$$

$$\text{s.t.} \quad \sum_j X_{ij}^k - \sum_j X_{ji}^k = \begin{cases} 1 - \hat{U}^k & \text{if } i = O(k) \\ 0 & \text{if } i \neq O(k), D(k) \\ \hat{U}^k - 1 & \text{if } i = D(k) \end{cases} \quad i \in I, k \in K, \tag{4.36}$$

$$\sum_{k \in K} d^k X_{ij}^k \leq w \hat{Y}_{ij} \quad (i,j) \in A, \tag{4.37}$$

$$0 \leq X_{ij}^k \leq 1 \quad (i,j) \in A, k \in K. \tag{4.38}$$

The objective function of SP aims to minimize the total cost of flows. Constraints (4.36) act as the flow conservation constraints if $\hat{U}^m = 0$ for a particular $m \in K$. Otherwise, together with the sense of optimization, it does not allow positive flows on corresponding arcs, as is the case in the NDPLD. Constraints (4.37) are the capacity constraints on arcs. The set of constraints (4.38) ensures that all flow variables take continuous values within $[0, 1]$.

In classical Benders decomposition implementations, the dual of the subproblem is solved to obtain the extreme points and extreme rays to generate Benders cuts. When this is the case, the feasible solution space of the subproblem remains the same whatever values are assigned to the MP variables, which allows an efficient implementation. In our case, we decide to solve the subproblem as is, i.e., the primal subproblem SP, with an off-the-shelf MIP solver. The reason behind this decision is threefold and related to the properties of the MIP solver used in the implementation. First, the SP is a multicommodity network flow problem and the MIP solver is able to solve it efficiently. Second, the MIP solver allows us to update the right-hand side of the SP easily and is able to exploit the solution of the previous subproblem as a warm start. Last, the extreme points (in case of optimality of the primal subproblem) and the extreme rays (in case of infeasibility of the primal subproblem) are directly reported along with optimal (or infeasibility status of the) solution to the SP.

A generic Benders decomposition implementation works in an iterative fashion. At each iteration, the MP is solved optimally and the solution $\{\hat{Y}_{ij} : (i,j) \in A\}$ and

$\{\hat{U}_k : k \in K\}$ is inserted into the SP. If the SP is solved optimally, then an optimality cut is added to the MP. If the SP is infeasible (or equivalently, the dual of the SP is unbounded), then a feasibility cut is added to the MP. The new MP is solved optimally from scratch and this fashion goes on until the termination. However, it is evident in the literature that such an implementation is inefficient and this is what we also face during preliminary experiments. One of the solutions to this inefficiency is to implement the algorithm on a single branch-and-cut tree to devise a B&BC method [116–118]. In the B&BC method, whenever an integer feasible solution is obtained in a node of the branch-and-bound tree of the MP, the SP is solved and a Benders cut (optimality or feasibility cut) is generated. Then, this cut is added to MP as a *lazy* constraint.

To speed up the Benders decomposition method, we also decide to implement the B&BC method. Computational results of our experiments on this method reveal that such an implementation is four times faster than the generic Benders decomposition framework on average. However, as another revelation of our experiments, the method returns poor optimality gaps within the time limit whenever an off-the-shelf MIP solver finds optimal solutions. To further accelerate the B&BC, we make use of the following observation: Benders cuts obtained by solving the subproblem are *low density cuts*, i.e., in a single Benders cut, a small portion of $\mathbf{Y}$ and $\mathbf{U}$ variables have nonzero coefficients. To overcome this issue we inherit the *covering cut bundle generation* mechanism of [119] and implement it in our B&BC method. Instead of a single Benders cut, the mechanism adds multiple cuts in an iteration. These multiple cuts are generated systematically to assure that a predefined portion of $\mathbf{Y}$ and $\mathbf{U}$ variables have nonzero coefficients in at least one of the cuts. Unfortunately, this effort yields no significant improvement in the efficiency, and we decide not to explore Benders decomposition for the NDPLD any further. Instead, we devise a set of valid inequalities to enhance the problem formulation to better exploit the abilities of the off-the-shelf MIP solvers.

4.2.2.2. A stronger NDPLD formulation. Let $\mathcal{T}$ be the set of interdicted arcs in the ULP. Then, the constraints (4.16) in the NDPLD for interdicted arcs $(i, j)$ with $T_{ij} = 1$ can be rewritten as $\sum_{(i,j)\in T} Y_{ij} = 0$. A better way is, however, to remove the interdicted

arcs altogether from the network to reduce the size of the NDPLD. We carry out this removal operation and also add valid inequalities to strengthen the formulation. Let $S \subset N$ be a nonempty subset of the nodes in $N$, and $\bar{S}$ the complement of $S$, i.e., $\bar{S} = N \setminus S$. Then, cut sets are defined as $(S, \bar{S}) = \{(i,j) \in A : i \in S, j \in \bar{S}\}$. Let $K(S, \bar{S})$ be the commodity subsets associated with $(S, \bar{S})$ such that $K(S, \bar{S}) = \{k \in K : O(k) \in S, D(k) \in \bar{S}\}$. Let us define $d_{(S,\bar{S})} = \sum_{k \in K(S,\bar{S})} d^k$ be the total flow of commodities demanded by the nodes in $\bar{S}$ from the nodes in $S$, i.e., commodities with a source node in $S$ and destination node in $\bar{S}$. A *cutset inequality* for a particular cut set $(S, \bar{S})$ is given with the constraint

$$\sum_{(i,j) \in (S,\bar{S})} w Y_{ij} \geq d_{(S,\bar{S})}. \tag{4.39}$$

and can be added to the NDP as a valid inequality [120]. However, this inequality is not valid for the NDPLD due to the variables $U^k$ that are used due to keeping track of the demand loss. Let $U^m = 1$ for a particular commodity $m$ such that $O(m) \in S$ and $D(m) \in \bar{S}$, and let $U^l = 0 : l \in K \setminus \{m\}$ in an NDPLD solution. Then, $X_{ij}^m = 0, (i,j) \in A$ holds. This indicates that the demand of commodity $m$ is lost and this commodity does not flow on any arc of the network. Hence, the total flow of commodities demanded by the nodes in $\bar{S}$ from the nodes in $S$ is equal to $\sum_{k \in K(S,\bar{S})} d^k - d^m$. By generalizing this observation, a modified version of the cutset inequality can be obtained by redefining $d_{(S,\bar{S})}$ as

$$d_{(S,\bar{S})} = \sum_{k \in K(S,\bar{S})} d^k (1 - U^k), \tag{4.40}$$

and we conclude that the inequality (4.39) with the right-hand side defined by (4.40) is a valid inequality for the NDPLD.

In addition to the modified cutset inequalities, we also devise a couple of other valid inequalities. Note that if the demand of a particular commodity $m$ is lost, then $\{X_{ij}^m = 0, (i,j) \in A\}$, as discussed before. In a similar vein, the inequalities

$$\sum_{(i,j) \in A : i = O(k)} X_{ij}^k + U^k \leq 1 \qquad k \in K, \tag{4.41}$$

$$\sum_{(j,i)\in A:i=D(k)} X_{ji}^{k} + U^{k} \leq 1 \qquad k \in K \tag{4.42}$$

can be written and these expressions are valid inequalities for the NDPLD.

To obtain a stronger formulation for the NDPLD, we directly add the valid inequalities (4.41) and (4.42) to the NDPLD formulation. However, it is a time consuming effort to enlist all $S$ and $\bar{S}$ sets and add the corresponding cutset inequalities to the NDPLD. Hence, we follow the common approach of choosing a subset of all cutsets such that $\{|S|, |\bar{S}|\} \in \{\{1,1\},\{1,2\},\{2,1\},\{2,2\}\}$. We call this stronger formulation $\text{NDPLD}_\text{S}$ and the formulation is optimally solved by an MILP solver whenever an exact solution of the LLP is needed as neighbor evaluation.

### 4.2.3. Pruning Procedure using Bounds

This procedure helps the TS heuristic to find at each iteration the best neighbor $s^*$ by pruning or eliminating some of the neighbors that can be proven to be not the best solution in $\mathcal{N}(s)$. This is made possible by using lower and upper bounds for the objective value of each neighbor. Let $s_{best}$ denote the incumbent solution for the bilevel problem, i.e., the best solution found throughout the iterations, and $f(s_{best})$ its objective function value. Note that the ULP is a maximization problem and thus $f(s_{best})$ provides a lower bound on the optimal objective value of the RI-NDPLD. Let $LB_{s'}$ and $UB_{s'}$ denote a lower bound and an upper bound on the optimal objective value of the neighboring solution $s'$ in the LLP, which is a minimization problem. The pruning procedure using bounds has two stage. The first stage involves the comparison of the lower bound $UB_{s'}$ of each neighbor $s'$ with the objective value $f(s_{best})$ of the incumbent solution. When $UB_{s'} < f(s_{best})$, then this neighbor cannot be better than the incumbent solution, and there is no need to optimally compute the objective value of this neighbor.

The second stage consists of comparisons between the lower bounds and upper bounds of different neighboring solutions. By recalling that the ULP has a maxi-

mization objective and the objective functions of the ULP and LLP are the same, $UB_{s'_i} < LB_{s'_j}$ implies that solution $s'_i$ can be eliminated from further consideration since its optimal objective value is smaller than that of solution $s'_j$. Using the pruning procedure, the number of the neighboring solutions can be reduced, which gives rise to a smaller computational effort required for optimally solving the LLP for all neighbors. This, in turn, increases the efficiency of the matheuristic.

An important issue to be explained is how to obtain $LB_{s'}$ and $UB_{s'}$ for the objective value $f(s')$ of each neighboring solution $s' \in \mathcal{N}(s)$. This is achieved by using linear programming (LP) relaxation of the LLP and a rounding heuristic. For a given interdiction pattern in the ULP of the RI-NDPLD, the LP relaxation of the LLP is obtained by relaxing the integrality restrictions on the decision variables $Y_{ij}$ and binary restrictions on decision variables $U^k$. The solution of the LP relaxation provides a lower bound. An upper bound is generated by converting the solution provided by the LP relaxation and transforming it to a feasible solution for the LLP by rounding up the fractional values of the decision variables $Y_{ij}$ to the nearest integer number. The objective value corresponding to the newly obtained solution is a valid upper bound because

i. If $\sum_{k \in K} d^k \bar{X}^k_{ij} \leq w\bar{Y}_{ij}$ holds, then $\sum_{k \in K} d^k \bar{X}^k_{ij} \leq w\lceil\bar{Y}_{ij}\rceil$ also holds since $\bar{Y}_{ij} \leq \lceil\bar{Y}_{ij}\rceil$ where $\bar{X}^k_{ij}$ and $\bar{Y}_{ij}$ are the optimal values of the decision variables in the LP relaxation.
ii. Due to the minimization objective of the LLP, the optimal solution of the LP relaxation allows either a positive flow or complete lost sales for each commodity. This means that the optimal values $\bar{U}^k$ are always integral and thus $\bar{U}^k = \lceil\bar{U}^k\rceil$.

The first TS based matheuristic, which is called TS/P (P stands for *P*runing), incorporates the pruning procedure using bounds into the basic TS matheuristic outlined in Figure 4.2 as a candidate list generation method with the goal of decreasing the computational time of the basic TS implementation. A flowchart for the matheuristic is given in Figure 4.4.

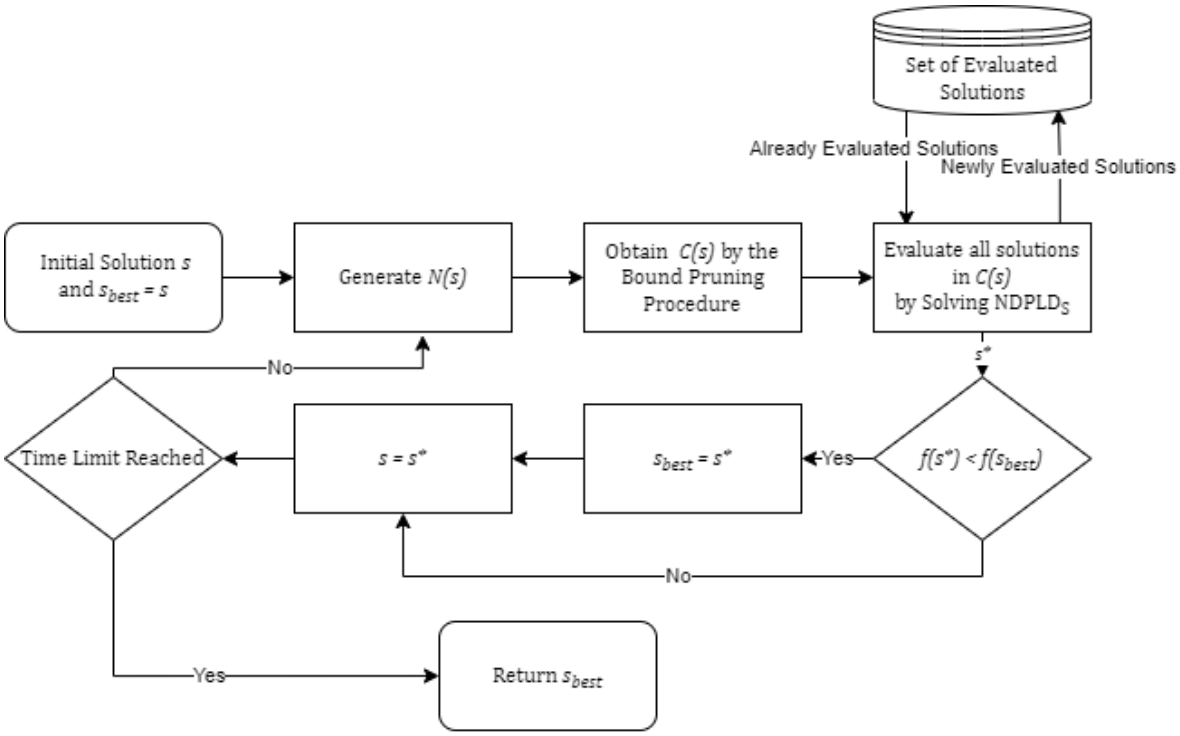


Figure 4.4. Flowchart of TS/P.

### 4.2.4. Data-Driven Procedures for Guiding the Search

4.2.4.1. Data-driven sorting and generation of candidate solution set. A basic TS implementation (a best improvement local search procedure as well) that performs an exhaustive search in the neighborhood of the current solution is destined to be inefficient for the RI-NDPLD due to the computational complexity of optimally solving the LLP. In such cases, a commonly adopted strategy is to randomly generate and evaluate a fraction of the existing solutions in the neighborhood. Unfortunately, random sampling (RS) of the neighborhood does not guarantee the selection of the best neighbor at each iteration and may deteriorate the intensification component of the search. To find a trade-off between these two neighbor selection procedures, i.e., exhaustive search and random sampling, we propose a *data-driven sorting* (DDS) procedure. In this procedure, we make use of a regression model $\hat{f}(s)$ that predicts the objective function values of *all* the neighboring solutions of the current solution. After sorting them in

nonincreasing order, a certain fraction of the solutions in the sorted list denoted by $\pi$ are selected, which form the list of candidate solutions. The LLP is solved to optimality only for these solutions. Obviously, as the predictive accuracy of the regression model becomes better, the probability of selecting the best neighbor increases.

---

**Input:** $s_0$, hash list of solutions

1: $s \leftarrow s_0$ ▷ Initial solution
2: Train $\hat{f}(s)$ ▷ With initial hash list of solutions
3: **while** stopping_criterion is not satisfied **do**
4: Generate $\mathcal{C}(s) \subset \mathcal{N}(s)$ by predicting obj. values using $\hat{f}(s)$
5: Evaluate $s' \in \mathcal{C}(s)$ by optimally solving the LLP ▷ Evaluate every solution in $\mathcal{C}(s)$
6: $s^* \leftarrow \arg\max_{s' \in \mathcal{C}(s)} f(s')$ ▷ Pick the best neighbor
7: $s \leftarrow s^*$ ▷ Accept the best neighbor as the current solution
8: Update the hash list of solutions ▷ Add currently visited solutions
9: **if** retraining_criterion holds **then**
10: Revise $\hat{f}(s)$ ▷ Retrain with updated hash list of solutions
11: **end if**
12: **end while**

**Output:** Best solution found.

---

Figure 4.5. Tabu Search with Data-Driven Neighbor Sorting.

The second TS based matheuristic in this thesis, referred to as the TS/DDS, uses the procedure of data-driven neighbor sorting as a candidate list strategy. Since the LLP is solved only a predefined fraction of the neighbors in terms of the decision variables of the ULP, we expect to save some computational time in the basic TS implementation. The details of TS/DDS is given in Figure 4.5 and the flowchart of the routine is given in Figure 4.6. As can be seen, the major deviation from TS/P is using a candidate list strategy based on DDS rather than pruning using bounds. As a matter of fact, both strategies can also be combined so as to obtain another version of the TS based matheuristic, which is called TS/P+DDS in the sequel. A flowchart for TS/P+DDS is given in Figure 4.7.

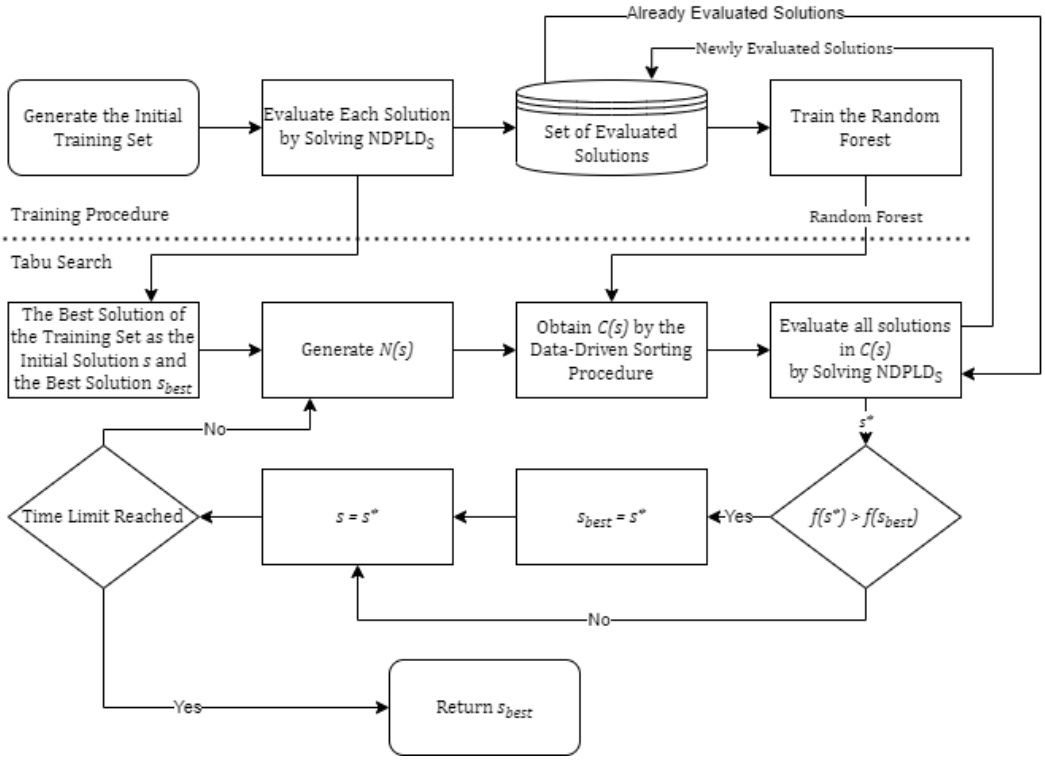


Figure 4.6. Flowchart of TS/DDS.

As mentioned earlier, an important component of TS/DDS is the regression model $\hat{f}(s)$ which is explained in the next subsection. Accurate predictions have the potential to discriminate the neighboring solutions in terms of their true objective value without solving their associated LLP. The latter is optimally solved only for the best $100 \times \pi$ neighbors that are selected based on the predicted objective values.

4.2.4.2. Building a regression model to predict objective values. As is the case for all predictive analytics models, there is a need for a training dataset consisting of observations (also called instances or records) with input attributes (also called features or predictors or independent variables), and an output attribute (also called outcome variable or response variable or dependent variable). In our case, each observation is a solution $s$ with the binary interdiction variables $T_{ij}$ of the ULP representing the input attributes and the optimal objective value of the LLP computed with the given values of the interdiction variables (interdiction pattern) representing the output attribute.

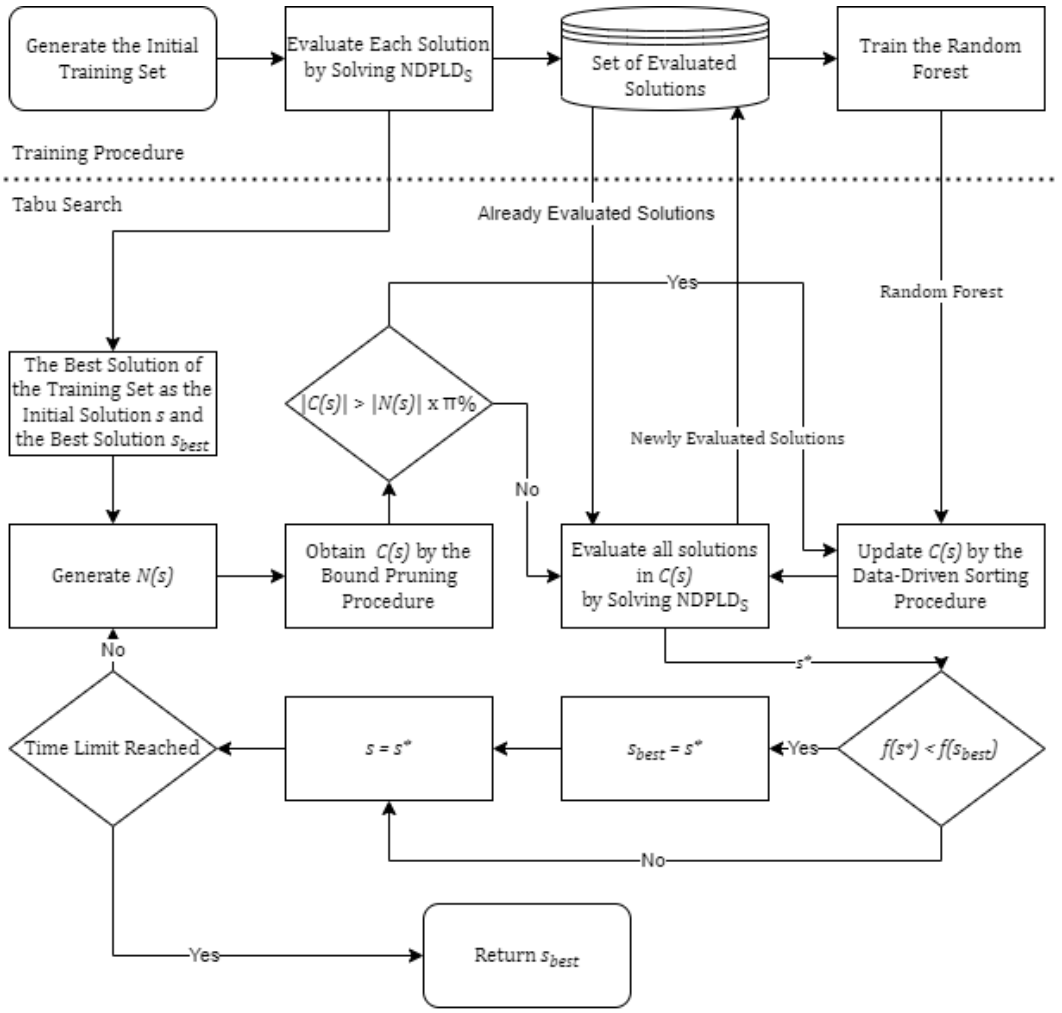


Figure 4.7. Flowchart of TS/P+DDS.

A sample training dataset for an RI-NDPLD instance with four arcs ($|A| = 4$) and two interdictions ($r = 2$) is given in Table 4.3. There are five solutions each of which corresponding to an observation, and the number of input attributes is equal to the number of interdiction variables which, in turn, is determined by the number of arcs.

The first observation represents interdiction pattern $s = [1, 0, 0, 1]$ with the optimal objective value $z_s^* = 60$ that is obtained by solving the LLP when the first and fourth arcs are interdicted by the leader. It is possible to add new features to the dataset to possible improve the accuracy of the regression model as can be seen in Table 4.4. We use a lower bound $LB_s$ given by the solution of the LP relaxation as

input attribute $I_5$ and an upper bound $UB_s$ obtained by the rounding heuristic as input attribute $I_6$ for a given interdiction pattern $s$.

Table 4.3. Sample dataset for training the regression model.

| | $I_1$ | $I_2$ | $I_3$ | $I_4$ | $O$ |
|---|---|---|---|---|---|
| $s_1$ | 1 | 0 | 0 | 1 | 60 |
| $s_2$ | 0 | 1 | 0 | 1 | 45 |
| $s_3$ | 1 | 0 | 1 | 0 | 40 |
| $s_4$ | 1 | 1 | 0 | 0 | 50 |
| $s_5$ | 0 | 0 | 1 | 1 | 65 |

In a recent review on embedding ML methods into combinatorial optimization [89], it is stated that ML methods can be trained in two ways. First, a universal ML model can be used to solve a particular problem with abundant data. To do so, a set of features are defined for a problem (e.g., TSP) and the features of many instances of the same problem with various sizes are extracted. The extracted features alongside with optimal solutions are fed to an ML model for training. Then, this ML model can be used to predict the objective values of new instances. Second, an ML model can be trained on-the-fly while solving a given instance of a particular problem. The framework is similar to the universal ML model. However, since the features are extracted from a given instance and the ML model is trained with these features, the corresponding ML can be used to predict the objective value of this particular instance. It is more likely that such an ML model is obsolete for other instances of the same problem. In TS/DDS, we adopt the latter idea and train the regression model in two phases. The first phase involves the creation of an initial training set. Recall that the optimal objective value is needed for each interdiction pattern as the output attribute to supervise the learning process. On the other hand, it is time-consuming to solve the LLP with a given interdiction pattern $s$, and it is inefficient to have many observations in the dataset. To make a compromise, the training dataset is initialized with a small number of interdiction patterns where each arc in the network is interdicted

at least once. Moreover, exactly $r$ interdicted arcs must exist in each pattern. Table 4.5 contains a sample initial training dataset for an RI-NDPLD instance $|A| = 6$ and $r = 3$.

Table 4.4. Sample data extended with lower/upper bounds for training.

| | $I_1$ | $I_2$ | $I_3$ | $I_4$ | $I_5$ | $I_6$ | $O$ |
|---|---|---|---|---|---|---|---|
| $s_1$ | 1 | 0 | 0 | 1 | 50 | 65 | 60 |
| $s_2$ | 0 | 1 | 0 | 1 | 30 | 60 | 45 |
| $s_3$ | 1 | 0 | 1 | 0 | 35 | 45 | 40 |
| $s_4$ | 1 | 1 | 0 | 0 | 40 | 70 | 50 |
| $s_5$ | 0 | 0 | 1 | 1 | 60 | 70 | 65 |

The procedure is outlined in Figure 4.8. Please note that the size of the initial training set is calculated as $|A| - r + 1$.

Table 4.5. A sample output of Algorithm 4.8.

| | $I_1$ | $I_2$ | $I_3$ | $I_4$ | $I_5$ | $I_6$ | $I_7$ | $I_8$ | $O$ |
|---|---|---|---|---|---|---|---|---|---|
| $s_1$ | 1 | 1 | 1 | 0 | 0 | 0 | 50 | 70 | 60 |
| $s_2$ | 0 | 1 | 1 | 1 | 0 | 0 | 40 | 55 | 50 |
| $s_3$ | 0 | 0 | 1 | 1 | 1 | 0 | 35 | 45 | 40 |
| $s_4$ | 0 | 0 | 0 | 1 | 1 | 1 | 50 | 80 | 70 |

Apart from obtaining an initial training dataset as the output of the procedure, it is also possible to select the solution with the highest objective value as the initial solution i.e., $s_0 \leftarrow \arg\max_{s_i}(z_{s_i})$. For all the matheuristic implementations (e.g.TS/P and TS/DDS) in this study, we use this strategy for the sake of comparison.

The size of the initial training dataset can be enlarged by adding new solutions that remain in the candidate list and have their optimal objective values computed by

solving their LLP. In order to utilize the growth of the dataset in superior regression models we incorporate *periodic retraining* into the TS/DDS. Newly visited solutions are stored in a hash list of solutions, and these solutions are used to fit a new regression model periodically. Please, see lines 9–11 of Figure4.5. By doing so, we expect the regression model to yield smaller prediction errors and the probability that the DDS procedure selects the best neighbor is increased.

---

**Input:** $r$ and $|A|$ ▷ Number of interdictions and number of arcs

1: $\mu \leftarrow \emptyset$ ▷ Initialize the set of interdiction patterns

2: $i \leftarrow 0$

3: **while** $i \leq |A| - r$ **do**

4: $\quad \mu \leftarrow \mu \cup \{s_i\}$ ▷ Add the next interdiction pattern

5: $\quad i \leftarrow i + 1$

6: **end while**

7: Initial Solution $s_0 \leftarrow \arg\max_{s_i}(z_{s_i})$

**Output:** Initial training set, and the initial solution for the matheuristics.

---

Figure 4.8. Building the Initial Training Set.

4.2.4.3. Using random forest as the regression model. An ML method has to be selected for the regression model explained in the previous subsections. Since most of the input attributes in the dataset (the interdiction variables) are binary-valued, we opt for using tree based regression methods. Following the tendency in the related literature [121], we conduct preliminary experiments by fitting regression models using DT and RF. Despite the fact that the RF models are less interpretable than DT models, the experimental results revealed that the RF model makes better predictions compared to the RT model, on average. Actually, this result is expected because the RF models are usually preferred as they can reduce the variance of the predictions in DT models by using many trees exploiting bootstrap aggregation and random selection of a subset of predictors at each split point in the trees. Therefore, they are known to be robust models [122]. The disadvantage of using an RF model over a DT model is usually the computational time. However, the difference between the computational efforts becomes insignificant for relatively larger RI-NDPLD instances considered in this

study. Thus, we decide to continue with the RF model in the DDS implementation without suffering from computational burden.

The third TS based matheuristic TS/P+DDS proposed in this study combines the pruning procedure using bounds and the data-driven neighbor sorting and selection procedure.

4.2.4.4. Restart diversification procedure. A local search algorithm performs search in a restricted region of the solution space to determine a local optimal solution. In addition to this intensification or exploitation process, there is a need for diversification or exploration procedures which direct the search into unexplored regions of the solution space to find better solutions. Different mechanisms are utilized in various metaheuristics for this purpose. A basic TS heuristic, for example, allows non-improving neighboring solutions to be visited and employs tabu moves that also prevents cycling. However, it is not always possible to escape a local optimal solution which represents a strong basin of attraction. A remedy that is used in the TS implementations to overcome this problem is the so-called long-term memory structure which is used to direct the search to unexplored regions of the search space to achieve diversification. There exist three major techniques applied in the literature [123]. The first one, *restart diversification* (RD), aims to force the inclusion of rarely visited components in the incumbent solutions obtained so far and restart the search from the best incumbent solution. In *continuous diversification*, a penalty term is added to the objective function to penalize frequently generated components of the solutions. The third one, *strategic oscillation*, allows the acceptance of infeasible solutions by using additional penalty terms in the objective function for infeasibility.

In Section 4.2.4.2, we mentioned that a regression model is fit at the beginning with an initial set of solutions, and it is periodically revised throughout the iterations with additional data obtained during the search. Since the model with the best predictive power is the one obtained most recently, it can be used to restart the search from the best solution obtained so far to explore the unvisited regions of the solution space

with more accurate neighbor sorting. Based on this observation, we propose here an RD procedure as a diversification scheme for the RI-NDPLD.

In the tree based regression models such as DT and RF methods, the importance of each feature/input attribute can be calculated. In the context of RI-NDPLD and the DDS, the importance of each feature (recall that each feature represents a directed arc) measures the effect of interdicting the corresponding arc on the objective function value of the LLP. Hence, feature importance measures can be used as a proxy for the long-term memory usage in TS matheuristics for the RI-NDPLD. In this study, we propose a data-driven perturbation (DDP) operator to be incorporated into the RD scheme described above. Suppose that an RI-NDPLD instance with $|A| = 6$ and $r = 3$. Let $s^* = [1, 0, 1, 0, 1, 0]$ be an incumbent solution, and the perturbation length is set to $\eta = 2$. The DDP operator perturbs $s^*$ in the following way. First, the operator removes the interdiction status of $\eta = 2$ interdicted arcs that have the most feature importance values. Then, it interdicts two $\eta = 2$ most important non-interdicted arcs. If $[0.20, 0.20, 0.15, 0.25, 0.05, 0.15]$ is a vector of importance values, the DDP removes the interdiction status of arcs 1 and 3, and interdicts arcs 2 and 4 to obtain a perturbed solution as $s' = [0, 1, 0, 1, 1, 0]$. In summary, the RD procedure starts by taking an incumbent solution $s^*$ found by TS/DDS given in Figure 4.5. Then, if diversification is invoked, $s^*$ is perturbed with the DDP operator to generate a new solution $s^{*\prime}$ from which search is restarted using TS/DDS.

# 5. RECONFIGURATION of REFUGEE CAMP NETWORKS

According to the "Global Trends: Forced Displacement in 2021" report published by United Nations High Commissioner for Refugees (UNHCR), 89.3 million people are forcibly displaced worldwide due to *"persecution, conflict, violence, human rights violations and events seriously disturbing public order"* [124]. This number is twice as high as that in 2012 and an increasing trend is expected for the following years. 69% of all current refugees in the world originate from Syria, Venezuela, Afghanistan, South Sudan and Myanmar, and 72% of the refugees are hosted by neighboring countries [124]. As a neighboring country of the Syrian Arab Republic, Turkey has the largest refugee population with 3.8 million refugees among all of the hosting countries, and 3.7 million of these refugees have Syrian origin [125].

When people in their home country are forced to leave, they are displaced and move to the host country as an *asylum seeker*. However, it is not possible to immediately gain a refugee status in the host country and an asylum seeker is located at refugee camps that provide short-term accommodation. If people do not have a chance to go back to their home country, they are settled in the host country by gaining a refugee status or resettled in a third country [126]. As the statistics given above indicate, Turkey as the neighbor of the Syrian Arab Republic, accepts Syrian asylum seekers, hosts them in the refugee camps, and then settles them in urban or suburban regions [127]. Currently, 50,736 Syrian refugees are located at eight refugee camps in the southeast region of Turkey [128].

Despite the fact that refugee camps are initially meant to be temporary *shelters* (or refugee warehouses [129]), the camps have become long-term accommodation places since 2008 due to the steady increase in the total number of refugees worldwide [130]. Such a paradigm shift in refugee camps leads in the planning approaches of refugee camps. In the current approach, the planning of public services offered in refugee camps

made based on long-term decisions [131]. Turkey is not an exception. For instance, Sarıçam refugee camp at Adana was established in 2013 with tents as houses. In 2017, the tents were replaced with *containers* as more durable housing options, and the camp is still hosting 16,575 refugees. As an OR study for refugee camp network design (RCND), a location-location routing problem (L-LRP) is introduced in [8] for locating refugee camps in the southeast region of Turkey considering long-term accommodation. The routing part of the study involves the planning of the routes of the service providers' periodical visits to the established camps .

In this study, we introduce the BOpt-RRC as an RCND problem for the reconfiguration of the current refugee camp network. Assume that a set of refugee camps is already established in a host country and public services provided in these camps are planned accordingly. Hence, there is a current refugee camp network configuration as a predetermined solution to an RCND problem. Also assume that new waves of refugees are expected in the host country in the future based on projections made by UNHCR. In the BOpt-RRC, we deal with minimum revision of the current configuration of the refugee camp network when assigning the new refugees into the existing camps. The reconfiguration decisions are based on increasing the capacities of these camps. However, it is not always possible to assign the new refugees to existing refugee camps by only increasing the capacities because of the practical upper limits on the capacity increase. In that case, the BOpt-RRC establishes new refugee camps in candidate locations to cover the total housing demand. In addition to the new refugees, the supply of the public service providers may evolve over time. Then, the BOpt-RRC also aims to update the public service provision plans to cover both the existing and the additional demand.

Reconfiguring an existing refugee camp network necessitates the cooperation of various government agencies. The tasks such as registration of new refugees, providing tents/containers and managing subcontractors for infrastructure construction are usually under the responsibility of different bodies. Hence, a set of hidden costs of cooperation also incur for the reconfiguration of the camp network in addition to the

foreseen expenses, e.g., cost of camp building and cost of capacity increase. It is possible to reduce the hidden costs by ensuring the minimum change in the configuration of an existing network. To this end, we formulate the BOpt-RRC as a bilevel optimization model to reconfigure a refugee camp network. In the LLP of the BOpt-RRC, the decisions building new refugee camps, the allocation of refugees to camps, and hospital assignments are determined by minimizing the corresponding cost components. A *virtual* coordinator as the decision maker of the ULP forces the smallest change in the network configuration by minimizing an objective function that consists of the sum of capacity increase costs and various penalty costs associated with changes.

The formal definition of the problem is given in Section 5.1 and a BMIP formulation is introduced in Section 5.2. The details of a set of generic and data-driven LS matheuristics proposed are given in Section 5.3.

## 5.1. Problem Definition

Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be an undirected graph whose vertex and edge sets are given as $\mathcal{V}$ and $\mathcal{E}$, respectively. $\mathcal{V}$ is given as the union of four type of vertices, i.e., $\mathcal{V} = E \cup C \cup S \cup H$. The set of vertices in $e \in E$ represents the existing refugee camps. The current capacity of an existing camp $e$ is given as $\bar{Q}_e$ and there are $\bar{T}_e$ refugees currently located in $e$. Medical services for existing refugees are covered by physicians located in hospitals $h \in H$, and each physician is allowed to take care of $\psi$ refugees at most. To satisfy this restriction, each existing camp $e \in E$ is currently assigned to one or more hospitals $h \in H$ with respect to the number of physicians, $c_h$, located at $h \in H$. This information is stored in the indicator variables $\bar{X}_{eh}$. However, in case of new refugee flow originating at sources $s \in S$, the decision maker has to readjust the current configuration of the built refugee camp network and hospital assignments.

Let $d_s$ be the number of new refugees originating at source $s$. Then, $\sum_{s \in S} d_s$ refugees must be allocated to existing refugee camps $E$ provided that the capacity restrictions are respected. However, if there is not adequate camp capacity, the de-

cision maker has to increase the total capacity of refugee camps either by increasing the capacity of existing camps up to an upper limit $\bar{a}_e$ or by building new camps at candidate locations defined by the set $C$. In addition to changes in the total number of refugees, the number of physicians at hospitals may also evolve over time. Let $c'_h$ be the current number of physicians at hospital $h$. Then, the BOpt-RRC is formally defined as follows. Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be an undirected graph such that $\mathcal{V} = E \cup C \cup S \cup H$, $E$ be the set of existing refugee camps, $C$ be the set of candidate locations for new refugee camps, $S$ be the sources of new refugee flows, and $H$ be the set of hospitals. The BOpt-RRC is defined as a bilevel optimization problem in which the ULP aims to readjust the current configuration of refugee camp network with smallest change by increasing the capacities of existing camps and the LLP deals with locating new refugees either in existing camps or in camps at candidate locations by building them and assigning/reassigning the refugee camps to hospitals whose resources are defined by the parameter $c'_h$. The problem ensures that (i) current locations of all existing refugees remain unchanged, (ii) all new refugees are located, (iii) all refugees are provided with necessary healthcare services and (iv) the cost related to the change in the current configuration is minimized.

## 5.2. A Bilevel Mixed Integer Programming Formulation

The BOpt-RRC is modeled using a BMIP formulation. The definitions of the sets, parameters and decision variables used in the formulation are given as follows.

*Sets:*

- $E$ Set of existing refugee camps.
- $C$ Set of candidate refugee camps.
- $S$ Set of sources of refugees.
- $H$ Set of hospitals.
- $\mathcal{V}$ Set of all vertices, i.e., $\mathcal{V} = E \cup C \cup S \cup H$.
- $\mathcal{E}$ Set of all edges.

*Parameters:*

$\alpha_e$ Unit cost of increase in the capacity of existing camp $e \in E$.

$\upsilon_c$ Penalty of building candidate camp $c \in C$.

$\rho_{eh}$ Penalty of reassigning existing camp $e \in E$ to $h \in H$.

$\bar{Q}_e$ Current capacity of existing camp $e \in E$.

$\bar{a}_e$ limit on the capacity of existing camp $e \in E$.

$\chi_{ch}$ Cost of assignment of candidate camp $c \in C$ to $h \in H$.

$\chi_{eh}$ Cost of assignment of existing camp $e \in E$ to $h \in H$.

$\xi_c$ Cost of establishing candidate camp $c \in C$.

$\phi_{se}$ Total cost of refugee flow from source $s \in S$ to existing camp $e \in E$.

$\phi_{sc}$ Total cost of refugee flow from source $s \in S$ to candidate camp $c \in C$.

$q_c$ Capacity of candidate camp $c \in C$

$\bar{T}_e$ Current number of refugees located in existing camp $e \in E$.

$d_s$ Total number of refugees originating at source $s \in S$.

$c_h$ Total number of physicians located at hospital $h \in H$.

$\bar{X}_{eh}$ Indicator variable of assignment of existing camp $e \in E$ to hospital $h \in H$.

$\psi$ Maximum number of refugees that can be assigned to a physician.

*Decision Variables:*

$Q_e$ Capacity of existing camp $e \in E$.

$A_e$ Amount of increase in the capacity of existing camp $e \in E$.

$Y_c$ Binary variable which is equal to one if $c \in C$ is built, and zero otherwise.

$R^+_{eh}$ Continuous variable to check if $e \in E$ is reassigned to $h \in H$ in the new configuration.

$R^-_{eh}$ Continuous variable to check if $e \in E$ is no more served by $h \in H$ in the new configuration.

$X_{ch}$ Binary variable which is equal to one if $c \in C$ is assigned to $h \in H$, and zero otherwise.

$X_{eh}$ Binary variable which is equal to one if $e \in E$ is assigned to $h \in H$, and zero otherwise.

$F_{sc}$ Fraction of refugee flow originating at $s \in S$ and destined at $c \in C$.

$F_{se}$ Fraction of refugee flow originating at $s \in S$ and destined at $e \in E$.

$T_c$ Total number of refugees located at candidate camp $c \in C$.

$T_e$ Total number of refugees located at existing camp $e \in E$.

The BMIP formulation for the BOpt-RRC is given as

$$\min_{\mathbf{Q},\mathbf{A}} \quad \sum_e \alpha_e A_e + \sum_c \upsilon_c Y_c + \sum_{e,h} \rho_{eh}(R^+_{eh} + R^-_{eh}), \tag{5.1}$$

$$\text{s.t.} \quad Q_e = \bar{Q}_e + A_e \qquad e \in E, \tag{5.2}$$

$$A_e \leq \bar{a}_e \qquad e \in E, \tag{5.3}$$

$$Q_e, A_e \in \mathbb{Z} \cup \{0\} \qquad e \in E, \tag{5.4}$$

$$\min_{\mathbf{X},\mathbf{Y},\mathbf{F}} \quad \sum_{c\in C}\sum_{h\in H} \chi_{ch}X_{ch} + \sum_{e\in E}\sum_{h\in H} \chi_{eh}X_{eh} + \sum_{c\in C} \xi_c Y_c + \tag{5.5}$$

$$+ \sum_{s\in S}\sum_{e\in E} \phi_{se}F_{se} + \sum_{s\in S}\sum_{c\in C} \phi_{sc}F_{sc},$$

$$\text{s.t.} \quad \sum_{e\in E} F_{se} + \sum_{c\in C} F_{sc} = 1 \qquad s \in S, \tag{5.6}$$

$$\bar{T}_e + \sum_{s\in S} d_s F_{se} = T_e \qquad e \in E, \tag{5.7}$$

$$\sum_{s\in S} d_s F_{sc} = T_c \qquad c \in C, \tag{5.8}$$

$$T_e \leq Q_e \qquad e \in E, \tag{5.9}$$

$$T_c \leq q_c Y_c \qquad c \in C, \tag{5.10}$$

$$\sum_{c\in C} T_c X_{ch} + \sum_{e\in E} T_e X_{eh} \leq \psi c'_h \qquad h \in H, \tag{5.11}$$

$$\sum_{h\in H} X_{eh} \geq 1 \qquad e \in E, \tag{5.12}$$

$$\sum_{h\in H} X_{ch} \geq Y_c \qquad c \in C, \tag{5.13}$$

$$X_{eh} = \bar{X}_{eh} + R^+_{eh} - R^-_{eh} \qquad e \in E, h \in H, \tag{5.14}$$

$$X_{eh} \in \{0,1\} \qquad e \in E, h \in H, \tag{5.15}$$

$$X_{ch} \in \{0,1\} \qquad c \in C, h \in H, \tag{5.16}$$

$$Y_c \in \{0,1\} \qquad c \in C, \tag{5.17}$$

$$T_e \in \mathbb{Z} \cup \{0\} \qquad e \in E, \tag{5.18}$$

$$T_c \in \mathbb{Z} \cup \{0\} \qquad c \in C, \quad (5.19)$$

$$F_{se} \in [0,1] \qquad s \in S, e \in E, \quad (5.20)$$

$$F_{sc} \in [0,1] \qquad s \in S, c \in C, \quad (5.21)$$

$$R^{+}_{eh}, R^{-}_{eh} \geq 0 \qquad e \in E, h \in H. \quad (5.22)$$

In BOpt-RRC, expressions (5.1)–(5.4) represent the ULP, and expressions (5.5)–(5.22) represent the LLP. The ULP deals with capacity increases of the built refugee camps and the LLP readjusts the current refugee camp network configuration with respect to given increased capacity values. The objective function of the ULP given in (5.1) aims to minimize the total cost of capacity increase and the penalty cost of changes in the current configuration. Equations (5.2) calculate the increased capacity of each existing camp $e \in E$, whereas Constraints (5.3) ensure that the total increased capacity of each camp $e$ does not exceed the corresponding upper limit $\bar{a}_e$ for the capacity increase. Restrictions (5.4) define the domains of the ULP variables. The objective function of the LLP (5.5) minimizes the sum of the costs of hospital assignments, building new refugee camps and refugee allocations to built/candidate camps. Equations (5.6) guarantee that all new refugees are located. Equations (5.7)–(5.8) work as flow conservation constraints. Constraints (5.9) prevent the total number of refugees $T_e$ located in each existing camp $e$ from exceeding the camp capacity. In a similar manner, Constraints (5.10) assure that the total number of refugees located in each candidate camp $c$ do not exceed the camp capacity. Equations (5.11) are the resource constraints for each hospital $h$ and do not allow assignments of camps to hospitals if the current number of physicians located at hospital $c'_h$ is not adequate. Constraints (5.12)–(5.13) guarantee the assignment of existing and candidate refugee camps to at least one hospital, respectively. Equations (5.14) keep track of the reassignment of each existing camp $e$ to hospitals. The remaining constraints of the LLP are domain restrictions for LLP variables.

The resource constraints given in Equations (5.11) incorporate multiplications of binary and integer variables, i.e., $X$ and $T$, and hence the LLP is a Mixed-Integer

Nonlinear Program (MINLP). Fortunately, it is possible to linearize these equations and transform the LLP into an MILP to reduce the computational complexity. To do so, we introduce the variables $U_{eh}$ and $U_{ch}$ such that $U_{eh} = T_e X_{eh}$ and $U_{ch} = T_c X_{ch}$ and interchange Constraints (5.11) with

$$\begin{aligned}
\sum_{c \in C} U_{ch} + \sum_{e \in E} U_{eh} &\leq \psi c'_h && h \in H, \\
T_c &\leq \sum_{h \in H} U_{ch} && c \in C, \\
T_e &\leq \sum_{h \in H} U_{eh} && e \in E, \\
U_{ch} &\leq q_c X_{ch} && c \in C, h \in H, \\
U_{eh} &\leq Q_e X_{eh} && e \in E, h \in H, \\
X_{ch} &\leq U_{ch} && c \in C, h \in H, \\
X_{eh} &\leq U_{eh} && e \in E, h \in H.
\end{aligned}$$

and, thanks to this transformation, it is possible to solve the LLP with given ULP variable values with an off-the-shelf MILP solver.

## 5.3. Solution Methods

In this study, we propose two TS based matheuristics and two VNS based matheuristics to solve the BOpt-RRC. This section is dedicated to the details of the solution methods. In Section 5.3.1 we discuss TS implementations where Section 5.3.2 is reserved for VNS implementations.

### 5.3.1. Tabu Search Based Matheuristics

In this section, we propose two TS based matheuristics to solve the BOpt-RRC. The first method called TS1 is a generic TS developed as a benchmark. The second one incorporates an adaptive neighborhood selection procedure to TS1. The following list provides the definitions of common components in both implementations:

- Solution Representation: Each solution $s$ is represented by an integer vector of size $|e|$ in which each component represents the capacity increase in the existing camp $e \in E$. For example, $s = [0, 500, 1000, 0, 0]$ indicates that capacities of existing camps 2 and 3 are increased by 500 and 1000, whereas the capacities of existing camps 1,4 and 5 remain unchanged. Hence, $s$ is the vector of $[A_1, A_2, ..., A_{|E|}]$ in terms of decision variables such that $A_e \in [0, \bar{a}_e]$ for all $e = 1, 2, ..., |E|$.
- Neighborhood Structure: Let $\delta$ be the increment size, and $k$ be the number of existing camps to be revised (i.e., their capacities are updated) such that $k \leq |E|$. Then, the neighborhood $\mathcal{N}_k^\delta$ consists of all combinations $\binom{|E|}{k}$ where the capacities of $k$ existing camps are changed by $\pm\delta$ in each combination. For instance, let $s = [500, 0]$ be a solution and $\delta = 500$. Then, $\mathcal{N}_1^\delta(s) = \{[1000, 0], [500, 500], [0, 0], [500, 0]\}$. In the first neighbor $[1000, 0]$, the capacity of the first camp is increased by 500, and in the second neighbor $[500, 500]$, the capacity of the second camp is increased by 500. In a similar manner, in the third neighbor $[0, 0]$ we decrease the capacity of the first existing camp by 500. For $k = 2$, the neighborhood $\mathcal{N}_2^\delta(s)$ of a solution $s$ is defined as $\{[0, 0], [1000, 0], [0, 500], [1000, 500]\}$. In the first neighbor, the capacities of both camps are decreased by 500 (capacity increase of the second camp remains unchanged due to the nonnegativity). In the second neighbor, the capacity of the first camp is increased and the capacity of the second camp is decreased by 500 (capacity increase of the second camp remains unchanged due to the nonnegativity). In the third neighbor, the capacity of the first camp is decreased and the capacity of the second existing camp is increased by 500. In the last neighbor, capacities of both camps are increased by 500. The size of the neighborhood as a function of $|E|$ and $k$ is given by the formula

$$|\mathcal{N}_k^\delta(s)| = \binom{|E|}{k} \times 2^k.$$

The size of the neighborhood $\mathcal{N}_k^\delta(s)$ for a given $\delta$ increases with the value of $k$. Larger $k$ values indicate more diverse neighborhoods. A concrete example can be given as follows. Let $|E| = 3$ and $s$ be the center of a cube with edge length $2\delta$. Then, $\mathcal{N}_1^\delta(s)$ is the set of centers of all faces, $\mathcal{N}_2^\delta(s)$ is the set of middle points

of all edges and $\mathcal{N}_3^\delta(s)$ is the set of all vertices of the cube and the sizes of these neighborhoods are equal to $\delta, \sqrt{2}\delta$ and $\sqrt{3}\delta$, respectively.

- Neighborhood Evaluation: Let $s = [A_1, A_2, A_3, A_4, A_5]$ be a solution for the BOpt-RRC. The revised capacities $Q_e$ are calculated by Equation $Q_e = \bar{Q}_e + A_e$ and fed into the LLP. Then, the LLP of the BOpt-RRC is optimally solved by an off-the-shelf solver, and the optimal variable values alongside with $A_e$ values are used to calculate the objective value of the solution $s$.
- Selection of the Neighbors: At each iteration, TS generates all members of the neighborhood $\mathcal{N}_k^\delta(s)$ for given $s$, $\delta$ and $k$, and computes the objective value of each $s' \in \mathcal{N}_k^\delta(s)$. The best solution $s^*$ among all evaluated neighbors is chosen as the next solution.
- Initial Solution: A random vector of size $|E|$ with values from the closed interval $[0,\bar{a}_e]$ as multiples of $\delta$ is generated and set as the initial solution $s_0$.
- Tabu Structure: Let $s$ be a solution and $s'$ be the next solution from $\mathcal{N}_k^\delta(s)$. The reverse of any increase/decrease move to obtain $s'$ from $s$ is stored in the tabu list as the tabu move during the tabu tenure. In the implementation, we adopt a static approach for the tabu tenure in which the size of the tabu list is fixed.
- Termination Criterion: The algorithm terminates if a time limit is reached.

Regarding the common components defined above, the pseudocode of the TS1 for BOpt-RRC is outlined in Figure 5.1. At each iteration, TS1 generates a candidate solution set $\mathcal{C}_k^\delta(s)$ as indicated in line 5. To do so, the set of all solutions in the neighborhood $\mathcal{N}_k^\delta(s)$ is enlisted and a subset of this set is chosen as $\mathcal{C}_k^\delta(s)$ prior to the objective function evaluation. For a given $\delta$, different $k$ values return different neighborhoods with various sizes. This fact makes the task of choosing $k$ a challenge from two perspectives. The number of solutions to be evaluated at each iteration varies by $k$ and each neighborhood $\mathcal{N}_k^\delta(s)$ introduces a different intensification/diversification capability into the TS implementation.

Let $|E| = 5$, then the neighborhood size is 10, 40, 80, 80 and 32 for $k = 1, 2, ..., 5$, respectively. To recover the complexity issues related to the neighborhood size differ-

ences, we propose the following candidate solution set generation procedure. Define $\nu$ as the fixed neighborhood size for each $k$, and let $\nu = |\mathcal{N}_1^\delta(s)|$. For a given $k$, generate the set $\mathcal{C}_k^\delta(s)$ by random sampling without replacement from the corresponding neighborhood $\mathcal{N}_k^\delta(s)$ such that $|\mathcal{C}_k^\delta(s)| = \nu$. The procedure is valid because for $|E| \geq 2$, $|\mathcal{N}_1^\delta(s)| \leq |\mathcal{N}_k^\delta(s)|$ holds for all $k = 1, 2, ..., |E|$, and $\mathcal{N}_1^\delta(s)$ is the only neighborhood for $|E| = 1$. By using this procedure, the number of objective function evaluations at each iteration can be fixed by compromising some set of solutions. This procedure is incorporated into TS1 implementation outlined in Figure 5.1.

---

**Input:** Initial solution $s_0$, $\delta$, $k$

$s \leftarrow s_0$ ▷ Initial solution

$s_{best} \leftarrow s$ ▷ Initialize the best solution

$TabuList \leftarrow \emptyset$

**while** stopping_criterion is not satisfied **do**

  Generate $\mathcal{C}_k^\delta(s) \subseteq \mathcal{N}_k^\delta(s)$ ▷ Generate the candidate list avoiding tabu moves

  Evaluate $s' \in \mathcal{C}_k^\delta(s)$ ▷ Evaluate every solution in the candidate list

  $s \leftarrow \arg\max_{s' \in \mathcal{C}_k^\delta(s)} f(s')$ ▷ Pick the best neighbor as the current solution

  **if** $f(s) < f(s_{best})$ **then**

    $s_{best} \leftarrow s$ ▷ Update the best solution

  **end if**

  Update $TabuList$

**end while**

**Output:** $s_{best}$

---

Figure 5.1. Tabu Search - TS1.

Although the size of the candidate solution sets $\mathcal{C}_k^\delta(s)$ is fixed for every $k$ and the number of objective function evaluations at each iteration is the same for all $\mathcal{C}_k^\delta(s)$, there is still an issue to be addressed. Each neighborhood definition has its own intensification/diversification capability and each of them (possibly) leads the search to different regions of the solution space. Hence, the following question should be asked. What should be the fixed value of $k$, and hence the corresponding candidate solution set $\mathcal{C}_k^\delta(s)$? Or, should $k$ be dynamic in the sense that the value of $k$ changes during the iterations? One of the answers to this well-studied question resides in *adaptive neighborhood selection* [132, 133]. This idea is adopted in the TS implementation where a

data-driven TS matheuristic is proposed. The TS heuristic and the newly introduced components are described in detail in Section 5.3.1.1.

5.3.1.1. Tabu search with adaptive neighborhood selection. In this section, we introduce a TS matheuristic for the BOpt-RRC, in which a value function based adaptive procedure (VF/ANS) is used for neighborhood selection at each iteration. The details of the implementation are outlined in Figure 5.2. The heuristic TS/ANS is an extension of TS1 outlined in Figure 5.1, where the value of $k$ is *dynamically* changed by the VF/ANS procedure instead of adopting the static approach applied in TS1.

---

**Input:** $s_0$, $Q^0$ $\tau_0$, $\alpha$, $\beta$, $\theta$, $\lambda$, $\delta$, $k_{max}$

$s \leftarrow s_0$ ▷ Initial solution
$s_{best} \leftarrow s$ ▷ Initialize the best solution
$TabuList \leftarrow \emptyset$
$i \leftarrow 0$
$\tau \leftarrow \tau_0$ ▷ Initial temperature
**while** stopping_criterion is not satisfied **do**
  $P^i \leftarrow ActionProbabilites(Q^i, \tau)$ ▷ Equation (5.23)
  $k \leftarrow RouletteWheel(P^i, \{1, 2, ..., k_{max}\})$ ▷ Pick an action, i.e., select the neighborhood
  Generate $\mathcal{C}_k^\delta(s) \subseteq \mathcal{N}_k^\delta(s)$ ▷ Generate the candidate list without tabu moves
  Evaluate $s' \in \mathcal{C}_k^\delta(s)$ ▷ Evaluate every solution in the candidate list
  $s \leftarrow \arg\max_{s' \in \mathcal{C}(s)} f(s')$ ▷ Accept the best neighbor as the current solution
  $r_k^i \leftarrow CalcRew(f(s), f(s_{best}), \alpha, \beta)$ ▷ Equation (5.24)
  $Q^{i+1} \leftarrow UpdRews(Q^i, r_k^i, \lambda)$ ▷ By only calculating $q_{n_k^i}^{i+1}$ with Equation (5.25)
  **if** $f(s) < f(s_{best})$ **then**
    $s_{best} \leftarrow s$ ▷ Update the best solution
  **end if**
  Update $TabuList$
  $\tau \leftarrow \theta\tau$ ▷ Reduce temperature
  $i \leftarrow i + 1$
**end while**

**Output:** $s_{best}$

---

Figure 5.2. Tabu search with adaptive neighborhood selection.

Before going into the details of the TS/ANS, it is useful to define the VF/ANS with some terminology adapted from the RL literature [37]. In the VF/ANS procedure,

*actions* are defined as the set of neighborhood definitions $\mathcal{N}_k^\delta(s)$ for $k \in \{1, 2, ..., k_{max}\}$, *action values* are defined as the expected *rewards* if the corresponding actions are taken and the reward is defined as the immediate outcome of an action.

The TS/ANS is initialized with a given solution and the initial vector of the action values $Q^0 = [1, 1, ..., 1]$ such that $|Q^0| = k_{max}$. In line 7, the action probabilities $p_k^i$ are calculated via the equation

$$p_k^i = \frac{e^{q_k^i/\tau}}{\sum_{k \in N} e^{q_k^i/\tau}} \tag{5.23}$$

where $i$ denotes the iteration number and $q_k^i$ represents the action value of $k$ at iteration $i$. At each iteration, action values of all $k$ are stored in the vector $P^i$ such that $P^i = [p_0^i, p_1^i, ..., p_{k_{max}}^i]$. The probability distribution defined by Equation (5.23) is known as Boltzmann (or Gibbs) distribution. The function assures that all action probabilities are between 0 and 1, and the summation of all probabilities adds up to 1. The term $\tau$ is known as the temperature, which is used to avoid assigning disproportionately large probability(ies) to a single action (or a few actions) at earlier iterations. After calculating $P^i$, the next action (the value of $k$) is selected with roulette wheel method in line 8. Once $k$ is selected, the heuristic finds the best solution in the neighborhood $\mathcal{N}_k^\delta(s)$ and accepts it as the current solution through the lines 9–11. The reward for action $k$ is calculated using the objective value $f(s_k^i)$ of this currently found solution by

$$r_k^i = \alpha\Big(1 - \frac{f(s_k^i) - f(s_{best})}{f(s_{best})}\Big)^\beta. \tag{5.24}$$

Once the reward $r_k^i$ is calculated, the corresponding action value is updated via the value function formula

$$q_k^{i+1} = q_k^i + \lambda(r_k^i - q_k^i), \tag{5.25}$$

while the values for all other actions remain unchanged. The term $\lambda$ is the adaptation rate [134], and it takes continuous values from the interval $(0, 1]$. In our implementation the value of $\lambda$ is fixed at throughout all iterations. In line 18, the temperature is

reduced by using the formula $\tau\theta$, such that $\theta \in (0,1)$. Using a decreasing function for $\tau$ ensures smaller differences between action probabilities at earlier iterations and larger differences at later ones. Such a structure helps the selection procedure to have a diversification/intensification balance by favoring diversification at the initial iterations and intensification at the later iterations of the heuristic.

### 5.3.2. Variable Neighborhood Search based Matheuristics

In this section, we propose two VNS based matheuristics for the BOpt-RRC. The first one is a generic VNS, and the latter is a data-driven VNS in which an association rule based injection (ARBI) procedure is used for shaking. In both VNS implementations, the solution representation, neighborhood evaluation, initial solution, and termination criterion components are adopted from the TS based matheuristics defined in Section 5.3.1. In addition to these common components, the following is the list of additional components used in both VNS implementations:

- Neighborhood Definition for Shaking: Recall the neighborhood definition $\mathcal{N}_k^{\delta}(s)$. Let $\kappa_{max}$ be a positive integer such that $\delta\kappa_{max} \leq \max\{\bar{a}_e : e \in E\}$ for a given $\delta$. In the VNS implementation, we use the neighborhoods

$$\bigcup_{n\in\{1,2,...,\kappa\}} \mathcal{N}_{|E|}^{n\delta}(s) \tag{5.26}$$

  for $\kappa \in \{1,2,...,\kappa_{max}\}$ for shaking. In the remainder of the text, we use $\mathcal{N}^{\kappa}(s)$ to represent the neighborhood defined by $\kappa$ for the sake of readability. Using nested neighborhoods are usually preferred in variable neighborhood descent (VND) and VNS heuristics, if not always [16]. The structure of $\mathcal{N}^{\kappa}(s)$ naturally implies a nested neighborhood with increasing $\kappa$ values for a given $\delta$, i.e., $\mathcal{N}^1(s) \subset \mathcal{N}^2(s) \subset ... \subset \mathcal{N}^{k_{max}}(s)$.
- Local Search: In both implementations, we use the TS/ANS heuristic introduced in Section 5.3.1.1 as the local search component.

The pseudocode of the generic VNS heuristic is outlined in Figure 5.3. The search is started with an initial solution $s_0$ and the neighborhood $\mathcal{N}^1(s)$ is selected as the current neighborhood by choosing $\kappa$ as 1. In line 5, a solution $s'$ is randomly sampled from the current neighborhood and fed into the local search procedure in line 6. If the resulting solution $s''$ of the local search is better than the current solution $s$, $s''$ is chosen as the current solution and the value of $\kappa$ is updated as 1 through the lines 7–9. Elsewhere, $\kappa$ is updated as $\kappa + 1$ and the next neighborhood $\mathcal{N}^{\kappa+1}(s)$ is selected as the current neighborhood in line 11. The heuristic returns the best solution found during the search.

**Input:** $s_0$, $\kappa_{max}$.
1: $s \leftarrow s_0$ ▷ Initial solution
2: **while** stopping_criterion is not satisfied **do**
3: $\kappa \leftarrow 1$
4: **while** $\kappa \leq \kappa_{max}$ **do**
5: $s' \leftarrow$ pick a random neighboring solution from $\mathcal{N}^{\kappa}(s)$ ▷ Shaking
6: $s'' \leftarrow LocalSearch(s')$ ▷ Apply local search
7: **if** $f(s'') < f(s)$ **then**
8: $s \leftarrow s''$
9: $\kappa \leftarrow 1$
10: **else**
11: $\kappa \leftarrow \kappa + 1$
12: **end if**
13: **end while**
14: **end while**
**Output:** Best solution found.

Figure 5.3. Variable Neighborhood Search.

It is known in the literature that adding bias to the shaking step of a generic VNS implementation may enhance the search capability [10, 135]. Hence, we introduce the ARBI procedure as a data-driven pattern injection method (an example for routing type problems is studied by [136]) and incorporate it into the VNS implementation for the BOpt-RRC outlined in Figure 5.3. The details of the new matheuristic VNS with Association Rules (VNS/AR) and its components are described in Section 5.3.2.1.

5.3.2.1. Shaking with association rules. In this section, we introduce the ARBI procedure and incorporate it into the generic VNS implementation outlined in Figure 5.3. The ARBI is a three-step procedure in which a set $\Gamma$ of *good* solutions is first filtered from the set $\Lambda$ of all visited solutions. A set of good solution components is extracted from $\Gamma$ as an association rule $\rho$ and these components are injected into the shaken solution $s'$ to obtain $s'_\rho$ to feed the local search. The details of these steps are as follows:

- Filtering Good Solutions: Let $\Lambda^i$ be a subset (details of generating this subset is given below) of all visited solutions up to iteration $i$. The ARBI uses the $k$-means clustering algorithm to cluster these solutions into $c$ clusters by their objective values. Then, the cluster of the solutions with minimum average objective values is selected as $\Gamma^i$ which is the set of good solutions at iteration $i$. After the filtering, the ARBI forgets all other visited solutions and only stores the good solutions to be expanded with newly visited solutions at iteration $i+1$. By doing so, the size of the set $\Lambda^i$ of visited solutions is kept manageable in terms of memory usage and computational time of the $k$-means clustering. From the heuristic point of view, this scheme of storing visited solutions resembles the replacement operators from EAs [137]. Filtering good solutions with clustering necessitates a single parameter, i.e., number of clusters $c$, and we suggest smaller values for $c$. The reason behind this suggestion is based on the ability of extracting diverse patterns. Larger $c$ values return a smaller set of good solutions (not necessarily, but most of the time) and rules extracted from smaller sets favor a few of the solution components. This may result in less-diverse solutions by leading the search into similar regions, and hence the heuristic may end up at local optimal solutions of low quality. The good solutions at iteration $i$ is given as the set $\{\gamma : \gamma \in \Gamma^i\}$ and each $\gamma$ is represented as an itemset in which the items corresponding to each existing camp $e$ has the information of the index $e$ and the value of $A_e$ for all $e \in 1, 2, ..., |E|$. Following is an example of an itemset in $\Gamma^i$ for $|E| = 3$: $\gamma = \{500\_1, 0\_2, 1000\_3\}$. The components of $\gamma$ indicate that $A_1 = 500, A_2 = 0$ and $A_3 = 1000$.

- Association Rule Extraction: In ARBI, the common components in the good solution set $\Gamma^i$ is extracted by using the procedure outlined in Figure 5.4. The procedure is based on the Apriori algorithm proposed by [43] used for extracting association rules. The algorithm necessitates a set of parameters and the conditions defined by these parameters may not generate a rule. The main idea of the algorithm is to relax the conditions iteratively down to a limit in the hope of extracting a rule. To do so, the procedure chooses the minimum support parameter $\sigma$ of the Apriori algorithm as $\sigma_{max}$ in line 3. In the main loop starting at line 4, the Apriori algorithm is called to extract the association rules $P$. If at least a rule is extracted, which is checked in line 6, the procedure returns the association rule $\rho$ by filtering the rules with maximum size in line 7, then selecting one of the filtered rules with the largest support value in line 8. If $|P| = 0$ after Apriori, then $\sigma$ is reduced by $\Delta$ in line 11. The loop continues until at least a rule is extracted or the value of $\sigma$ drops below $\sigma_{min}$. Depending on the value of $\sigma_{min}$, the procedure may return no rule at all, i.e., the output is $\rho = \emptyset$.
- Injection of Good Components: The components of a shaken solution $s'$ indicated by the extracted rule $\rho$ is replaced by the corresponding values in $\rho$. Let $s' = [500, 0, 1000]$ and $\rho = \{1000\_1, 0\_3\}$. The ARBI injects the components given by $\rho$ into $s'$ and returns the resulting solution $s'_\rho = [1000, 0, 0]$ to initialize the local search.

The generic VNS including the ARBI procedure as a pattern injection procedure, VNS/AR, is outlined in Figure 5.5. The algorithm starts as the generic VNS. Lines 10–12 call the $k$-means clustering algorithm to obtain the set of good solutions $\Gamma^i$ if the size of $\Lambda^i$ allows. If the size of the good solutions $\Gamma^i$ returned by the clustering is larger than $\gamma$, which is checked in line 13, the ARBI extracts the rule in line 14. If the procedure returns an association rule $\rho$, the algorithm injects the good components indicated by $\rho$ into the shaken solution $s'$ to obtain $s'_\rho$ in line 16. Then, the solution $s'_\rho$ is fed into the local search as the initial solution. If the conditions to call the ARBI procedure is not satisfied in an iteration, the algorithm outlined in Figure 5.5 uses $s'$ as the initial solution of the local search instead of $s'_\rho$ in that iteration by calling the

lines 19 or 22. The heuristic returns the best solution found during the search as the output. Please note that, whenever the local search is called, the algorithm feeds a set of solutions (in the form of $\Lambda^i$ or $\Gamma^i$) into the search alongside with the initial solution. The search then, returns the set of visited solutions $\Lambda^{i+1}$ by adding the newly visited solutions to the provided set of solutions in compliance with the ARBI procedure.

---

**Input:** $\Gamma, \sigma_{min}, \sigma_{max}, \Delta$.

1: $P \leftarrow \emptyset$ ▷ Set of association rules
2: $\rho \leftarrow \emptyset$ ▷ Association rule
3: $\sigma \leftarrow \sigma_{max}$ ▷ Initial minimum support
4: **while** $\sigma_{min} \leq \sigma \leq \sigma_{max}$ **do**
5: $P \leftarrow Apriori(\Gamma, \sigma)$ ▷ Learn a set of rules for given minimum support
6: **if** $P \neq \emptyset$ **then**
7: $P' \leftarrow \{r \in P : |\rho_r| = \max\{|\rho_r| : \forall r \in P\}\}$ ▷ Subset of rules with maximum rule size
8: $\rho \leftarrow \{r \in P' : \sigma_r = \max\{\sigma_r : \forall r \in P'\}\}$ ▷ The rule with maximum support
9: **break**
10: **else**
11: $\sigma \leftarrow \sigma - \Delta$ ▷ Reduce the minimum support
12: **end if**
13: **end while**

**Output:** $\rho$.

---

Figure 5.4. Extracting association rule.

**Input:** $s_0$, $\delta$, $\kappa_{max}$.

$s \leftarrow s_0$ ▷ Initial solution
$i \leftarrow 0$
$\Lambda^i \leftarrow \emptyset$ ▷ Set of visited solutions
$\Gamma^i \leftarrow \emptyset$ ▷ Set of *good* solutions
$\rho \leftarrow \emptyset$ ▷ Association rule
**while** stopping_criterion is not satisfied **do**
    $\kappa \leftarrow 1$
    **while** $\kappa \leq \kappa_{max}$ **do**
        $s' \leftarrow$ pick a random neighboring solution from $\mathcal{N}^{\kappa}(s)$ ▷ Shaking
        **if** $|\Lambda^i| > c - 1$ **then**
            $\Gamma^i \leftarrow Clustering(\Lambda^i, c)$ ▷ Filter *good* solutions
        **end if**
        **if** $|\Gamma^i| > \gamma_{min}$ **then**
            $\rho \leftarrow RuleExtraction(\Gamma^i)$ ▷ Extract the association rule
            **if** $\rho \neq \emptyset$ **then**
                $s'_{\rho} \leftarrow RuleBasedInjection(s', \rho)$ ▷ Inject *good* components
                $s'', \Lambda^{i+1} \leftarrow LocalSearch(s'_{\rho}, \Gamma^i)$ ▷ Apply local search and store newly visited solution
            **else**
                $s'', \Lambda^{i+1} \leftarrow LocalSearch(s', \Gamma^i)$ ▷ Apply local search and store newly visited solution
            **end if**
        **else**
            $s'', \Lambda^{i+1} \leftarrow LocalSearch(s', \Lambda^i)$ ▷ Apply local search and store newly visited solution
        **end if**
        **if** $f(s'') < f(s)$ **then**
            $s \leftarrow s''$ ▷ Update the current solution
            $\kappa \leftarrow 1$
        **else**
            $\kappa \leftarrow \kappa + 1$
        **end if**
        $i \leftarrow i + 1$
    **end while**
**end while**

**Output:** Best solution found.

Figure 5.5. Variable neighborhood search with association rules.

# 6. COMPUTATIONAL RESULTS

All computational experiments are carried out on a workstation with Microsoft Windows 7 Professional operating system and Intel Xeon CPU E5-1650 v3 @ 3.50 GHz processor with 16.0 GB RAM. All procedures explained in Sections 4.2 are implemented in a Python environment, and Python API of Gurobi (v9.1.2) is employed as the MILP solver. We set a time limit of 600 seconds in the small-sized instances and 1800 seconds in the large-sized instances for each matheuristic. The number of threads is set to eight for Gurobi while keeping other settings at their default values.

## 6.1. r-Interdiction Network Design Problem with Lost Demand

### 6.1.1. Experimental Settings and Instance Generation

In order to analyse the effectiveness of the three matheuristics explained in Section 4.2, we randomly generate a set of RI-NDPLD test instances. The size of the test set is determined by the number of nodes $|N|$ and the arc density $\zeta$. The number of arcs $|A|$ in each instance is determined as a fraction of the arcs that exist in a complete network. Namely, $|A| = \lceil \zeta \cdot |N| \cdot (|N| - 1) \rceil$. If this value odd, then we increase the number of arcs by one so as to obtain an even number. The number of commodities $|K|$ is set to $\binom{|N|}{2}$. Table 6.1 includes the test instances and their properties.

In total, there are 80 instances in the test bed. They can be categorized as small-sized ($|N| = 7, 8$) and large-sized ($|N| = 9, 10$) instances. The number of arcs, (i.e., $|A|$), in the largest instances of our test bed exceeds the number of direct flights by all US based major/non-major carriers. We remark that the largest direct flight network is operated by the Southwest Airlines with $|A| = 58$ [138]. In this sense, we can say that the test bed is a realistic representation of the size of the problems encountered in the industry. There exist two budget scenarios with $r = 3$ and $r = 5$ interdictions.

Table 6.1. Test instances.

| $\|N\|$ | $\zeta$ | $\|A\|$ | $\|K\|$ | $\|N\|$ | $\zeta$ | $\|A\|$ | $\|K\|$ |
|---|---|---|---|---|---|---|---|
| 7 | 40 | 18 | 42 | 9 | 40 | 30 | 72 |
| 7 | 50 | 22 | 42 | 9 | 50 | 36 | 72 |
| 7 | 60 | 26 | 42 | 9 | 60 | 44 | 72 |
| 7 | 70 | 30 | 42 | 9 | 70 | 52 | 72 |
| 8 | 40 | 24 | 56 | 10 | 40 | 36 | 90 |
| 8 | 50 | 28 | 56 | 10 | 50 | 46 | 90 |
| 8 | 60 | 34 | 56 | 10 | 60 | 54 | 90 |
| 8 | 70 | 40 | 56 | 10 | 70 | 64 | 90 |

6.1.1.1. Setting the parameters of the random forest. The RF regression model is implemented with the `RandomForestRegressor` class of the Python based ML package Scikit-Learn (v0.24.2) [139]. We apply 5-fold cross validation (CV) to fix the parameter values used in the RF models of the DDS procedure implemented in the TS/DDS and TS/P+DDS matheuristics. To do so, we randomly select the smallest instance with $|N| = 7$ and $|N| = 8$. Then, we enumerate all the interdiction patterns for $r = 3$, and $r = 5$, and obtain the optimal LLP solutions using Gurobi and create a dataset by calculating the lower and upper bounds for all interdiction patterns. Lastly, we apply 5-fold CV for the RF model by only varying the number of trees. Other parameters of the RF model are set to their default values by respecting the observation of [140], which declares that the RF regression models are known to work well under default settings without elaborated parameter tuning. Among the three values tried (50, 100, 150), the latter provided slightly smaller CV errors, and we decided to conduct all comparative experiments discussed in Section 6.1.2 with an RF model having 150 trees.

### 6.1.2. Numerical Results

The computational experiments are designed to analyse the efficiency of the matheuristics developed in this study based on 80 randomly generated test instances having interdiction budgets $r = 3$ and $r = 5$. However, we also aim to investigate the effect of each component proposed, namely the pruning procedure using bounds, the DDS procedure, and the RD procedure. To this end, we consider the following variants of the TS based matheuristics:

(i) **TS:** Basic TS based matheuristic with the best improvement strategy, i.e., algorithm outlined in Figure 5.1.

(ii) **TS/P:** TS that incorporates the pruning procedure using bounds where some neighboring solutions are eliminated to generate a candidate list of solutions for which the LLP is solved optimally.

(iii) **TS/DDS:** TS that incorporates the DDS procedure where the objective value of every neighboring solution is first predicted and then sorted. The candidate list of solutions is determined as the top 10% ($\pi = 0.1$) in the list for which the LLP is solved optimally.

(iv) **TS/P+DDS:** TS that incorporates both the pruning procedure using bounds and the DDS procedure.

(v) **TS/P+RS:** TS that incorporates the pruning procedure using bounds and random sampling where 10% of the neighbors remaining after pruning are randomly selected. The LLP is solved optimally only for these solutions.

(vi) **TS/P+DDS+RD:** TS/P+DDS that incorporates RD where feature_importances values are calculated by the RF regression model subsequent to training. These values are used to detect the effect of the change in the interdiction status of each arc on the objective value of solutions.

Since the efficiency of the DDS procedure and random sampling is directly affected by the performance of the pruning procedure in TS/P+DDS, TS/P+RS, and TS/P+DDS+RD, we first discuss the benefit of the BP procedure in Section 6.1.2.1.

Then, we analyse the performance of all matheuristics over small-sized and the large-sized instances in Sections 6.1.2.2 and 6.1.2.3, respectively. Last, we analyse the predictive ability of RF models in Section 6.1.2.4.

6.1.2.1. Analysis of the benefit of the pruning procedure. We measure the benefit of the pruning procedure in terms of the percentage of neighboring solutions eliminated from the whole neighborhood $\mathcal{N}(s)$. In other words, for each instance considered we compute the percent reduction $100 \times (|\mathcal{N}(s)| - |\mathcal{C}(s)|)/|\mathcal{N}(s)|$ where $\mathcal{C}(s)$ is the set of solutions in the candidate after pruning. The results are presented in Table 6.2 where $|N|$ represents the instance size in terms of number of nodes, and $|A|$ is the number of arcs, $|\mathcal{N}(s)|$ is the neighborhood size before pruning, and $|\mathcal{C}(s)|$ is the neighborhood size after pruning. Note that each row is averaged over five instances. For example, when $|N| = 8$ and $|A| = 28$, there are 75 interdiction patterns to be considered in the neighborhood for $r = 3$ and 115 patterns for $r = 5$. After the pruning procedure, the size of the neighboring solutions in the candidate list becomes 37.2 on average for both $r = 3$ and $r = 5$.

Table 6.2. Pruning performance of the BP procedure.

| | | $r = 3$ | | | $r = 5$ | | |
|---|---|---|---|---|---|---|---|
| $\|N\|$ | $\|A\|$ | $\|\mathcal{N}(s)\|$ | $\|\mathcal{C}(s)\|$ | % Pruned | $\|\mathcal{N}(s)\|$ | $\|\mathcal{C}(s)\|$ | % Pruned |
| 7 | 18 | 45 | 9.4 | 79.1 | 65 | 11.4 | 82.5 |
| 7 | 22 | 57 | 30.8 | 56.0 | 85 | 30.8 | 43.8 |
| 7 | 26 | 69 | 42.6 | 28.3 | 105 | 42.4 | 59.6 |
| 7 | 30 | 81 | 60.4 | 25.5 | 125 | 60.4 | 51.7 |
| 8 | 24 | 63 | 24.3 | 61.4 | 95 | 24.2 | 74.5 |
| 8 | 28 | 75 | 37.2 | 50.4 | 115 | 37.2 | 67.3 |
| 8 | 34 | 93 | 80.9 | 13.0 | 145 | 38.6 | 73.4 |
| 8 | 40 | 111 | 104.5 | 5.8 | 175 | 81.0 | 53.7 |
| 9 | 30 | 81 | 41.1 | 49.3 | 125 | 104.6 | 16.3 |
| 9 | 36 | 99 | 62.4 | 36.9 | 155 | 29.5 | 80.9 |
| 9 | 44 | 123 | 118.1 | 4.0 | 195 | 85.8 | 56.0 |
| 9 | 52 | 147 | 141.6 | 3.7 | 235 | 197.1 | 16.1 |
| 10 | 36 | 99 | 36.5 | 63.1 | 155 | 30.2 | 80.5 |
| 10 | 46 | 129 | 70.1 | 45.7 | 205 | 116.9 | 43.0 |
| 10 | 54 | 153 | 128.1 | 16.3 | 245 | 199.9 | 18.4 |
| 10 | 64 | 183 | 183.0 | 0.0 | 295 | 295.0 | 0.0 |

We can conclude that the pruning procedure is able to prune on the average 39.3% and 27.4% of the neighboring solutions for small-sized and large-sized RI-NDPLD instances with $r = 3$, respectively. When $r = 5$, the corresponding reductions amount to 66.3% and 47.0% for small-sized and large-sized instances, respectively. These values indicate that the pruning procedure is indeed beneficial and makes it possible to prune considerable portions of neighboring solutions with an increasing success for larger interdiction budget and larger-sized instances. This shows that pruning is a promising approach from the implementation point of view since the neighborhood size grows with a large budget and network size. Recall that the pruning procedure makes use of LP relaxation and a rounding heuristic for finding a lower bound and upper bound, respectively. It could be possible to further increase the performance of the procedure by developing more elaborate solution methods to calculate the bounds. This idea presents a future research direction since the pruning performance reduces with increasing number of arcs providing evidence that the LP relaxation the rounding heuristic do not provide sufficiently tight bounds in those cases.

6.1.2.2. Analysis of small-sized instances. Due to the complexity of the RI-NDPLD, it is possible to optimally solve only small-sized instances ($|N| = 7$ and $|N| = 8$) with budget $r = 3$ within a reasonable computational time. This is accomplished by enumerating all interdiction patterns in the ULP and solving the LLP using the Gurobi solver given the interdiction pattern. Hence, we compare the quality of the solutions generated by each matheuristic in terms of the percent deviation from the optimal objective value (PD) using the formula $100 \times (z^* - z_h)/z^*$ where $z_h$ is the objective value found by matheuristic $h$. For small-sized instances with budget $r = 3$, we compare the matheuristics on the basis of the best objective value obtained by any matheuristic using the formula $100 \times (z^{\text{best}} - z_h)/z^{\text{best}}$ where $z^{\text{best}} = \max_h(z_h)$. For all instances of the same size we also report the number of optimal solutions (#OS) or the number of best solutions (#BS) found by heuristic $h$.

The performance of the matheuristics on small-sized instances for $r = 3$ are displayed in Table 6.3. PD values are averaged over five instances. The third column

in the table represent the number of interdiction patterns (#IP) computed as $\binom{A}{r}$.

Table 6.3. Performance comparison on small-sized instances when $r = 3$.

| | | | **TS** | | **TS/P** | | **TS/DDS** | | **TS/P+RS** | | **TS/P+DDS** | | **TS/P+DDS+RD** | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $\|N\|$ | $\|A\|$ | #IP | PD | #OS | PD | #OS | PD | #OS | PD | #OS | PD | #OS | PD | #OS |
| 7 | 18 | 816 | 0.00 | 5 | 0.00 | 5 | 0.00 | 5 | 0.00 | 5 | 0.00 | 5 | 0.00 | 5 |
| 7 | 22 | 1540 | 0.00 | 5 | 0.00 | 5 | 0.00 | 5 | 0.00 | 5 | 0.00 | 5 | 0.00 | 5 |
| 7 | 26 | 2600 | 0.26 | 4 | 0.00 | 5 | 0.00 | 5 | 0.00 | 5 | 0.00 | 5 | 0.00 | 5 |
| 7 | 30 | 4060 | 6.95 | 1 | 0.00 | 5 | 0.34 | 4 | 0.34 | 4 | 0.34 | 4 | 0.00 | 5 |
| 8 | 24 | 2024 | 0.59 | 4 | 0.00 | 5 | 0.00 | 5 | 0.00 | 5 | 0.00 | 5 | 0.00 | 5 |
| 8 | 28 | 3276 | 0.59 | 4 | 0.00 | 5 | 0.00 | 5 | 0.00 | 5 | 0.00 | 5 | 0.00 | 5 |
| 8 | 34 | 5984 | 0.04 | 4 | 0.00 | 5 | 5.70 | 4 | 0.00 | 5 | 0.04 | 4 | 0.00 | 5 |
| 8 | 40 | 9880 | 7.23 | 0 | 0.40 | 2 | 4.86 | 3 | 6.40 | 1 | 4.86 | 3 | 0.00 | 5 |
| Average: | | | 1.96 | 3.38 | 0.05 | 4.63 | 1.36 | 4.50 | 0.84 | 4.38 | 0.65 | 4.50 | 0.00 | 5.00 |

On the basis of the average PD values given in the last row of Table 6.3, the only method capable of attaining the optimal solutions for all the instances considered is TS/P+DDS+RD, and it ranks the first among all matheuristics. TS/P shows a similar performance with the TS/P+DDS+RD and ranks the second. In fact, we can observe that all matheuristics incorporating the pruning procedure outperform the basic TS matheuristic. Note that TS/P follows the very same trajectory with TS, but since a number of neighboring solutions are eliminated in the TS/P it is expected to conduct more iterations compared to TS. The average #OS value is 4.50 for both TS/DDS and TS/P+DDS indicating their success is the same in attaining the optimal solutions. Due to the same reasoning as before, as TS/P+DDS can carry out more iterations compared to TS/DDS, its average PD value 0.65 is smaller than 1.36 of TS/DDS. The comparison of TS/P+DDS and TS/P+RS reveals that the DDS procedure has a better performance than random sampling which is also expected.

Table 6.4 provides the results when $r = 5$. Unfortunately, the optimal solutions cannot be obtained due to the increase in the number of interdiction patterns in the ULP and the LLP has to be optimally solved for each pattern to determine the optimal solution and objective value. Therefore, we report the number of best solutions

(#BS) found by the matheuristics instead of optimal solutions, and the PD values are computed based on the best objective values attained.

Table 6.4. Performance comparison on small-sized instances when $r = 5$.

| | | | **TS** | | **TS/P** | | **TS/DDS** | | **TS/P+RS** | | **TS/P+DDS** | | **TS/P+DDS+RD** | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $\|N\|$ | $\|A\|$ | #IP | PD | #BS | PD | #BS | PD | #BS | PD | #BS | PD | #BS | PD | #BS |
| 7 | 18 | 8568 | 0.00 | 5 | 0.00 | 5 | 0.00 | 5 | 0.00 | 5 | 0.00 | 5 | 0.00 | 5 |
| 7 | 22 | 26334 | 0.44 | 4 | 0.44 | 4 | 1.21 | 4 | 0.00 | 5 | 0.00 | 5 | 0.00 | 5 |
| 7 | 26 | 65780 | 3.06 | 2 | 0.01 | 4 | 0.17 | 4 | 0.01 | 4 | 0.00 | 5 | 0.00 | 5 |
| 7 | 30 | 142506 | 5.09 | 1 | 0.00 | 5 | 3.59 | 4 | 2.26 | 3 | 0.65 | 4 | 0.61 | 4 |
| 8 | 24 | 42504 | 1.29 | 3 | 0.12 | 4 | 0.12 | 4 | 0.00 | 5 | 0.00 | 5 | 0.12 | 4 |
| 8 | 28 | 98280 | 6.85 | 1 | 0.00 | 5 | 0.00 | 5 | 0.00 | 5 | 0.00 | 5 | 0.00 | 5 |
| 8 | 34 | 278256 | 8.20 | 2 | 0.00 | 5 | 5.80 | 2 | 1.14 | 3 | 1.90 | 4 | 2.71 | 3 |
| 8 | 40 | 658008 | 4.79 | 1 | 2.93 | 3 | 2.64 | 1 | 4.09 | 2 | 6.64 | 0 | 5.21 | 1 |
| Average: | | | 3.72 | 2.38 | 0.44 | 4.38 | 1.69 | 3.63 | 0.94 | 4.00 | 1.15 | 4.13 | 1.08 | 4.00 |

It can be observed that TS performs the worst due to the inferior diversification capability among all matheuristics; it evaluates all neighboring solutions and thus spends too much computation time without being able to diversify the search. Pruning always helps as can be seen by the comparison between TS/P and TS as well as between TS/P+DDS and TS/DDS. TS/P+DDS outperforms the later one due to the superior intensification capability of the DDS procedure. TS/P performs the best and the difference between TS/P+RS, TS/P+DDS, and TS/P+DDS+RD is not significant. The analysis of large-sized instance are required to better assess the quality of the DDS and RD procedures.

6.1.2.3. Analysis of large-sized instances. The performance of all matheuristics on large-sized instances are presented in Table 6.5 when $r = 3$ and in Table 6.6 when $r = 5$. The structure of the tables is the same as that of the table prepared for small-sized instances with $r = 5$ in which the best feasible solution obtained among all matheuristics is used for the basis of comparison.

Table 6.5. Performance comparison on large-sized instances when $r = 3$.

| | | | **TS** | | **TS/P** | | **TS/DDS** | | **TS/P+RS** | | **TS/P+DDS** | | **TS/P+DDS+RD** | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $\|N\|$ | $\|A\|$ | #IP | PD | #BS | PD | #BS | PD | #BS | PD | #BS | PD | #BS | PD | #BS |
| 9 | 30 | 4060 | 0.00 | 5 | 0.00 | 5 | 0.00 | 5 | 0.00 | 5 | 0.00 | 5 | 0.00 | 5 |
| 9 | 36 | 7140 | 0.00 | 5 | 0.00 | 5 | 0.00 | 5 | 0.00 | 5 | 0.00 | 5 | 0.00 | 5 |
| 9 | 44 | 13,244 | 10.34 | 3 | 10.34 | 3 | 6.13 | 4 | 2.44 | 3 | 0.00 | 5 | 0.00 | 5 |
| 9 | 52 | 22,100 | 0.66 | 3 | 0.66 | 3 | 0.00 | 5 | 0.65 | 2 | 0.00 | 5 | 0.00 | 5 |
| 10 | 36 | 7140 | 0.12 | 4 | 0.00 | 5 | 0.00 | 5 | 0.00 | 5 | 0.00 | 5 | 0.00 | 5 |
| 10 | 46 | 15,180 | 0.00 | 5 | 0.00 | 5 | 0.00 | 5 | 0.00 | 5 | 0.00 | 5 | 0.00 | 5 |
| 10 | 54 | 24,804 | 0.77 | 1 | 0.55 | 2 | 4.37 | 3 | 0.59 | 1 | 0.16 | 4 | 0.29 | 3 |
| 10 | 64 | 41,664 | 0.36 | 2 | 0.34 | 2 | 0.25 | 3 | 0.15 | 3 | 0.06 | 4 | 0.06 | 4 |
| **Average:** | | | 1.53 | 3.50 | 1.49 | 3.75 | 1.34 | 4.38 | 0.48 | 3.63 | 0.03 | 4.75 | 0.04 | 4.63 |

As can be observed, TS/P performs slightly better than TS in terms of both the average PD and average #BS. Although the overall performance of TS/P is relatively lower than that on small-sized instances, the incorporation of the DDS procedure resulting in TS/P+DDS yields the best performance. The DDS procedure also increases the efficiency of a TS implementation with random sampling, i.e., TS/P+RS. In addition to the overall superior performance of both TS/P+DDS and TS/P+DDS+RD, these matheuristics are more robust to an increase in the number of arcs comparing to other matheuristics as can be seen in the #BS columns corresponding to instances with $|A| = 54$ and $|A| = 64$.

Table 6.6 includes the results obtained on the large-sized instances with $r = 5$. As a matter of fact, they are the largest instances considered in this study, as is clear by the numbers of interdiction patters given in the #IP column of the table.

The joint benefit of the pruning and DDS procedures is evident in the results. In terms of the average PD and #BS values, TS/P+DDS outperforms other matheuristics except TS/P+DDS+RD. As expected, TS performs poorly with a PD value of 9.23% and is only able to find the best solution on 1.5 instances on average. Although TS/P is one of the competitive matheuristics in other cases, its performance is rather inferior on these largest instances and cannot compete with the best performing TS/P+DDS

and TS/P+DDS+RD. We believe that this outcome is due to the decreased pruning capability of the pruning procedure with larger number of nodes and arcs (please see Table 6.2). Following a trend similar to that in the results given in Table 6.5 on large-sized instances with $r = 5$, TS/P+DDS outperforms both TS/DDS and TS/P+RS indicating that the DDS procedure can successfully eliminate unpromising neighboring solutions and allows the search to consider promising ones. TS/P+DDS+RD returns the best results in terms of both performance criteria. As before, both TS/P+DDS and TS/P+DDS+RD are less affected by the increase in the number of arcs, which shows their robustness compared to other matheuristics. Hence, we conclude that the proposed data-driven neighbor sorting and data-driven RD procedures are indeed helpful in solving the RI-NDPLD instances.

Table 6.6. Performance comparison on large-sized instances when $r = 5$.

| | | | **TS** | | **TS/P** | | **TS/DDS** | | **TS/P+RS** | | **TS/P+DDS** | | **TS/P+DDS+RD** | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $\|N\|$ | $\|A\|$ | #IP | PD | #BS | PD | #BS | PD | #BS | PD | #BS | PD | #BS | PD | #BS |
| 9 | 30 | 142,506 | 0.36 | 4 | 0.00 | 5 | 0.00 | 5 | 0.00 | 5 | 0.00 | 5 | 0.00 | 5 |
| 9 | 36 | 376,992 | 3.09 | 2 | 0.00 | 5 | 0.00 | 5 | 0.92 | 3 | 0.00 | 5 | 0.00 | 5 |
| 9 | 44 | 1,086,008 | 5.16 | 3 | 5.07 | 4 | 4.25 | 4 | 6.05 | 3 | 6.09 | 3 | 0.06 | 4 |
| 9 | 52 | 2,598,960 | 24.59 | 0 | 24.04 | 0 | 1.22 | 2 | 6.44 | 3 | 0.46 | 4 | 0.00 | 5 |
| 10 | 36 | 376,992 | 1.03 | 2 | 0.52 | 3 | 0.03 | 4 | 0.03 | 4 | 0.00 | 5 | 0.00 | 5 |
| 10 | 46 | 1,370,754 | 22.56 | 0 | 12.59 | 1 | 4.59 | 1 | 2.87 | 3 | 1.27 | 4 | 1.39 | 4 |
| 10 | 54 | 3,162,510 | 15.95 | 0 | 11.68 | 1 | 4.70 | 3 | 6.85 | 1 | 4.47 | 4 | 0.06 | 4 |
| 10 | 64 | 7,624,512 | 1.07 | 1 | 1.20 | 0 | 0.12 | 4 | 0.46 | 2 | 0.12 | 4 | 0.12 | 4 |
| **Average:** | | | 9.23 | 1.50 | 6.89 | 2.38 | 1.86 | 3.50 | 2.95 | 3.00 | 1.55 | 4.25 | 0.21 | 4.50 |

6.1.2.4. Prediction accuracy of the random forest. The efficiency of the DDS procedure relies on its capability in sorting the real objective values. To measure the strength of this capability, we work on an experimental setting in which we compare the ordered lists of neighboring solutions sorted by actual and predicted objective function values and represent the similarity of these lists with the Spearman's rank correlation (also known as Spearman's $\rho$) calculated by

$$\rho = 1 - \frac{6 \sum_{i \in \{1,2,\dots n\}} d_i^2}{n(n^2 - 1)}, \tag{6.1}$$

where $d_i$ is the difference between the rank of the neighbor $i$ in the two lists and $n$ is the number of all solutions in the neighborhood of a solution $s$, i.e., $|\mathcal{N}(s)|$. Spearman's $\rho$ can take values from the interval $[-1, 1]$. If $\rho = 1$ for two sorted lists, each item has the same rank in both lists. For instance, $\rho = 1$ for the lists $A > B > C > D$ and $A > B > C > D$. On the other extreme, $\rho = -1$ indicates a perfect negative correlation, i.e., the sorted lists are $A > B > C > D$ and $D > C > B > A$.

In our experimental setting, we calculate Spearman's rank correlation values at each iteration to analyse the sorting capability of the DDS procedure. Let $\mathcal{N}(s) = \{s'_1, s'_2, s'_3\}$ be the set of neighboring solutions of solution $s$ at iteration $i$, and $f(s)$ be the actual objective value for $s$, and $\hat{f}_i(s)$ be the objective value for s predicted by the RF model retrained with all visited solutions up to iteration $i$. If $f(s'_1) > f(s'_2) > f(s'_3)$ and $\hat{f}(s'_2) > \hat{f}(s'_1) > \hat{f}(s'_3)$ holds, then $d_1 = -1$, $d_2 = 1$ and $d_3 = 0$, and Spearman's $\rho$ at iteration $i$ is equal to 0.5. In this setting, if $\rho = 1$ at an iteration, the selection of the best neighboring solution is guaranteed. However, this rarely occurs due to the fact that the search visits unseen parts of the solution space at each iteration and RF is supposed to predict the objective values of newly seen solutions. From an ML point of view, this setting corresponds to the test error and errors in predictions are inevitable. However, we expect positive $\rho$ values to claim that DDS works properly. To this end, we keep track of the $\rho$ values through TS-DDS iterations (with $\pi = 0.1$) while solving the fifth RI-NDPLD instance with $|N| = 7$, $\zeta = 50$ in Figure 6.1 as an example. For this instance, TS-DDS finds the optimal solution for $r = 3$ and finds the best known solution for $r = 5$.

As can be seen in Figure 6.1, Spearman's $\rho$ values fluctuate around 0.54 for $r = 3$ and 0.43 for $r = 5$ with 0.27 as the minimum for $r = 3$ and 0.16 as the minimum for $r = 5$ indicating the practicality of the proposed candidate set generation procedure incorporating RF models. Since similar patterns are also observed in almost all of the instances with $|N| = 7$ and $|N| = 8$ with a few exceptions, we believe that the quality of the predictions is sufficient.

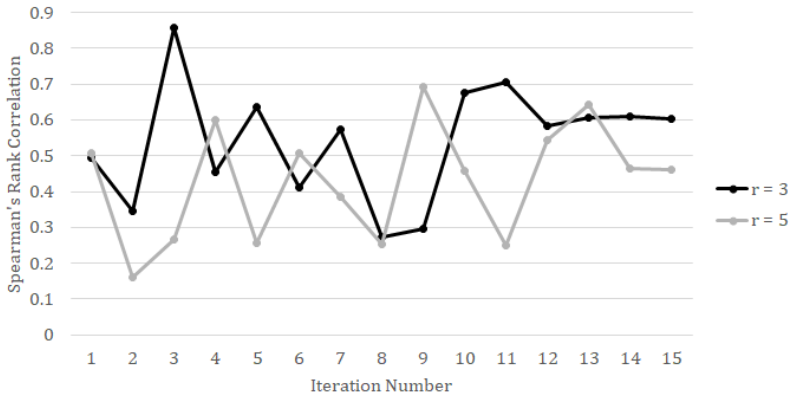


Figure 6.1. Spearman's $\rho$ values at each TS-DDS iteration for the fifth instance with $|N| = 7$, $\zeta = 50$.

## 6.2. Bilevel Optimization Problem for Reconfiguration of Refugee Camp Network

### 6.2.1. Experimental Settings and Instance Generation

In order to assess the computational capabilities of the matheuristics proposed for solving the BOpt-RRC, we randomly generate a set $\mathcal{I}$ of BOpt-RRC instances. The instances are based on the structure of the large size real life dataset introduced in [127]. The real life dataset represents the refugee camp network and public service providers in the southeast region of Turkey including the cities Adana, Adıyaman, Gaziantep, Hatay, Kahramanmaraş, Kilis, Malatya, Mardin, Osmaniye and Şanlıurfa. There is a total of 244 nodes in the network: 74 candidate locations for refugee camps, 60 hospitals, 77 high schools and 33 municipality buildings. In all our instances, we adopt a fixed subset $\mathcal{G}$ of this network: 74 candidate locations for refugee camps and 60 hospitals. We omit the high schools and municipality buildings for the sake of simplicity and assume that the only public service is the healthcare. However, it is a straightforward task to include the remaining nodes. In addition to the candidate location and hospital nodes of [127], there are also *source* nodes in $\mathcal{G}$ and these nodes represent the regions where the new refugees originate.

In all instances of $\mathcal{I}$, the capacity of each candidate refugee camp is randomly chosen from the set $\{5000, 15000, 25000\}$ and the upper limits on the capacity increase in the existing camps are set to 20% of the capacity of each camp. The number of physicians at each hospital is randomly generated from integer numbers in the set $\{3, 4, 5, 6, 7, 8\}$, and the maximum number of refugees $\psi$ that can be served by a physician is selected as 2500. Each instance $i \in \mathcal{I}$ is characterized by the number of current refugees $\omega$ located in the corresponding existing camp, and the ratio $\omega_{new}$ of the number of new refugees to the number of existing refugees. We choose $\omega$ from $\{50000, 100000, 150000, 200000\}$ and generate a uniform random integer from the interval $[\omega \pm 0.25 \times \omega]$ and set it as the number of existing refugees. In a similar manner, we determine $\omega_{new}$ from the set $\{0.02, 0.06, 0.10, 0.20\}$ and generate a uniform random integer from the interval $[\Omega - \Omega \times 0.25, \Omega + \Omega \times 0.25]$ where $\Omega = \omega \times \omega_{new}$ and choose it as the number of new refugees originating at the source nodes. We fix $\bar{a}_e$ values for all of the existing camps as 20% of the current capacities. We determine the minimum value of $\omega$ as 50000 regarding the current status in the Southeast region of Turkey: There are 50,736 refugees located at seven refugee camps scattered within the region [128].

As discussed above, the BOpt-RRC updates the existing refugee camp network configuration. Hence, the initial configurations of all instances are needed for the analysis. We implement a modified version of the LLP of the BOpt-RRC given in Equations (5.5)–(5.22). Then, we input the indices, capacities, and populations of the existing camps and the current hospital assignments to the BOpt-RRC. Regarding the discussed experimental settings we generate 80 instances in which we have five different instances of the same size defined by the Cartesian product of $\{50000, 100000, 150000, 200000\} \times \{0.02, 0.06, 0.10, 0.20\}$. The details of the instances grouped by $\omega_{new}$ values are given in Tables A.1–A.4 in the Appendix.

### 6.2.2. Numerical Results

In this section, we assess the quality of each matheuristic method proposed for the BOpt-RRC. The computational experiments are based on 80 randomly generated

BOpt-RRC test instances with various population parameters $\omega$ and $\omega_{new}$. To this end, we implement the following matheuristics where the increment size $\delta = 500$ and all methods are terminated after the CPU time of 1800 seconds:

(i) **TS/4:** TS1, i.e., Algorithm outlined in Figure 5.1, with the neighborhood $\mathcal{N}_4^{\delta}(s)$. On the basis of preliminary computational studies on various $k$ values, we set $k = 4$.

(ii) **TS/ANS:** Algorithm outlined in Figure 5.2 with $k_{max} = 4$. We set $\alpha = 0.01$ and $\beta = 1$ regarding a set of preliminary experiments.

(iii) **VNS:** Algorithm outlined in Figure 5.3 with $\kappa_{max} = 3$. The LS procedure is TS/ANS with $k_{max} = 4$ and it is terminated after 10 consecutive iterations with no improving solutions.

(iv) **VNS/AR:** Algorithm outlined in Figure 5.5, with $\kappa_{max} = 3$. The LS procedure is TS/ANS with $k_{max} = 4$ and it is terminated after 10 consecutive iterations with no improving solutions. The visited solutions are clustered into two groups, i.e., $k = 2$. The minimum confidence $c$ in the apriori algorithm is 80% and the minimum support takes values between $\sigma_{min} = 0.20$ and $\sigma_{max} = 0.90$ with the step size $\Delta = 0.10$.

Due to the complexity of the BOpt-RRC, it is not possible to obtain the optimal objective values of all instances in a reasonable computational time. The optimal solutions could only be obtained for instances with $\omega = 50000$ by complete enumeration method. These results ares used to assess the performance of all proposed heuristics for the instances with $\omega = 50000$. To this end, the percent deviation (PD) from the optimal objective value is computed using the formula $100 \times (z_h - z^*)/z^*$ where $z_h$ is the objective value found by matheuristic $h$, and $z^*$ is the optimal objective value. For other instances without the optimal solution, the PD values are computed using the formula $100 \times (z_h - z^{\text{best}})/z^{\text{best}}$ where $z^{\text{best}} = \max_h\{z_h\}$. For all instances of the same size we also report the number of best (or optimal if exists) solutions (#BS) found by heuristic $h$.

Before going into the details of the overall comparative study, we first analyse the ML components of TS/ANS and VNS/AR in the Sections 6.2.2.1 and 6.2.2.2, respectively.

6.2.2.1. Analysis of TS/ANS. In this section we briefly discuss the effect of the adaptive neighborhood selection procedure. The average fractions of selected neighborhoods over the instances with the same $\omega$ values are given in Figure 6.2.

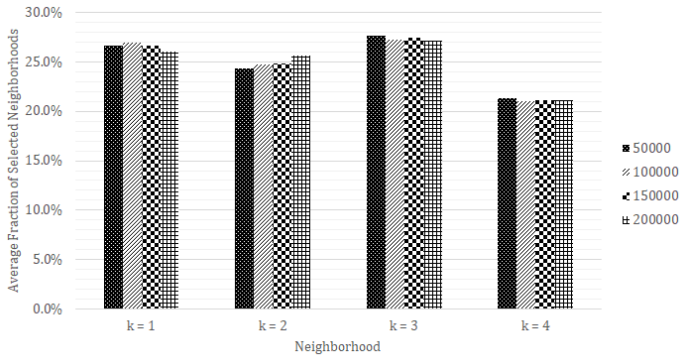


Figure 6.2. Average fraction of selected neighborhoods over the instances with the same $\omega$ values.

Each column in the bar chart represents the average fraction of the corresponding $\omega$ and $k$. A certain trend favoring $k = 3$ is evident in the figure. As discussed above, we choose $k = 4$ for the generic TS implementation, i.e., TS/4, by experiment. However, in TS/ANS solutions, $k = 4$ corresponds to the least selected neighborhood. This is probably caused by the fact that the more diversified search provided by TS/ANS visits the regions of the solution space which are not visited by TS/4 and these regions are searched better by $k = 3$.

The reason behind using an adaptive neighborhood selection procedure is to increase the diversification ability of metaheuristics by dynamically changing the neighborhood definitions. Since all the components but the neighborhood selection procedure are the same for TS/4 and TS/ANS, we compare the trajectories of TS/4 and

TS/ANS for a subset of instances to analyse the diversification capability of TS/ANS. The trajectories of Instances #48 and #50 are given in the leftmost and the rightmost parts of Figure 6.3 as the instances with $\omega = 150000$.

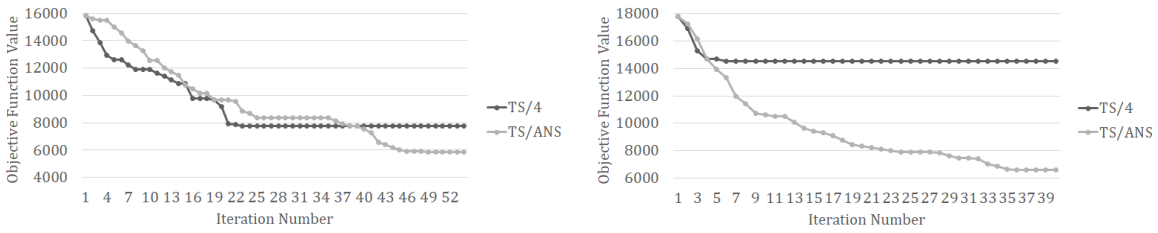


Figure 6.3. Comparison of trajectories, $\omega = 150000$: TS/4 and TS/ANS.

As indicated in Figure 6.3, the higher performance of TS/4 compared to that of TS/ANS can be seen in earlier iterations. This is expected because TS/4, as a local search algorithm, concentrates on intensification and exploits a basin of attraction. However, TS/ANS spends the earlier iterations on exploring other basins of attractions, and is therefore outperformed by TS/ANS. In later iterations, TS/4 is trapped at local optimal solutions and TS/ANS enjoys the exploration and returns better solutions eventually.

Similar patterns are observed in the trajectories of Instances #67 and #68, too. In Figure 6.4, one may find the corresponding trajectories of these instances as the subset of instances with $\omega = 200000$. Regarding this trajectory based analysis, we conclude that given a sufficient amount of time, TS/ANS is expected to outperform TS/4 due to its diversification capability.

6.2.2.2. Analysis of VNS/AR. In the ARBI procedure, embedded into the generic VNS to obtain VNS/AR, association rules are extracted at each iteration after the shaking step. Since the rules are extracted by the Apriori algorithm with respect to minimum confidence $c = 80\%$ and minimum support $s = 20\%$, there is always a possibility of obtaining no rules in an iteration.

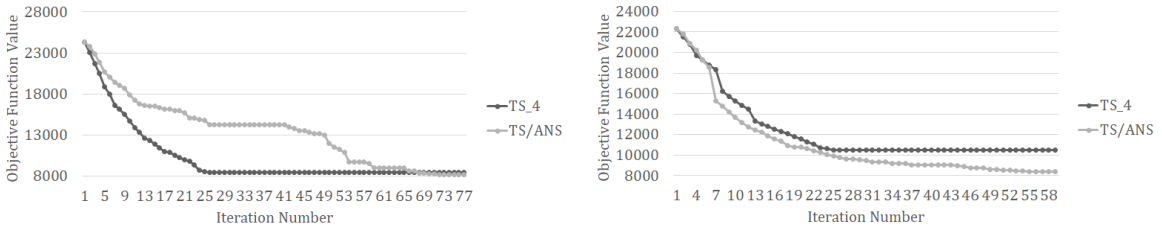


Figure 6.4. Comparison of trajectories, $\omega = 200000$: TS/4 and TS/ANS.

In Table 6.7, one may find a set of statistics related to the extracted rules of VNS/AR that runs on BOpt-RRC instances with $\omega_{new} = 0.02$. The first column gives the $\omega$ values, and the remaining ones are reserved for the statistics. In the second column, the number of shakings averaged over the instances of the same size is given. For example, in all five instances with $\omega = 50000$ and $\omega_{new} = 0.02$ there are 54.8 shakings on average. The next column shows the number of empty rules averaged over the instances of the same size. For instance, out of 54.8 Apriori algorithm calls, 1.6 of them returns no rule on average. The fourth column indicates the average rule size and the last column is reserved for sharing average number of distinct rules averaged over the instances of the same size.

The statistics in Table 6.7 reveal that, with larger $\omega$ values, the number of shakings reduces. This is actually expected because the total computational time devoted to the local search component in VNS/AR increases as $\omega$ increases. Since the number of empty rules are insignificant for all instances with $\omega_{new} = 0.02$, we are free to conclude that it is valid to compare VNS and VNS/AR for the sake of performance analysis. The average rule size also increases with larger $\omega$ values due to the necessity of larger number of existing camps to serve larger populations. An increase in the number of distinct rules with larger populations is also attributed to the same reasoning.

The statistics on the properties of extracted rules of VNS/AR obtained on BOpt-RRC instances with $\omega_{new} = 0.06$ are given in Table 6.8. A pattern similar to that in Table 6.7 can be seen through the columns. We can conclude that the performance

comparison of VNS and VNS/AR is valid for BOpt-RRC instances with $\omega_{new} = 0.06$, too.

Table 6.7. Properties of extracted rules for $\omega_{new} = 0.02$.

| $\omega$ | **#Shaking** | **#EmptyRules** | **RuleSize** | **#DistinctRules** |
|---|---|---|---|---|
| 50000 | 54.8 | 1.6 | 1.0 | 3.0 |
| 100000 | 18.4 | 0.0 | 1.1 | 3.8 |
| 150000 | 6.8 | 0.0 | 1.5 | 4.4 |
| 200000 | 6.2 | 0.0 | 1.6 | 4.4 |
| Average: | 21.6 | 0.4 | 1.3 | 3.9 |

Table 6.8. Properties of extracted rules for $\omega_{new} = 0.06$.

| $\omega$ | **#Shaking** | **#EmptyRules** | **RuleSize** | **#DistinctRules** |
|---|---|---|---|---|
| 50000 | 61.6 | 0.6 | 1.0 | 2.6 |
| 100000 | 12.6 | 0.0 | 1.3 | 3.4 |
| 150000 | 7.8 | 0.0 | 1.3 | 4.4 |
| 200000 | 5.2 | 0.0 | 1.7 | 4.4 |
| Average: | 21.8 | 0.2 | 1.3 | 3.7 |

In Table 6.9, one may find the statistics on the ARBI calls in VNS/AR to solve the BOpt-RRC instances with $\omega_{new} = 0.10$. Though a pattern similar pattern to that in both Tables 6.7 and 6.8 can be detected, there are two exceptions in Table 6.9. First, all of Apriori algorithm calls return nonempty rules. Second, the number of distinct rules extracted for the instances with $\omega = 200000$ is smaller, comparatively.

Table 6.10 is reserved for the VNS/AR statistics over the BOpt-RRC instances with $\omega_{new} = 0.20$. The only empty rules are seen in the instances with $\omega = 50000$, too. In this sense, there is nothing new in terms of patterns seen in the other instances with $\omega_{new} = 0.02, 0.06, 0.10$. However, there is a decreasing trend in the number of distinct rules with larger $\omega$ values, which is not seen in the remaining instances. During the comparative analysis, we briefly discuss the possible effect of this trend and denote a research question in Section 6.2.3.

Table 6.9. Properties of extracted rules for $\omega_{new} = 0.10$.

| $\omega$ | **#Shaking** | **#EmptyRules** | **RuleSize** | **#DistinctRules** |
|---|---|---|---|---|
| 50000 | 35.4 | 0.0 | 1.1 | 4.0 |
| 100000 | 15.2 | 0.0 | 1.3 | 4.4 |
| 150000 | 7.0 | 0.0 | 1.6 | 4.6 |
| 200000 | 4.6 | 0.0 | 1.8 | 2.6 |
| Average: | 15.6 | 0.0 | 1.4 | 3.9 |

Table 6.10. Properties of extracted rules for $\omega_{new} = 0.20$.

| $\omega$ | **#Shaking** | **#EmptyRules** | **RuleSize** | **#DistinctRules** |
|---|---|---|---|---|
| 50000 | 32.2 | 2.4 | 0.9 | 3.0 |
| 100000 | 9.0 | 0.0 | 1.2 | 3.2 |
| 150000 | 4.0 | 0.0 | 1.9 | 1.4 |
| 200000 | 2.2 | 0.0 | 1.4 | 1.0 |
| Average: | 11.9 | 0.6 | 1.3 | 2.2 |

### 6.2.3. Analysis of Matheuristics

The performance of the matheuristics on BOpt-RRC instances with $\omega_{new} = 0.02$ is displayed in Table 6.11. The PD columns under all matheuristics are the PD values averaged over the instances of the same size, i.e., average of five instances. For all the instances of the same size, the columns #BS gives the total number of instances in which the best found solutions (or optimal solutions for $\omega = 50000$) are returned.

The results in Table 6.11 indicate that the performance of both TS implementations deteriorates with increasing $\omega$ values, i.e., existing refugee populations. However, we cannot detect a certain performance trend in VNS and VNS/AR with changing $\omega$ values. Overall, TS/4 is the worst method and VNS/AR outperforms all other matheuristics significantly. The computational results reveal that the average #BS values are equal to each other for TS/ANS and VNS. Although the average PD value is lower for VNS, we cannot claim the superiority of VNS over TS/ANS. In summary, the adaptive neighborhood selection and ARBI procedures perform better.

Table 6.11. Performance comparison on the instances with $\omega_{new} = 0.02$.

| | **TS/4** | | **TS/ANS** | | **VNS** | | **VNS/AR** | |
|---|---|---|---|---|---|---|---|---|
| $\omega$ | PD | #BS | PD | #BS | PD | #BS | PD | #BS |
| 50 | 8.1% | 3 | 7.0% | 3 | 3.0% | 3 | 0.0% | 5 |
| 100 | 42.1% | 2 | 34.0% | 2 | 68.7% | 0 | 3.1% | 2 |
| 150 | 128.0% | 0 | 40.7% | 1 | 2.8% | 2 | 6.6% | 2 |
| 200 | 171.2% | 0 | 22.5% | 0 | 12.6% | 1 | 0.0% | 5 |
| Average: | 87.3% | 1.25 | 26.1% | 1.50 | 21.8% | 1.50 | 2.4% | 3.50 |

A comparison of all matheuristics over the instances with $\omega_{new} = 0.06$ is given in Table 6.12. In the overall, TS/ANS significantly improves the results of TS and the positive effect of using the adaptive neighborhood selection procedure is evident in the table. However, both VNS implementations' average PD values lead us to the superiority of these implementations on the BOpt-RRC instances with $\omega_{new} = 0.06$. Adding the ARBI procedure to VNS significantly increases the performance in terms of #BS. In addition, the average PD value of VNS/AR is lower than that of VNS. By the comparison of TS/4 with TS/ANS and VNS with VNS/AR, we conclude that the proposed data-driven LS matheuristics outperform the generic versions.

A pattern similar to that in Table 6.12, is also seen in the performance comparison of all matheuristics over the instances with $\omega_{new} = 0.10$ given in Table 6.13. The superiority of VNS/AR is evident in the table in terms of both average PD and #BS. Although the performances of both TS implementations are significantly below those of VNS implementations, it is worth noting that the adaptive neighborhood selection procedure considerably improves the performance of the generic TS implementation.

The performance of the matheuristics on BOpt-RRC instances with $\omega_{new} = 0.20$ are given in Table 6.14. In all instances, we select the upper limits on the capacity increase of existing camps as 20% of the existing capacity values. Since the number of new refugees is generated with uniform random distribution with mean $\omega \times \omega_{new}$, the *expected* number of new refugees match with the limit on the capacity increase. Due to this fact, we expect all the matheuristic methods to show similar performance, and

this is what happens in practice. Compared to the results of instances with $\omega_{new} = 0.02, 0.06, 0.10$, the performance of all matheuristic methods are closer to each other. However, VNS/AR still outperforms all other methods.

Table 6.12. Performance comparison on the instances with $\omega_{new} = 0.06$.

| | **TS/4** | | **TS/ANS** | | **VNS** | | **VNS/AR** | |
|---|---|---|---|---|---|---|---|---|
| $\omega$ | PD | #BS | PD | #BS | PD | #BS | PD | #BS |
| 50 | 40.0% | 0 | 9.8% | 2 | 1.1% | 4 | 1.1% | 4 |
| 100 | 132.8% | 0 | 30.9% | 1 | 13.0% | 1 | 17.1% | 3 |
| 150 | 125.9% | 0 | 46.2% | 0 | 3.4% | 2 | 2.2% | 4 |
| 200 | 60.2% | 1 | 23.9% | 1 | 9.7% | 2 | 2.4% | 3 |
| Average: | 89.7% | 0.25 | 27.7% | 1.00 | 6.8% | 2.25 | 5.7% | 3.50 |

Table 6.13. Performance comparison on the instances with $\omega_{new} = 0.10$.

| | **TS/4** | | **TS/ANS** | | **VNS** | | **VNS/AR** | |
|---|---|---|---|---|---|---|---|---|
| $\omega$ | PD | #BS | PD | #BS | PD | #BS | PD | #BS |
| 50 | 68.3% | 1 | 2.6% | 1 | 1.5% | 2 | 1.2% | 3 |
| 100 | 33.8% | 0 | 28.7% | 0 | 6.0% | 1 | 1.0% | 4 |
| 150 | 66.8% | 0 | 28.2% | 0 | 2.0% | 3 | 0.8% | 3 |
| 200 | 58.8% | 0 | 13.9% | 2 | 7.1% | 1 | 1.6% | 2 |
| Average: | 56.9% | 0.25 | 18.4% | 0.75 | 4.2% | 1.75 | 1.1% | 3.00 |

Please recall the discussion based on the results in Table 6.10. For all instances except the ones with $\omega_{new} = 0.20$, both the rule size and the number of distinct rules increase with larger $\omega$ values. Also, the superiority of VNS/AR is more prominent for all instances but the ones with $\omega_{new} = 0.20$. Regarding this observation, we may claim that there is a relationship between VNS/AR performance and the properties of the extracted rules, intuitively. Hence, a further study is needed to analyse this relationship and we keep this study as a future research opportunity.

Table 6.14. Performance comparison on the instances with $\omega_{new} = 0.20$.

| | **TS/4** | | **TS/ANS** | | **VNS** | | **VNS/AR** | |
|---|---|---|---|---|---|---|---|---|
| $\omega$ | PD | #BS | PD | #BS | PD | #BS | PD | #BS |
| 50 | 6.3% | 1 | 9.9% | 2 | 3.1% | 2 | 3.1% | 2 |
| 100 | 7.7% | 1 | 3.7% | 1 | 1.4% | 1 | 0.6% | 3 |
| 150 | 0.7% | 1 | 0.8% | 2 | 0.3% | 4 | 0.5% | 3 |
| 200 | 1.8% | 0 | 0.6% | 4 | 0.5% | 2 | 0.7% | 2 |
| Average: | 4.1% | 0.75 | 3.8% | 2.25 | 1.3% | 2.25 | 1.2% | 2.50 |

### 6.2.4. Rationalization of the Bilevel Approach

The decision maker at the ULP of the BOpt-RRC aims to readjust the refugee camp network with smallest change in the current network configuration. To this end, we incorporate a set of penalty terms into the upper-level objective function. Alongside with the penalties, we also add the total cost of capacity increase. In this section, we briefly analyse the effect of penalties on reconfiguration decisions by devising a single level optimization model (SOpt-RRC) for the problem. The problem is obtained by (i) discarding the penalty terms and adding the total cost of capacity increase to the LLP objective function at (5.5), (ii) discarding constraints (5.2) and adding constraints (5.3) and (5.4) into the constraint set of the LLP and (iii) manipulating constraints (5.9) in the LLP. All of the set, parameter and variable definitions are directly adopted from BOpt-RRC. The proposed single-level model is given as

$$\min \quad \sum_{e} \alpha_e A_e + \sum_{c\in C}\sum_{h\in H} \chi_{ch} X_{ch} + \sum_{e\in E}\sum_{h\in H} \chi_{eh} X_{eh} + \sum_{c\in C} \xi_c Y_c + \\ + \sum_{s\in S}\sum_{e\in E} \phi_{se} F_{se} + \sum_{s\in S}\sum_{c\in C} \phi_{sc} F_{sc}, \tag{6.2}$$

$$\text{s.t.} \quad 0 \le A_e \le \bar{a}_e \qquad e \in E, \tag{6.3}$$

$$T_e \le \bar{Q}_e + A_e \qquad e \in E, \tag{6.4}$$

$$A_e \in \mathbb{Z} \cup \{0\} \qquad e \in E. \tag{6.5}$$

$$\text{Constraints (5.6)–(5.8) and (5.10)–(5.22).} \tag{6.6}$$

The objective function at (6.2) aims to minimize the total cost of capacity increase, hospital assignments, building new refugee camps and refugee flows to existing and/or candidate camps. Constraints (6.3) ensure that the total increased capacity of each camp $e \in E$ does not exceed the corresponding upper limit on the increase $\bar{a}_e$. Domains of the capacity increase variables are defined by constraints (6.5). Remaining constraints are adopted from the BOpt-RRC formulation and they are used as is.

Let $s^*_{B_i}$ be the optimal solution of the BOpt-RRC for instance $i$, $f(s)$ be the objective function of the ULP of the BOpt-RRC and $f(s^*_{B_i})$ be the corresponding optimal objective function value. Also, let $s^*_{S_i}$ be the optimal solution of the SOpt-RRC for instance $i$. Here, we analyse the effect of the penalty terms on reconfiguration decisions by comparing the objective values $f(s^*_{B_i})$ and $f(s^*_{S_i})$. Percent deviations of the SOpt-RRC solutions from the BOpt-RRC solutions are given in Figure 6.5 in which the percent deviation is calculated by $100 \times (f(s^*_{S_i}) - f(s^*_{B_i}))/f(s^*_{B_i})$ for each instance $i$. The computational study is based on the BOpt-RRC instances with $\omega = 50000$, since these are the only instances with optimal solutions at hand. The y-axis gives the percent deviation where the x-axis gives the instance number in the figure.

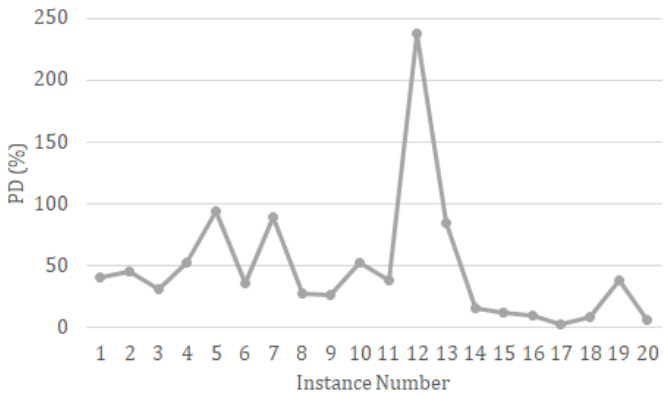


Figure 6.5. Percent deviation of the SOpt-RRC solutions.

As indicated by the figure, the SOpt-RRC solutions return objective function values larger than the BOpt-RRC solutions for all instances as expected, since all

solutions are evaluated by the objective function of the ULP of the BOpt-RRC. Percent deviations averaged over the instances of the same size are calculated as 52.9% for $\omega_{new} = 0.02$, 46.4% for $\omega_{new} = 0.06$, 77.6% for $\omega_{new} = 0.10$ and 13.5% for $\omega_{new} = 0.20$ and the overall percent deviation is 47.6%. These results indicate that the single-level formulation, in which the objective of the smallest change is discarded, increase the total cost of capacity increase and penalty almost by half on the average comparing to the bilevel formulation. Hence, we conclude that the proposed bilevel formulation significantly reduces the readjustment costs and should be preferred in NDPs where the smallest change in the current configuration is prioritised.

# 7. CONCLUSION

## 7.1. Summary of the Contributions

In this thesis, two bilevel NDP problems are introduced. First, in the RI-NDPLD as a network interdiction problem, we model the strategic flight network design decisions of an SAC aiming to enter an airline market in which a set of ACs are already operating. The problem is relevant in modelling the competitive environment of the airline market. In the second problem, the BOpt-RRC, we model the readjustment decisions of an existing refugee camp network configuration in case of new refugee flows. The proposed model has a practical value due to the fact that the total number of refugees are in an increasing trend and the refugee camps are no more temporary. We propose BMIP formulations for both problems such that the LLPs are NDP variants. In particular, RI-NDPLD is a contribution to the network interdiction literature and the BOpt-RRC contributes to humanitarian logistics literature as a bilevel NDP.

From a methodological point of view, we propose a set of ML embedded data-driven local search matheuristics to solve both problems heuristically. Metaheuristic methods are already proposed for many COPs. However, in the recent literature, there is an increasing interest on improving the search capabilities of metaheuristic methods by exploiting ML models. ML models are used for a set of tasks such as algorithm selection, fitness evaluation, initialization, evolution operators, parameter setting and cooperation of multiple metaheuristics. In this study, we employ a set of ML models and embed them into a set of TS and VNS metaheuristics. In the TS implementation for the RI-NDPLD we rely on RF for fitness evaluation to reduce the total computational time of objective function evaluations in the form of a candidate solution set generation procedure. We also embed RF models into a perturbation scheme as a substitute of the long-term memory usage. For the BOpt-RRC, we implement a TS and a VNS matheuristic. In the TS implementation, we use a value function based adaptive neighborhood selection procedure and in the VNS implementation, we use the Apriori

algorithm to generate a set of association rules to find the common components in good solutions and we rely on these rules to find initial solutions for the local search procedures in the VNS framework.

The generic TS heuristic implemented for the RI-NDPLD does not perform well on large sized instances. The main reason behind this result is that the method spends significant amount of time to evaluate all the neighboring solutions of a current solution at each iteration, and hence it terminates after a few iterations. Adding a procedure into the TS to prune a subset of neighboring solutions by bounds leads to more number of iterations. Thanks to this, a larger portion of the solution space is explored, and hence the performance of the TS is increased. However, the computational study reveals that the performance increase depends on the quality of the bounds and in the larger instances the bounds are weak in a sense that a smaller portions of all neighboring solutions are pruned. In the data-driven sorting procedure, a certain fraction of neighboring solutions to be evaluated is guaranteed and the number of objective function evaluations is fixed at each iteration. Hence, the issue of the weak bounds is resolved by incorporating the data-driven sorting procedure into the TS implementation. However, due to the larger prediction errors of the RF models trained in earlier iterations, the search may be directed into non-promising areas. To recover the problems of weak bounds and the prediction errors, we simultaneously incorporate both procedures into the TS implementation and obtain a more robust method. By employing this TS heuristic into a data-driven restart diversification scheme, we finalize the implementation and conclude that the proposed TS heuristic with restart diversification is capable of solving practical size RI-NDPLD instances.

We implement a set of generic LS matheuristics to solve the BOpt-RRC and enhanced their search capabilities by embedding a couple of data-driven procedures into these generic implementations. First, we define a new neighborhood definition for the BOpt-RRC and propose a generic TS method. Second, we devise an adaptive neighborhood selection procedure and incorporated into the generic TS. To do so, we resolve the premature convergence issue of TS by dynamically changing the neighborhood def-

inition with a value function based approach. The computational study revealed that, in almost all instances, the generic TS is outperformed by the implementation with the adaptive procedure. Third, to obtain a more exploration-oriented LS matheuristic, we implement a generic VNS to solve the BOpt-RRC in which the local search component is the TS with adaptive neighborhood selection procedure. Regarding the computational study, the generic VNS performs better than the TS implementation with the adaptive procedure in virtually all BOpt-RRC instances. Later, we devise an association rule based injection procedure to add a set of *good* components into the initial solutions obtained by the usual random shaking. The results of the overall comparison of all matheuristics indicate that the VNS implementation with data-driven injection procedure outperforms all other methods in terms of percentage deviations and total number of instances in which the best found solutions are returned.

## 7.2. Future Research Directions

As a future research direction, we plan to work on increasing the computational efficiency of solving the LLP by devising problem-specific heuristics. The benefit of such a research study is to eliminate more neighboring solutions and thus increase the pruning performance by obtaining better upper bounds. Another research opportunity could be using data-driven procedures in developing metaheuristic methods to solve various bilevel programming models that have a difficult combinatorial optimization problem in the lower level which needs to be solved many times. Since resolving the LLP significantly increases the computational effort, these generic data-driven procedures, i.e., fitting a regression model to make predictions on the objective value of the LLP given the decision in the ULP and using feature importance values calculated by tree based regression models for restart diversification, can have a dramatic impact as they can reduce the neighborhood size by eliminating solutions.

The studies on the BOpt-RRC also lead us to a set of future research questions. In terms modelling, BOpt-RRC can be enhanced by including a set of additional decisions such as closing a refugee camp, settling/resettling of existing refugees and explicit

inclusion of infrastructural resources into capacity increase in terms of budgetary restrictions. We also have a set of future research directions from a methodological perspective. The adaptive neighborhood selection procedure in the TS implementation can be seen as a version of the single-state multi-armed bandit problem as an RL model. We plan to investigate the usage of more elaborated RL models in the adaptive neighborhood selection procedure for the sake of further performance increase. The experimental results indicate that there is a possible relationship with the diversification capability of the VNS implementation with association rule based injection and the number of distinct rules extracted. We aim to analyse this relationship thoroughly to obtain a more robust version of the aforementioned VNS implementation.

# REFERENCES


1. Johnson, D. S., J. K. Lenstra and A. R. Kan, "The Complexity of the Network Design Problem", *Networks*, Vol. 8, No. 4, pp. 279–285, 1978.

2. O'Kelly, M. E. and H. J. Miller, "The Hub Network Design Problem: A Review and Synthesis", *Journal of Transport Geography*, Vol. 2, No. 1, pp. 31–40, 1994.

3. Keskin, M. E., I. K. Altınel, N. Aras and C. Ersoy, "Wireless Sensor Network Design by Lifetime Maximisation: An Empirical Evaluation of Integrating Major Design Issues and Sink Mobility", *International Journal of Sensor Networks*, Vol. 20, No. 3, pp. 131–146, 2016.

4. Crainic, T. G., M. Hewitt, M. Toulouse and D. M. Vu, "Scheduled Service Network Design with Resource Acquisition and Management", *EURO Journal on Transportation and Logistics*, Vol. 7, No. 3, pp. 277–309, 2018.

5. Katayama, N., "MIP Neighborhood Search Heuristics for a Service Network Design Problem with Design-Balanced Requirements", *Journal of Heuristics*, Vol. 26, No. 4, pp. 1–28, 2020.

6. Mubarak, M., H. Üster, K. Abdelghany and M. Khodayar, "Strategic Network Design and Analysis for In-Motion Wireless Charging of Electric Vehicles", *Transportation Research Part E: Logistics and Transportation Review*, Vol. 145, p. 102179, 2021.

7. Karsu, Ö., B. Y. Kara, E. Akkaya and A. Ozel, "Clean Water Network Design for Refugee Camps", *Networks and Spatial Economics*, Vol. 21, No. 1, pp. 1–24, 2021.

8. Arslan, O., G. Ç. Kumcu, B. Y. Kara and G. Laporte, "The Location and Location-Routing Problem for the Refugee Camp Network Design", *Transporta-*

*tion Research Part B: Methodological*, Vol. 143, pp. 201–220, 2021.

9. von Stackelberg, H., *The Theory of the Market Economy*, Oxford University Press, Oxford, 1952.

10. Karimi-Mamaghan, M., M. Mohammadi, P. Meyer, A. M. Karimi-Mamaghan and E.-G. Talbi, "Machine Learning at the Service of Meta-Heuristics for Solving Combinatorial Optimization Problems: A State-of-the-Art", *European Journal of Operational Research*, Vol. 296, No. 2, pp. 393–422, 2022.

11. Kleinert, T., M. Labbé, I. Ljubić and M. Schmidt, "A Survey on Mixed-Integer Programming Techniques in Bilevel Optimization", *EURO Journal on Computational Optimization*, Vol. 9, p. 100007, 2021.

12. Smith, J. C. and Y. Song, "A Survey of Network Interdiction Models and Algorithms", *European Journal of Operational Research*, Vol. 283, No. 3, pp. 797–811, 2020.

13. Wollmer, R., "Removing Arcs from a Network", *Operations Research*, Vol. 12, No. 6, pp. 934–940, 1964.

14. Wood, R. K., "Deterministic Network Interdiction", *Mathematical and Computer Modelling*, Vol. 17, No. 2, pp. 1–18, 1993.

15. Hochba, D. S., "Approximation Algorithms for NP-Hard Problems", *ACM Sigact News*, Vol. 28, No. 2, pp. 40–52, 1997.

16. Talbi, E.-G., *Metaheuristics: From Design to Implementation*, John Wiley & Sons, New Jersey, 2009.

17. Glover, F., "Tabu Search—Part I", *ORSA Journal on Computing*, Vol. 1, No. 3, pp. 190–206, 1989.

18. Knox, J., "Tabu Search Performance on the Symmetric Traveling Salesman Problem", *Computers & Operations Research*, Vol. 21, No. 8, pp. 867–876, 1994.

19. Schneider, J. J. and S. Kirkpatrick, "Tabu Search Applied to TSP", *Stochastic Optimization*, pp. 441–447, Springer, Berlin, 2006.

20. Cordeau, J.-F. and M. Maischberger, "A Parallel Iterated Tabu Search Heuristic for Vehicle Routing Problems", *Computers & Operations Research*, Vol. 39, No. 9, pp. 2033–2050, 2012.

21. Barbarosoglu, G. and D. Ozgur, "A Tabu Search Algorithm for the Vehicle Routing Problem", *Computers & Operations Research*, Vol. 26, No. 3, pp. 255–270, 1999.

22. Crainic, T. G. and M. Gendreau, "Cooperative Parallel Tabu Search for Capacitated Network Design", *Journal of Heuristics*, Vol. 8, No. 6, pp. 601–627, 2002.

23. Xie, C. and M. A. Turnquist, "Lane-Based Evacuation Network Optimization: An Integrated Lagrangian Relaxation and Tabu Search Approach", *Transportation Research Part C: Emerging Technologies*, Vol. 19, No. 1, pp. 40–63, 2011.

24. Aksen, D. and N. Aras, "A Matheuristic for Leader-Follower Games Involving Facility Location-Protection-Interdiction Decisions", *Metaheuristics for Bi-level Optimization*, pp. 115–151, Springer, Berlin, 2013.

25. Akbari-Jafarabadi, M., R. Tavakkoli-Moghaddam, M. Mahmoodjanloo and Y. Rahimi, "A Tri-Level r-Interdiction Median Model for a Facility Location Problem Under Imminent Attack", *Computers & Industrial Engineering*, Vol. 114, pp. 151–165, 2017.

26. Sadati, M. E. H., D. Aksen and N. Aras, "The r-Interdiction Selective Multi-Depot Vehicle Routing Problem", *International Transactions in Operational Research*, Vol. 27, No. 2, pp. 835–866, 2020.

27. Mladenović, N. and P. Hansen, "Variable Neighborhood Search", *Computers & Operations Research*, Vol. 24, No. 11, pp. 1097–1100, 1997.

28. Hansen, P., N. Mladenović, J. Brimberg and J. A. M. Pérez, "Variable Neighborhood Search", *Handbook of Metaheuristics*, pp. 57–97, Springer, Cham, 2019.

29. Cheimanoff, N., F. Fontane, M. N. Kitri and N. Tchernev, "Exact and Heuristic Methods for the Integrated Berth Allocation and Specific Time-Invariant Quay Crane Assignment Problems", *Computers & Operations Research*, Vol. 141, p. 105695, 2022.

30. Luo, Q., Y. Rao, X. Guo and B. Du, "A Biased Genetic Algorithm Hybridized with VNS for the Two-Dimensional Knapsack Packing Problem with Defects", *Applied Soft Computing*, Vol. 118, p. 108479, 2022.

31. Soares, L. C. and M. A. Carvalho, "Application of a Hybrid Evolutionary Algorithm to Resource-Constrained Parallel Machine Scheduling with Setup Times", *Computers & Operations Research*, Vol. 139, p. 105637, 2022.

32. Olmez, O. B., C. Gultekin, B. Balcik, A. Ekici and O. Ö. Özener, "A Variable Neighborhood Search Based Matheuristic for a Waste Cooking Oil Collection Network Design problem", *European Journal of Operational Research*, Vol. 302, No. 1, pp. 187–202, 2022.

33. Tawfik, C., B. Gendron and S. Limbourg, "An Iterative Two-Stage Heuristic Algorithm for a Bilevel Service Network Design and Pricing Model", *European Journal of Operational Research*, Vol. 300, No. 2, pp. 512–526, 2022.

34. Wu, Y., Z. Chen, H. Gong, Q. Feng, Y. Chen and H. Tang, "Defender–Attacker–Operator: Tri-Level Game-Theoretic Interdiction Analysis of Urban Water Distribution Networks", *Reliability Engineering & System Safety*, Vol. 214, p. 107703, 2021.

35. Samanta, S., T. Mohandass, G. Sen and S. K. Ghosh, "A VNS-Based Metaheuristic Approach for Escape Interdiction on Transportation Networks", *Computers & Industrial Engineering*, Vol. 169, p. 108253, 2022.

36. Alpaydin, E., *Machine Learning*, MIT Press, Cambridge, MA, 2021.

37. Sutton, R. S. and A. G. Barto, *Reinforcement Learning: An Introduction*, MIT Press, Cambridge, MA, 2018.

38. Quinlan, J. R., "Learning Decision Tree Classifiers", *ACM Computing Surveys (CSUR)*, Vol. 28, No. 1, pp. 71–72, 1996.

39. Breiman, L., "Random Forests", *Machine Learning*, Vol. 45, No. 1, pp. 5–32, 2001.

40. MacQueen, J., "Classification and Analysis of Multivariate Observations", *5th Berkeley Symposium on Mathematical Statistics and Probability*, pp. 281–297, 1967.

41. Hartigan, J. A. and M. A. Wong, "Algorithm AS 136: A k-Means Clustering Algorithm", *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, Vol. 28, No. 1, pp. 100–108, 1979.

42. Agrawal, R., H. Mannila, R. Srikant, H. Toivonen, A. I. Verkamo *et al.*, "Fast Discovery of Association Rules.", *Advances in Knowledge Discovery and Data Mining*, Vol. 12, No. 1, pp. 307–328, 1996.

43. Agrawal, R., R. Srikant *et al.*, "Fast Algorithms for Mining Association Rules", *Proc. 20th Int. Conf. Very Large Data Bases, VLDB*, Vol. 1215, pp. 487–499, Citeseer, 1994.

44. Tanınmış, K., N. Aras and I. Altınel, "Influence Maximization with Deactivation in Social Networks", *European Journal of Operational Research*, Vol. 278, No. 1,

pp. 105–119, 2019.

45. Tanınmış, K., N. Aras, İ. K. Altınel and E. Güney, "Minimizing the Misinformation Spread in Social Networks", *IISE Transactions*, Vol. 52, No. 8, pp. 850–863, 2020.

46. Martelli, E., M. Freschini and M. Zatti, "Optimization of Renewable Energy Subsidy and Carbon Tax for Multi Energy Systems Using Bilevel Programming", *Applied Energy*, Vol. 267, p. 115089, 2020.

47. Wogrin, S., S. Pineda and D. A. Tejada-Arango, "Applications of Bilevel Optimization in Energy and Electricity Markets", *Bilevel Optimization*, pp. 139–168, Springer, Cham, 2020.

48. Candler, W., J. Fortuny-Amat and B. McCarl, "The Potential Role of Multilevel Programming in Agricultural Economics", *American Journal of Agricultural Economics*, Vol. 63, No. 3, pp. 521–531, 1981.

49. Küçükaydin, H., N. Aras and I. K. Altınel, "Competitive Facility Location Problem with Attractiveness Adjustment of the Follower: A Bilevel Programming Model and Its Solution", *European Journal of Operational Research*, Vol. 208, No. 3, pp. 206–220, 2011.

50. DeNegre, S., *Interdiction and Discrete Bilevel Linear Programming*, Ph.D. Thesis, Lehigh University, Bethlehem, PA, 5 2011.

51. Sinha, A., P. Malo and K. Deb, "A Review on Bilevel Optimization: From Classical to Evolutionary Approaches and Applications", *IEEE Transactions on Evolutionary Computation*, Vol. 22, No. 2, pp. 276–295, 2017.

52. Kalashnikov, V. V., S. Dempe, G. A. Pérez-Valdés, N. I. Kalashnykova and J.-F. Camacho-Vallejo, "Bilevel Programming and Applications", *Mathematical Problems in Engineering*, pp. 1–16, 2015.

53. Farvaresh, H. and M. M. Sepehri, "A Branch and Bound Algorithm for Bi-Level Discrete Network Design Problem", *Networks and Spatial Economics*, Vol. 13, No. 1, pp. 67–106, 2013.

54. Yu, B., L. Kong, Y. Sun, B. Yao and Z. Gao, "A Bi-Level Programming for Bus Lane Network Design", *Transportation Research Part C: Emerging Technologies*, Vol. 55, pp. 310–327, 2015.

55. Fontaine, P. and S. Minner, "A Dynamic Discrete Network Design Problem for Maintenance Planning in Traffic Networks", *Annals of Operations Research*, Vol. 253, No. 2, pp. 757–772, 2017.

56. Lin, B., C. Liu, H. Wang and R. Lin, "Modeling the Railway Network Design Problem: A Novel Approach to Considering Carbon Emissions Reduction", *Transportation Research Part D: Transport and Environment*, Vol. 56, pp. 95–109, 2017.

57. Di, Z., L. Yang, J. Qi and Z. Gao, "Transportation Network Design for Maximizing Flow-Based Accessibility", *Transportation Research Part B: Methodological*, Vol. 110, pp. 209–238, 2018.

58. Msigwa, R. E., Y. Lu and L.-W. Zhang, "A Perturbation-Based Approach for Continuous Network Design Problem with Link Capacity Expansion", *International Journal of Operational Research*, Vol. 37, No. 1, pp. 105–134, 2020.

59. Zhou, Z., M. Yang, F. Sun, Z. Wang and B. Wang, "A Continuous Transportation Network Design Problem with the Consideration of Road Congestion Charging", *Sustainability*, Vol. 13, No. 13, pp. 1–16, 2021.

60. Kara, B. Y. and V. Verter, "Designing a Road Network for Hazardous Materials Transportation", *Transportation Science*, Vol. 38, No. 2, pp. 188–196, 2004.

61. Fontaine, P. and S. Minner, "Benders Decomposition for the Hazmat Transport

Network Design Problem", *European Journal of Operational Research*, Vol. 267, No. 3, pp. 996–1002, 2018.

62. Gutjahr, W. J. and N. Dzubur, "Bi-Objective Bilevel Optimization of Distribution Center Locations Considering User Equilibria", *Transportation Research Part E: Logistics and Transportation Review*, Vol. 85, pp. 1–22, 2016.

63. Sahinyazan, F. G., M.-È. Rancourt and V. Verter, "Food Aid Modality Selection Problem", *Production and Operations Management*, Vol. 30, No. 4, pp. 965–983, 2021.

64. Takebayashi, M. and A. Kanafani, "Network Competition in Air Transportation Markets: Bi-Level Approach", *Research in Transportation Economics*, Vol. 13, pp. 101–119, 2005.

65. Yang, T.-H., C.-H. Tang and H.-C. Hsiao, "Strategic Airline Network Design Problem in a Duopolistic Market", *Transportation Planning and Technology*, Vol. 43, No. 6, pp. 586–601, 2020.

66. Abdelghany, A., K. Abdelghany and F. Azadian, "Airline Flight Schedule Planning Under Competition", *Computers & Operations Research*, Vol. 87, pp. 20–39, 2017.

67. Takebayashi, M., "Managing the Multiple Airport System by Coordinating Short/Long-Haul Flights", *Journal of Air Transport Management*, Vol. 22, pp. 16–20, 2012.

68. Parvaresh, F., S. M. Husseini, S. Golpayegany and B. Karimi, "Hub Network Design Problem in the Presence of Disruptions", *Journal of Intelligent Manufacturing*, Vol. 25, No. 4, pp. 755–774, 2014.

69. Church, R. L., M. P. Scaparra and R. S. Middleton, "Identifying Critical Infrastructure: The Median and Covering Facility Interdiction Problems", *Annals of*

*the Association of American Geographers*, Vol. 94, No. 3, pp. 491–502, 2004.

70. Ullmert, T., S. Ruzika and A. Schöbel, "On the p-Hub Interdiction Problem", *Computers & Operations Research*, Vol. 124, p. 105056, 2020.

71. Caprara, A., M. Carvalho, A. Lodi and G. J. Woeginger, "Bilevel Knapsack with Interdiction Constraints", *INFORMS Journal on Computing*, Vol. 28, No. 2, pp. 319–333, 2016.

72. Fischetti, M., I. Ljubić, M. Monaci and M. Sinnl, "Interdiction Games and Monotonicity with Application to Knapsack Problems", *INFORMS Journal on Computing*, Vol. 31, No. 2, pp. 390–410, 2019.

73. Mahdavi Pajouh, F., V. Boginski and E. L. Pasiliao, "Minimum Vertex Blocker Clique Problem", *Networks*, Vol. 64, No. 1, pp. 48–64, 2014.

74. Zenklusen, R., "Matching Interdiction", *Discrete Applied Mathematics*, Vol. 158, No. 15, pp. 1676–1690, 2010.

75. Wollmer, R. D., "Algorithms for Targeting Strikes in a Lines-of-Communication Network", *Operations Research*, Vol. 18, No. 3, pp. 497–515, 1970.

76. McMasters, A. W. and T. M. Mustin, "Optimal Interdiction of a Supply Network", *Naval Research Logistics Quarterly*, Vol. 17, No. 3, pp. 261–268, 1970.

77. Israeli, E. and R. K. Wood, "Shortest-Path Network Interdiction", *Networks: An International Journal*, Vol. 40, No. 2, pp. 97–111, 2002.

78. Lim, C. and J. C. Smith, "Algorithms for Discrete and Continuous Multicommodity Flow Network Interdiction Problems", *IIE Transactions*, Vol. 39, No. 1, pp. 15–26, 2007.

79. Bayrak, H. and M. Bailey, "Shortest Path Network Interdiction with Asymmetric

Information", *Networks: An International Journal*, Vol. 52, No. 3, pp. 133–140, 2008.

80. Salmeron, J., K. Wood and R. Baldick, "Analysis of Electric Grid Security Under Terrorist Threat", *IEEE Transactions on Power Systems*, Vol. 19, No. 2, pp. 905–912, 2004.

81. Arroyo, J. M. and F. D. Galiana, "On the Solution of the Bilevel Programming Formulation of the Terrorist Threat Problem", *IEEE Transactions on Power Systems*, Vol. 20, No. 2, pp. 789–797, 2005.

82. Motto, A., J. Arroyo and F. Galiana, "MILP for the Analysis of Electric Grid Security Under Disruptive Threat", *IEEE Transactions on Power Systems*, Vol. 20, No. 3, pp. 1357–1365, 2005.

83. Salmerón, J., K. Wood and R. Baldick, "Worst-Case Interdiction Analysis of Large-Scale Electric Power Grids", *IEEE Transactions on Power Systems*, Vol. 24, No. 1, pp. 96–104, 2009.

84. Cappanera, P. and M. Scaparra, "Optimal Allocation of Protective Resources in Shortest-Path Networks", *Transportation Science*, Vol. 45, No. 1, pp. 64–80, 2011.

85. Alguacil, N., A. Delgadillo and J. Arroyo, "A Trilevel Programming Approach for Electric Grid Defense Planning", *Computers & Operations Research*, Vol. 41, No. 1, pp. 282–292, 2014.

86. Lozano, L., J. Smith and M. Kurz, "Solving the Traveling Salesman Problem with Interdiction and Fortification", *Operations Research Letters*, Vol. 45, No. 3, pp. 210–216, 2017.

87. Starita, S. and M. Scaparra, "Optimizing Dynamic Investment Decisions for Railway Systems Protection", *European Journal of Operational Research*, Vol. 248, No. 2, pp. 543–557, 2016.

88. Malaviya, A., C. Rainwater and T. Sharkey, "Multi-Period Network Interdiction Problems with Applications to City-Level Drug Enforcement", *IIE Transactions*, Vol. 44, No. 5, pp. 368–380, 2012.

89. Bengio, Y., A. Lodi and A. Prouvost, "Machine Learning for Combinatorial Optimization: A Methodological Tour D'horizon", *European Journal of Operational Research*, pp. 405–421, 2020.

90. Jin, Y. and B. Sendhoff, "Reducing Fitness Evaluations Using Clustering Techniques and Neural Network Ensembles", *Genetic and Evolutionary Computation Conference*, pp. 688–699, Springer, Seattle, 2004.

91. Karimi-Mamaghan, M., M. Mohammadi, A. Pirayesh, A. M. Karimi-Mamaghan and H. Irani, "Hub-and-Spoke Network Design Under Congestion: A Learning Based Metaheuristic", *Transportation Research Part E: Logistics and Transportation Review*, Vol. 142, p. 102069, 2020.

92. Loshchilov, I., M. Schoenauer and M. Sebag, "A Mono Surrogate for Multiobjective Optimization", *Proceedings of the 12th Annual Conference on Genetic and Evolutionary Computation*, pp. 471–478, Portland, 2010.

93. González-Juarez, D. and E. Andrés-Pérez, "Study of the Influence of the Initial a Priori Training Dataset Size in the Efficiency and Convergence of Surrogate-Based Evolutionary Optimization", *Evolutionary and Deterministic Methods for Design Optimization and Control with Applications to Industrial and Societal Problems*, pp. 181–194, Springer, Cham, 2019.

94. Singh, H. K., T. Ray and W. Smith, "Surrogate Assisted Simulated Annealing (SASA) for Constrained Multi-Objective Optimization", *IEEE Congress on Evolutionary Computation*, pp. 1–8, IEEE, Barcelona, 2010.

95. Louis, S. J. and J. McDonnell, "Learning with Case-Injected Genetic Algorithms",

*IEEE Transactions on Evolutionary Computation*, Vol. 8, No. 4, pp. 316–328, 2004.

96. Li, C., X. Chu, Y. Chen and L. Xing, "A Knowledge-Based Technique for Initializing a Genetic Algorithm", *Journal of Intelligent & Fuzzy Systems*, Vol. 31, No. 2, pp. 1145–1152, 2016.

97. Li, X. and S. Olafsson, "Discovering Dispatching Rules Using Data Mining", *Journal of Scheduling*, Vol. 8, No. 6, pp. 515–527, 2005.

98. Nasiri, M. M., S. Salesi, A. Rahbari, N. Salmanzadeh Meydani and M. Abdollai, "A Data Mining Approach for Population-Based Methods to Solve the JSSP", *Soft Computing*, Vol. 23, No. 21, pp. 11107–11122, 2019.

99. Gelareh, S. and S. Nickel, "Hub Location Problems in Transportation Networks", *Transportation Research Part E: Logistics and Transportation Review*, Vol. 47, No. 6, pp. 1092–1111, 2011.

100. Fan, T. P. C., "Strategic Response from Singapore Airlines to the Rapid Expansion of Global, Full-Service Hub Carriers in the Middle East", *Airline Economics in Asia*, pp. 33–60, Emerald Publishing Limited, Bingley, 2018.

101. Števárová, L. and B. Badánik, "Performance of Hub and Spoke Networks of Selected Airlines", *Transportation Research Procedia*, Vol. 35, pp. 240–249, 2018.

102. Bryan, D. L. and M. E. O'Kelly, "Hub-and-Spoke Networks in Air Transportation: An Analytical Review", *Journal of Regional Science*, Vol. 39, No. 2, pp. 275–295, 1999.

103. Mintzberg, H., S. Ghoshal, J. Lampel and J. B. Quinn, *The Strategy Process: Concepts, Contexts, Cases*, Pearson Education, Essex, 2003.

104. Porter, M. E., "What is Strategy?", *Harvard Business Review*, Vol. 74, pp. 61–78,

1996.

105. Boguslaski, C., H. Ito and D. Lee, "Entry Patterns in the Southwest Airlines Route System", *Review of Industrial Organization*, Vol. 25, No. 3, pp. 317–350, 2004.

106. Burchett, D. L., *Multi-Commodity Fixed-Charge Capacitated Network Design: Polyhedral Characteristics, Network Resilience, and Algorithms*, Ph.D. Thesis, University of Florida, 2015.

107. Kim, D. and C. Barnhart, "Transportation Service Network Design: Models and Algorithms", *Computer-Aided Transit Scheduling*, pp. 259–283, Springer, Berlin, 1999.

108. Hewitt, M., G. L. Nemhauser and M. W. Savelsbergh, "Combining Exact and Heuristic Approaches for the Capacitated Fixed-Charge Network Flow Problem", *INFORMS Journal on Computing*, Vol. 22, No. 2, pp. 314–325, 2010.

109. Costa, A. M., "A Survey on Benders Decomposition Applied to Fixed-Charge Network Design Problems", *Computers & Operations Research*, Vol. 32, No. 6, pp. 1429–1450, 2005.

110. Dempe, S. and A. B. Zemkoho, "The Bilevel Programming Problem: Reformulations, Constraint Qualifications and Optimality Conditions", *Mathematical Programming*, Vol. 138, No. 1, pp. 447–473, 2013.

111. DeNegre, S. T. and T. K. Ralphs, "A Branch-and-Cut Algorithm for Integer Bilevel Linear Programs", *Operations Research and Cyber-Infrastructure*, pp. 65–78, Springer, New York, 2009.

112. Saharidis, G. K. and M. G. Ierapetritou, "Resolution Method for Mixed Integer Bi-Level Linear Problems Based on Decomposition Technique", *Journal of Global Optimization*, Vol. 44, No. 1, pp. 29–51, 2009.

113. Lozano, L. and J. C. Smith, "A Value-Function-Based Exact Approach for the Bilevel Mixed-Integer Programming Problem", *Operations Research*, Vol. 65, No. 3, pp. 768–786, 2017.

114. Glover, F., "Future Paths for Integer Programming and Links to Artificial Intelligence", *Computers & Operations Research*, Vol. 13, No. 5, pp. 533–549, 1986.

115. Boyce, D., A. Farhi and R. Weischedel, "Optimal Network Problem: A Branch-and-Bound Algorithm", *Environment and Planning A*, Vol. 5, No. 4, pp. 519–533, 1973.

116. Şuvak, Z., İ. K. Altınel and N. Aras, "Exact Solution Algorithms for the Maximum Flow Problem with Additional Conflict Constraints", *European Journal of Operational Research*, Vol. 287, No. 2, pp. 410–437, 2020.

117. Gendron, B., M. G. Scutellà, R. G. Garroppo, G. Nencioni and L. Tavanti, "A Branch-and-Benders-Cut Method for Nonlinear Power Design in Green Wireless Local Area Networks", *European Journal of Operational Research*, Vol. 255, No. 1, pp. 151–162, 2016.

118. Taşkın, Z. C. and M. Cevik, "Combinatorial Benders Cuts for Decomposing IMRT Fluence Maps Using Rectangular Apertures", *Computers & Operations Research*, Vol. 40, No. 9, pp. 2178–2186, 2013.

119. Saharidis, G. K., M. Minoux and M. G. Ierapetritou, "Accelerating Benders Method Using Covering Cut Bundle Generation", *International Transactions in Operational Research*, Vol. 17, No. 2, pp. 221–237, 2010.

120. Chouman, M., T. G. Crainic and B. Gendron, "Commodity Representations and Cut-Set-Based Inequalities for Multicommodity Capacitated Fixed-Charge Network Design", *Transportation Science*, Vol. 51, No. 2, pp. 650–667, 2017.

121. Baycik, N. O., "Machine Learning Based Approaches to Solve the Maximum Flow

Network Interdiction Problem", *Computers & Industrial Engineering*, p. 107873, 2021.

122. Friedman, J., T. Hastie, R. Tibshirani *et al.*, *The Elements of Statistical Learning*, Springer Series in Statistics, New York, 2009.

123. Gendreau, M., "An Introduction to Tabu Search", *Handbook of Metaheuristics*, pp. 37–54, Springer, 2003.

124. United Nations High Commissioner for Refugees, *Global Trends: Forced Displacement in 2021*, `https://www.unhcr.org/globaltrends.html`, accessed on July 29, 2022.

125. Turkish Red Crescent, *Göç Direktörlüğü Aylık Faaliyet Raporu*, `https://www.kizilay.org.tr/Upload/Dokuman/Dosya/06-haziran-2022-goc-direktorlugu-aylik-rapor-09-08-2022-50228995.pdf`, accessed on July 29, 2022.

126. Cilali, B., K. Barker and A. D. González, "A Location Optimization Approach to Refugee Resettlement Decision-Making", *Sustainable Cities and Society*, Vol. 74, p. 103153, 2021.

127. Kumcu, G. Ç., *Location-Location Routing Problem and Its Application on Refugee Camps*, Ph.D. Thesis, Bilkent Universitesi (Turkey), 2019.

128. United Nations High Commissioner for Refugees, *Syrian Refugee Camps and Provincial Breakdown of Syrian Refugees Registered in South East Turkey - April 2022*, `https://data.unhcr.org/en/documents/details/92045`, accessed on July 29, 2022.

129. Ilcan, S. and K. Rygiel, ""Resiliency Humanitarianism": Responsibilizing Refugees through Humanitarian Emergency Governance in the Camp", *International Political Sociology*, Vol. 9, No. 4, pp. 333–351, 2015.

130. Jahre, M., J. Kembro, A. Adjahossou and N. Altay, "Approaches to the Design of Refugee Camps: An Empirical Study in Kenya, Ethiopia, Greece, and Turkey", *Journal of Humanitarian Logistics and Supply Chain Management*, pp. 323–345, 2018.

131. Karsu, O., B. Y. Kara and B. Selvi, "The Refugee Camp Management: A General Framework and a Unifying Decision-Making Model", *Journal of Humanitarian Logistics and Supply Chain Management*, Vol. 9, No. 2, pp. 131–150, 2019.

132. Li, J., P. M. Pardalos, H. Sun, J. Pei and Y. Zhang, "Iterated Local Search Embedded Adaptive Neighborhood Selection Approach for the Multi-Depot Vehicle Routing Problem with Simultaneous Deliveries and Pickups", *Expert Systems with Applications*, Vol. 42, No. 7, pp. 3551–3561, 2015.

133. Wang, J., Y. Sun, Z. Zhang and S. Gao, "Solving Multitrip Pickup and Delivery Problem with Time Windows and Manpower Planning Using Multiobjective Algorithms", *IEEE/CAA Journal of Automatica Sinica*, Vol. 7, No. 4, pp. 1134–1153, 2020.

134. Benlic, U., M. G. Epitropakis and E. K. Burke, "A Hybrid Breakout Local Search and Reinforcement Learning Approach to the Vertex Separator Problem", *European Journal of Operational Research*, Vol. 261, No. 3, pp. 803–818, 2017.

135. Thevenin, S. and N. Zufferey, "Learning Variable Neighborhood Search for a Scheduling Problem with Time windows and Rejections", *Discrete Applied Mathematics*, Vol. 261, pp. 344–353, 2019.

136. Arnold, F., Í. Santana, K. Sörensen and T. Vidal, "PILS: Exploring High-Order Neighborhoods by Pattern Mining and Injection", *Pattern Recognition*, Vol. 116, p. 107957, 2021.

137. Holland, J. H., "Genetic Algorithms", *Scientific American*, Vol. 267, No. 1, pp.

66–73, 1992.

138. Liu, C.-M., "Entry Behaviour and Financial Distress: An Empirical Analysis of the US Domestic Airline Industry", *Journal of Transport Economics and Policy (JTEP)*, Vol. 43, No. 2, pp. 237–256, 2009.

139. Pedregosa, F., G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot and E. Duchesnay, "Scikit-learn: Machine Learning in Python", *Journal of Machine Learning Research*, Vol. 12, pp. 2825–2830, 2011.

140. Probst, P., M. N. Wright and A.-L. Boulesteix, "Hyperparameters and Tuning Strategies for Random Forest", *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, Vol. 9, No. 3, pp. 1301–1315, 2019.

# APPENDIX A: Random Instances for the BOpt-RRC

Table A.1. BOpt-RRC - Instances with $\omega_{new} = 0.02$.

| Instance | rho | rho% | No. | #Existing Ref. | #New Ref. | #ExistingCamps | TotalCap |
|---|---|---|---|---|---|---|---|
| 1 | 50 | 2 | 1 | 46298 | 916 | 4 | 50000 |
| 2 | 50 | 2 | 2 | 47882 | 1038 | 4 | 50000 |
| 3 | 50 | 2 | 3 | 46886 | 1001 | 4 | 50000 |
| 4 | 50 | 2 | 4 | 48482 | 932 | 6 | 50000 |
| 5 | 50 | 2 | 5 | 49832 | 955 | 6 | 50000 |
| 21 | 100 | 2 | 1 | 107466 | 2074 | 6 | 110000 |
| 22 | 100 | 2 | 2 | 91766 | 1863 | 7 | 95000 |
| 23 | 100 | 2 | 3 | 103266 | 2239 | 7 | 105000 |
| 24 | 100 | 2 | 4 | 107298 | 2240 | 8 | 110000 |
| 25 | 100 | 2 | 5 | 89699 | 1828 | 8 | 90000 |
| 41 | 150 | 2 | 1 | 142200 | 3177 | 13 | 145000 |
| 42 | 150 | 2 | 2 | 159800 | 2824 | 12 | 160000 |
| 43 | 150 | 2 | 3 | 153150 | 2855 | 15 | 155000 |
| 44 | 150 | 2 | 4 | 132250 | 3435 | 15 | 135000 |
| 45 | 150 | 2 | 5 | 147550 | 2768 | 10 | 150000 |
| 61 | 200 | 2 | 1 | 221733 | 3453 | 19 | 225000 |
| 62 | 200 | 2 | 2 | 194865 | 4164 | 15 | 195000 |
| 63 | 200 | 2 | 3 | 220732 | 3402 | 16 | 225000 |
| 64 | 200 | 2 | 4 | 197599 | 3616 | 14 | 200000 |
| 65 | 200 | 2 | 5 | 182866 | 3558 | 13 | 185000 |

Table A.2. BOpt-RRC - Instances with $\omega_{new} = 0.06$.

| Instance | rho | rho% | No. | #Existing Ref. | #New Ref. | #ExistingCamps | TotalCap |
|---|---|---|---|---|---|---|---|
| 6 | 50 | 6 | 1 | 51666 | 2809 | 7 | 55000 |
| 7 | 50 | 6 | 2 | 50249 | 2996 | 3 | 55000 |
| 8 | 50 | 6 | 3 | 54049 | 3384 | 5 | 55000 |
| 9 | 50 | 6 | 4 | 47899 | 2835 | 4 | 50000 |
| 10 | 50 | 6 | 5 | 57232 | 2933 | 4 | 60000 |
| 26 | 100 | 6 | 1 | 106833 | 5174 | 8 | 110000 |
| 27 | 100 | 6 | 2 | 106865 | 6868 | 10 | 110000 |
| 28 | 100 | 6 | 3 | 98066 | 5306 | 8 | 100000 |
| 29 | 100 | 6 | 4 | 111733 | 6086 | 11 | 115000 |
| 30 | 100 | 6 | 5 | 108166 | 5664 | 10 | 110000 |
| 46 | 150 | 6 | 1 | 137100 | 7323 | 8 | 140000 |
| 47 | 150 | 6 | 2 | 143700 | 9456 | 11 | 145000 |
| 48 | 150 | 6 | 3 | 150000 | 8757 | 12 | 150000 |
| 49 | 150 | 6 | 4 | 141700 | 7935 | 13 | 141700 |
| 50 | 150 | 6 | 5 | 146750 | 8883 | 12 | 146750 |
| 66 | 200 | 6 | 1 | 182732 | 11480 | 15 | 185000 |
| 67 | 200 | 6 | 2 | 188066 | 12280 | 14 | 190000 |
| 68 | 200 | 6 | 3 | 216599 | 12212 | 18 | 220000 |
| 69 | 200 | 6 | 4 | 193466 | 12948 | 15 | 195000 |
| 70 | 200 | 6 | 5 | 192199 | 13112 | 17 | 195000 |

Table A.3. BOpt-RRC - Instances with $\omega_{new} = 0.10$.

| **Instance** | **rho** | **rho%** | **No.** | **#Existing Ref.** | **#New Ref.** | **#ExistingCamps** | **TotalCap** |
|---|---|---|---|---|---|---|---|
| 11 | 50 | 10 | 1 | 47232 | 5482 | 4 | 50000 |
| 12 | 50 | 10 | 2 | 47633 | 4494 | 8 | 50000 |
| 13 | 50 | 10 | 3 | 52299 | 5141 | 7 | 55000 |
| 14 | 50 | 10 | 4 | 53549 | 5717 | 5 | 55000 |
| 15 | 50 | 10 | 5 | 48999 | 4636 | 6 | 50000 |
| 31 | 100 | 10 | 1 | 97799 | 9965 | 8 | 100000 |
| 32 | 100 | 10 | 2 | 103999 | 8746 | 9 | 105000 |
| 33 | 100 | 10 | 3 | 91566 | 9289 | 9 | 95000 |
| 34 | 100 | 10 | 4 | 95233 | 10442 | 10 | 100000 |
| 35 | 100 | 10 | 5 | 104098 | 11129 | 7 | 105000 |
| 51 | 150 | 10 | 1 | 157750 | 14495 | 11 | 160000 |
| 52 | 150 | 10 | 2 | 148750 | 14170 | 10 | 150000 |
| 53 | 150 | 10 | 3 | 156300 | 15350 | 14 | 160000 |
| 54 | 150 | 10 | 4 | 158800 | 14780 | 14 | 160000 |
| 55 | 150 | 10 | 5 | 153050 | 14895 | 15 | 155000 |
| 71 | 200 | 10 | 1 | 203865 | 18758 | 17 | 205000 |
| 72 | 200 | 10 | 2 | 180000 | 22485 | 20 | 180000 |
| 73 | 200 | 10 | 3 | 222465 | 17386 | 17 | 225000 |
| 74 | 200 | 10 | 4 | 154866 | 22499 | 15 | 155000 |
| 75 | 200 | 10 | 5 | 231132 | 18819 | 17 | 235000 |

Table A.4. BOpt-RRC - Instances with $\omega_{new} = 0.20$.

| Instance | rho | rho% | No. | #Existing Ref. | #New Ref. | #ExistingCamps | TotalCap |
|---|---|---|---|---|---|---|---|
| 16 | 50 | 20 | 1 | 59966 | 10676 | 6 | 60000 |
| 17 | 50 | 20 | 2 | 50783 | 10659 | 3 | 55000 |
| 18 | 50 | 20 | 3 | 44633 | 10739 | 3 | 45000 |
| 19 | 50 | 20 | 4 | 50800 | 11422 | 5 | 55000 |
| 20 | 50 | 20 | 5 | 54232 | 10789 | 5 | 55000 |
| 36 | 100 | 20 | 1 | 99432 | 16706 | 8 | 100000 |
| 37 | 100 | 20 | 2 | 99732 | 20886 | 10 | 100000 |
| 38 | 100 | 20 | 3 | 96899 | 20925 | 10 | 100000 |
| 39 | 100 | 20 | 4 | 109866 | 20759 | 10 | 110000 |
| 40 | 100 | 20 | 5 | 101432 | 18233 | 9 | 105000 |
| 56 | 150 | 20 | 1 | 125600 | 28890 | 10 | 130000 |
| 57 | 150 | 20 | 2 | 132750 | 31210 | 11 | 135000 |
| 58 | 150 | 20 | 3 | 137550 | 33420 | 10 | 140000 |
| 59 | 150 | 20 | 4 | 133100 | 30300 | 14 | 135000 |
| 60 | 150 | 20 | 5 | 141900 | 31610 | 9 | 145000 |
| 76 | 200 | 20 | 1 | 173066 | 39333 | 15 | 175000 |
| 77 | 200 | 20 | 2 | 221465 | 39732 | 15 | 225000 |
| 78 | 200 | 20 | 3 | 192532 | 37052 | 17 | 195000 |
| 79 | 200 | 20 | 4 | 197332 | 42292 | 16 | 200000 |
| 80 | 200 | 20 | 5 | 183799 | 45666 | 13 | 185000 |