Delft University of Technology

Delft Center for Systems and Control

Technical report 17-013

A Node Current-based 2-Index Formulation for the Fixed-Destination Multi-Depot Travelling Salesman Problem*

M. Burger, Z. Su, and B. De Schutter

To cite this work, please refer to the published version:

M. Burger, Z. Su, and B. De Schutter, "A node current-based 2-index formulation for the fixed-destination multi-depot travelling salesman problem," *European Journal of Operational Research*, vol. 265, no. 2, pp. 463–477, Mar. 2018. doi:10.1016/j.ejor.2017.07.056

Delft Center for Systems and Control
Delft University of Technology
Mekelweg 2, 2628 CD Delft
The Netherlands
phone: +31-15-278.24.73 (secretary)
URL: https://www.dcsc.tudelft.nl

* This report can also be downloaded via https://dspub.eu/17-013

A Node Current-based 2-Index Formulation for the Fixed-Destination Multi-Depot Travelling Salesman Problem

M. Burger[a,*], Z. Su[b], B. De Schutter[b]

[a] *TBA, Karrepad 2a, 2623 AP Delft, The Netherlands*
[b] *Delft Center for Systems and Control, Delft University of Technology, Mekelweg 2, 2628 CD Delft, The Netherlands*

Abstract

The travelling salesman problem forms a basis for many optimisation problems in logistics, finance, and engineering. Several variants exist to accommodate for different problem types. In this article we discuss the fixed-destination, multi-depot travelling salesman problem, where several salesmen will start from different depots, and they are required to return to the depot they originated from. We propose a novel formulation for this problem using 2-index binary variables and node currents, and compare it to other 2-index formulations from the literature. This novel formulation requires less binary variables and continuous variables to formulate a problem, resulting in lower computation times. Using a large benchmark the effectiveness of the new formulation is demonstrated.

Keywords: travelling salesman, node current, integer programming, fixed-destination problem

1. INTRODUCTION

Although the travelling salesman problem (TSP) can be stated simply as *"Find the shortest route that connects all cities on a map"*, solving this problem has kept people busy for decades. The ongoing quest for faster algorithms for finding (an approximation of) the optimal solution of the TSP has led to a large amount of literature on the subject. Heuristic methods [22, 29] can be used to find solutions of large TSP instances quickly, but no guarantees can be given for finding the optimal solution. In this article we consider exact formulations that guarantee finding the globally optimal solution. A comprehensive discussion of the history and state-of-the-art for solving the TSP can be found in [1, 14].

1.1. Literature Review

The power of the TSP [15, 30, 33] does not only lie in finding tours of minimal distance along cities, but also in the fact that it forms the mathematical basis of many scheduling and routing problems. Extensions such as the vehicle routing problem [25, 46] and the pick-up and delivery problem [36, 37, 42, 43] are important problems in the fields of logistics and economics. Recent applications include the optimal maintenance routing and scheduling for offshore wind farms [23], and the optimal delivery or pickup of goods using hybrid electric vehicles [17]. In those problems one usually tries to minimise some 'cost' (e.g. distance, time, money, or a combination) using multiple 'salesmen' (e.g. people, trucks, air planes, vessels) that can visit the 'cities' (e.g. shops, harbours, airports, or actual cities). The use of multiple salesmen to visit the cities makes the problems harder to solve due to the increase in possible solutions. The multiple travelling salesmen problem (mTSP) is at the basis of the vehicle routing problem and the pick-up and delivery problem.

The essence of mTSP is to find the shortest total travel distance for multiple salesmen starting from and returning to a single depot/home city. Since certain problems require more than one depot (e.g. for delivering goods to shops that can be supplied from multiple storage facilities), an extension to the multi-depot multiple-salesmen TSP (MmTSP) has been made [3]. In this case the problem consists of finding the shortest distance such that several salesmen will start at a depot, they visit all the cities once (and only once), and return to a depot again. When it is not important at what depot the salesmen end their route, we talk about a **nonfixed-destination prob-**

*Telephone: +3115 380 5775, Fax: +3115 380 5763
Email addresses: mernout.burger@tba.group (M. Burger), z.su-1@tudelft.nl (Z. Su), b.deschutter@tudelft.nl (B. De Schutter)

[1]This work is supported by the European Union Seventh Framework Programme [FP7/2007-2013] under grant agreement no. 257462 HYCON2 Network of Excellence.

lem; when the salesmen are supposed to return to their original depot we talk about a **fixed-destination problem** [24]. The work of [6] is also concerned with multi-depot TSPs, where the number of salesmen per depot is not limited and the travel distances are symmetric. In this article we will focus on problems with a fixed number of salesmen per depot, and asymmetric costs.

The fixed-destination Multi-depot multiple-salesmen TSP (FMmTSP) is a restricted case of the nonfixed-destination problem, with the additional constraint that all salesmen should return to their original location. Therefore, the former is more difficult to solve than the latter; the solutions to the FMmTSP are a subset of the solutions to the MmTSP. In [24] a mixed-integer linear programming (MILP) description for the fixed-destination problem has been proposed using 3-index decision variables, resulting in a large increase in binary variables for each added depot.

Cycle (or subtour) elimination constraints (CECs) are used to ensure that no cycles exist within the set of city nodes. They have been a topic of active research over many decennia, starting with the use of loop constraints by Dantzig et al. [15] in 1954, the node potentials by Miller et al. [30] in 1960, and (multi)-commodity flow-based constraints in [20] starting from 1978. Loop conditions give strong linear programming relaxations, but the number of constraints grows exponentially with the problem size. The number of node-potential-based constraints only grows quadratically with the problem size, but they result in much weaker relaxations. Using multi-commodity flow formulations it is possible to obtain strong relaxations, but with a number of constraints growing cubically in the problem size.

Cycle imposement constraints (CICs) can be used to ensure a (minimum) number of cycles in a set of nodes. Fixed-destination solutions for TSP-like problems can be created by enforcing that there should be at least D cycles in the combined set of depot and city nodes, while using CECs to ensure that no subtours exist in the set of city nodes; when D equals the number of depots this will result in exactly D cycles in the network; one for each of the depots. CICs have only recently been discussed in the literature, starting with the path elimination constraints of Belenguer et al. [5] in 2011, the multi-commodity flow-based constraints of Bektaş [4] in 2012, and the node currents of [8] in 2014.

Table 1 shows the order of the number of the discussed CECs and CICs. This article will introduce the node current-based CICs, which can be seen as the equivalent of the node potentials for cycle imposement.

Table 1: **Overview of CECs and CICs and the order of their numbers**

Order	Cycle elimination	Cycle imposement
$O(2^N)$	loop conditions [15]	path elimination [5]
$O(N^3)$	commodity flow [20]	commodity flow [4]
$O(N^2)$	node potentials [30]	**node currents** [8]

1.2. Contributions

The main contribution of this article is the generalisation of the node current constraints from [8] for the one-salesman-per-depot case to the multiple-salesmen-per-depot case, resulting in a novel formulation for FMmTSPs with asymmetric costs. Furthermore, assignment constraints are presented that limit the number of salesmen per depot for the MmTSP.

Node currents are used for *cycle imposement constraints*, which ensure each salesman to return to his original depot. They can be seen as the dual to the node potentials introduced by Miller et al. [30] in their subtour elimination constraints. We have used a preliminary version of these cycle imposement constraints for micro-ferry scheduling problems with the purpose of identifying which micro-ferry will pick up which customer [10], and for the routing of multiple harvesters [9]. In the current article this method is used to enforce fixed-destination solutions to the MmTSP.

For an FMmTSP with L nodes/locations, including D depots and C cities, two approaches where $\mathsf{L} = 2\mathsf{D} + \mathsf{C}$ exist in the literature [4, 32]; we will introduce an approach that uses $\mathsf{L} = \mathsf{D} + \mathsf{C}$ nodes, thereby reducing the amount of costly binary variables[2]. This formulation has been presented for the fixed-destination, multi-depot, single-salesman-per-depot TSP in [8]; here we introduce the generalisation for problems with multiple salesmen per depot.

1.3. Outline

Section 2 provides an introduction to the FMmTSP, and defines the problem discussed in this article. An FMmTSP formulation consists of four components, which are discussed in detail in Section 3. The main contribution of this article is the introduction of node currents as a means to enforce fixed-destination solutions through cycle imposement constraints. This approach is discussed in Section 4, and it is compared to two existing approaches. A computational comparison for three distinct FMmTSP formulations will be given in Section 5, followed by concluding remarks in Section 6.

[2] In general, using more binary variables will lead to larger computation times for solving the problem.

2. PROBLEM DESCRIPTION

For the mTSP with one depot (and multiple salesmen) it is obvious that all salesmen should return to this single depot. However, when considering multiple depots, two situations can occur: either the salesmen may end their tour at any depot, or they are required to return to their original depot. The latter is a restricted case of the former and can be obtained by using additional constraints, as will be discussed next.

2.1. Description of the MmTSP

We consider the MmTSP, where there are D depots available from where C cities should be visited. Each depot d has a certain number of salesmen available, indicated by m_d. Each city should be visited by *one and only one* salesman. The cost to travel from city i to j is denoted by the constant c_{ij}. The costs can be asymmetric, i.e., c_{ij} and c_{ji} might be different. The decision variable x_{ij} indicates whether ($x_{ij} = 1$) or not ($x_{ij} = 0$) city j is visited *directly after* city i by a salesman.

The locations of both the cities and the salesmen can be taken into account in the modelling by denoting both the D locations of the depots (where the salesmen are) and the C locations of the cities as one set of $\mathsf{L} = \mathsf{C} + \mathsf{D}$ locations. The sets $\mathcal{D}$, C, and $\mathcal{L}$—associated with the depots, the cities, and the locations respectively— are defined as

$$\mathcal{D} = \{1, \ldots, \mathsf{D}\},\ C = \{\mathsf{D}+1, \ldots, \mathsf{L}\},\ \mathcal{L} = \mathcal{D} \cup C. \quad (1)$$

2.2. Description of the FMmTSP

For the fixed-destination problems, the additional restriction that all salesmen should return to their original depots makes the FMmTSP more difficult to solve. The reason is that compared with the non-fixed destination MmTSP, new auxiliary variables or decision variables of higher index are required for the fixed-destination setting to impose the additional restriction that each salesman must return to his departing depot. In the literature often the (fixed-destination) multi-depot problems (or multi-vehicle problems when considering vehicle routing) are solved using a 3-index variant of the decision variables [3, 12, 39, 46], thereby drastically increasing the number of binary variables (that are computationally expensive) with each depot added to the problem. The three indices represent 1) the origin, 2) the destination, and 3) the depot (vehicle) number. An example of an MILP description of both nonfixed- and fixed-destination MmTSP formulations using 3-index formulations is given in [24].

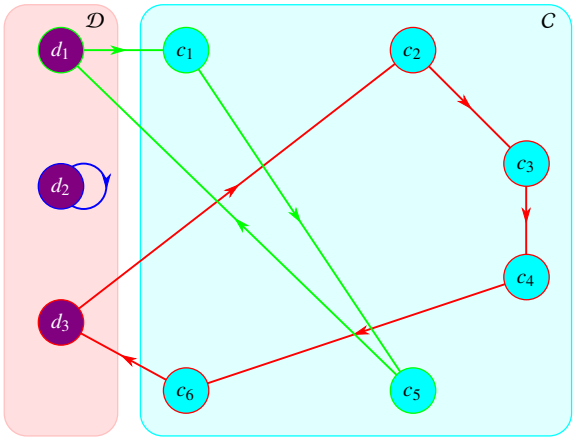

Figure 1: *A solution to the FmMTSP. Cycles should be eliminated in the set C, whereas they should be imposed in the set $\mathcal{L} = \mathcal{D} \cup C$.*

Recently, formulations of the MmTSP [32] and FMmTSP [4] using 2-index binary variables have been presented in the literature. Both methods make use of a copy $\mathcal{D}'$ of the set of depot nodes $\mathcal{D}$, where one depot serves as a start point and the other depot serves as the end point of a tour for a salesman. For D depot nodes and C city nodes in the original problem, there will be $2\mathsf{D}$ depot nodes and C city nodes in the extended problem. The resulting set of nodes in the graph is

$$\mathcal{L}' = \mathcal{D} \cup C \cup \mathcal{D}', \quad (2)$$

where the copied set of the D depots in $\mathcal{D}$ is defined as

$$\mathcal{D}' = \{1', \ldots, \mathsf{D}'\} = \{\mathsf{L}+1, \ldots, \mathsf{L}+\mathsf{D}\}, \quad (3)$$

such that node i and node $i' = \mathsf{L}+i$ represent the start and end depot of depot $i \in \mathcal{D}$, respectively. This formulation results in $\mathsf{L}' = 2\mathsf{D} + \mathsf{C}$ nodes in the graph[3].

A transformation of the MmTSP problem to an asymmetric TSP has been proposed by Oberlin et al. [31, 32] by using a copy of the depot nodes as in (3). More recently Bektaş [4] has proposed a method to solve the FMmTSP using $\mathcal{D}'$ based on commodity flows. In Section 4 we will introduce a formulation that only requires $\mathsf{L} = \mathsf{D} + \mathsf{C}$ nodes to represent the same problem.

[3] With an efficient implementation –where the start node only has outgoing arcs and the end node only has incoming arcs– the number of arcs (and thereby the number of binary variables) remains the same.

3. PROBLEM FORMULATION

The FMmTSP can be described as a mathematical program consisting of the following components[4]

$$\begin{aligned} &\text{minimise} && \textit{costs} && \text{(4a)}\\ &\text{subject to} && \textit{assignment constraints} && \text{(4b)}\\ &&& \textit{cycle elimination constraints} && \text{(4c)}\\ &&& \textit{cycle imposement constraints} && \text{(4d)} \end{aligned}$$

In this section we provide a brief overview of the currently available types of constraints for each of the components. For a more thorough discussion on the available cycle (subtour) elimination constraints (including the relations of their linear relaxation strengths) we refer to [41] and the references therein.

For ease of notation we will use the index d' and the sets $\mathcal{D}'$ and $\mathcal{L}'$ in this section, where their definition will depend on the type of formulation that is used, as specified in Table 2.

Table 2: **The definition of prime symbols for formulations with and without depot-node copies.**

	with copies of depots	without copies of depots
d'	$d + \mathsf{L}$	d
$\mathcal{D}'$	$\{\mathsf{L}+1, \dots, \mathsf{L}+\mathsf{D}\}$	$\mathcal{D}$
$\mathcal{L}'$	$\{1, \dots, \mathsf{L}+\mathsf{D}\}$	$\mathcal{L}$

3.1. Costs

The cost in (FMm)TSPs is the distance the salesmen travel, resulting in minimising the *total travel distance*

$$J_{\text{td}} = \sum_{i \in \mathcal{L}'} \sum_{j \in \mathcal{L}'} \mathsf{c}_{ij} x_{ij}, \tag{5}$$

where $\mathsf{c}_{ij} \geq 0$ is the travel distance between cities i and j, and $x_{ij} \in \{0, 1\}$ is a decision variable satisfying

$$x_{ij} = \begin{cases} 1 & \text{if city } j \text{ is visited directly after city } i, \\ 0 & \text{otherwise.} \end{cases} \tag{6}$$

When using 3-index decision variables x_{ijk} the total travel distance is given by

$$J_{\text{td}} = \sum_{i \in \mathcal{L}} \sum_{j \in \mathcal{L}} \sum_{k \in \mathcal{D}} \mathsf{c}_{ij} x_{ijk}, \tag{7}$$

where $\mathsf{c}_{ij} \geq 0$ is the travel distance between cities i and j, and $x_{ijk} \in \{0, 1\}$ is a decision variable satisfying

$$x_{ijk} = \begin{cases} 1 & \text{if city } j \text{ is visited directly after city } i \text{ by a} \\ & \text{salesman originating from depot } k, \\ 0 & \text{otherwise.} \end{cases} \tag{8}$$

3.2. Assignment Constraints

The assignment constraints ensure that each node has exactly one incoming arc and one outgoing arc (see Figure 1), thereby satisfying a necessary condition for visiting the cities *once and only once*.

3.2.1. Description of the assignment constraints

The assignment constraints for the (F)MmTSP [3, 27] are given by

$$\sum_{j \in \mathcal{L}'} x_{dj} = m_d \;\forall\, d \in \mathcal{D} \tag{9a}$$

$$\sum_{j \in \mathcal{L}'} x_{ij} = 1 \quad \forall\, i \in C \tag{9b}$$

$$\sum_{i \in \mathcal{L}'} x_{ij} = 1 \quad \forall\, j \in C \tag{9c}$$

$$\sum_{i \in \mathcal{L}'} x_{id'} = m_d \;\forall\, d' \in \mathcal{D}' \tag{9d}$$

$$x_{ij} \in \{0, 1\} \quad \forall\, i, j \in \mathcal{L}' \tag{9e}$$

Due to (9a) all of the m_d salesmen will leave their depot d, and by (9b) each city i is succeeded by exactly one location (a salesman leaves the city). Furthermore, equations (9c) ensure that each city j is preceded by exactly one location (a salesman enters the city), whereas (9d) ensures that m_d salesmen will return to depot d'. The set $\mathcal{D}'$ denotes —depending on the problem formulation— either a copy of the depot nodes, or the original set $\mathcal{D}$ of depot nodes, as defined in Table 2. Finally, (9e) ensures that the decision variable x_{ij} is treated as a binary variable.

3.2.2. Variants for the assignment constraints

The assignment constraints in (9) force all m_d salesmen to leave their depot, and also require m_d salesmen to return to depot d. The former is restrictive for both fixed- and nonfixed-destination problems, whereas the latter only restricts the solutions for the FMmTSP. Both restrictions can be loosened as shown next.

Idle salesmen: To allow salesmen to stay at the depot without visiting a city (hence some salesmen may be

[4] Constraints (4c) are commonly known as subtour elimination constraints (SECs). Since the term 'subtour' has the implicit property of being undesired, we will present the counterpart of the SECs as cycle imposement constraints, thereby using the modern term 'cycle' instead of 'subtour'. For consistency we therefore also use the term cycle elimination constraints.

'idle'; they do not provide any work), the equality constraints (9a) and (9d) can be substituted by (see [45])

$$\sum_{j\in\mathcal{L}'} x_{dj} \le m_d \qquad \forall\, d\in\mathcal{D} \tag{9a*}$$

$$\sum_{i\in\mathcal{L}'} x_{id'} = \sum_{j\in\mathcal{L}} x_{dj} \;\forall\, d'\in\mathcal{D}' \tag{9d*}$$

where (9a*) limits the amount of salesmen that can leave depot d to m_d (which equals the number of salesmen present at depot d), whereas (9d*) ensures that the same amount of salesmen that have left the depot, will also return to the depot.

Fixed-capacity depots: For nonfixed-destination problems the number of salesmen at a depot will in general be different before and after the salesmen travelled. To avoid solutions where certain depots will receive more salesmen than they can facilitate, an upper bound on the number of salesmen that are allowed to return to each specific depot should be set. To accomplish this we propose the following.

If the capacity of depot d (with d' the associated end depot) is $q_{d'}$ salesmen one could substitute (9d*) with

$$m_d + \sum_{i\in\mathcal{L}'} x_{id'} \le q_d + \sum_{j\in\mathcal{L}'} x_{dj} \qquad \forall\, d'\in\mathcal{D}' \tag{9d$_1^\star$}$$

$$\sum_{d'\in\mathcal{D}'}\sum_{j\in\mathcal{L}'} x_{d'j} = \sum_{d\in\mathcal{D}}\sum_{i\in\mathcal{L}'} x_{id} \tag{9d$_2^\star$}$$

Inequalities (9d$_1^\star$) ensure that no more than $q_{d'}$ salesmen end up in depot d',[5] whereas (9d$_2^\star$) ensures that all the salesmen that leave a depot will also return to a depot.

3.3. Cycle Elimination Constraints

Note that the constraints (9) do not avoid

i) the existence of cycles (subtours) in C, resulting in routes along cities that do not have a salesman associated with them,
ii) the existence of cycles in $\mathcal{L}'$ containing more than one node from $\mathcal{D}$, resulting in a schedule where salesmen end their tour at an arbitrary depot.

The former situation would result in a schedule where some cities will not be visited by a salesman (since no-one is assigned to do so), whereas the latter situation would result in a schedule where the salesmen do not have the guarantee that they return to their original depot. To assure that *each city is visited by a salesman*, solutions using *cycle elimination constraints* have been proposed in the literature [15, 19, 20, 30]. These constraints are based on different concepts, for which a brief description will be provided next; a more detailed description can be found in [34].

3.3.1. Loop conditions

The loop conditions were introduced in the seminal work of Dantzig et al. [15], which can be stated as

$$\sum_{i,j\in\mathcal{S}} x_{ij} \le |\mathcal{S}| - 1 \quad \forall\, \mathcal{S}\subset\mathcal{L}',\, 2\le|\mathcal{S}|\le \mathsf{L}'-1, \tag{10}$$

where $|\mathcal{S}|$ represents the cardinality of set $\mathcal{S}$ (see [21]). These constraints provide strong linear programming relaxations, but the number of constraints grows exponentially with the number of nodes.

3.3.2. Node potentials

Miller et al. [30] proposed an approach for eliminating cycles by using additional variables u_i that represent node potentials. Using the strengthened formulation of Desrochers and Laporte [16], the C continuous variables u_i should satisfy

$$u_i - u_j + \mathsf{C}x_{ij} + (\mathsf{C}-2)x_{ji} \le \mathsf{C}-1 \quad \forall\, i,j\in C, \tag{11}$$

resulting in C^2 inequality constraints.

The node potential representation has been extended by Kara and Bektaş [4, 24] to set *workload bounds*[6] on the number of cities a salesman should visit. Denoting $\underline{\mathsf{u}}$ and $\overline{\mathsf{u}}$ as the minimum and maximum number of cities the salesmen may visit respectively, the cycle imposement constraints

$$u_i - u_j + \overline{\mathsf{u}}x_{ij} + (\overline{\mathsf{u}}-2)x_{ij} \le \overline{\mathsf{u}}-1 \qquad \forall\, i,j\in C \tag{12a}$$

$$u_i + (\overline{\mathsf{u}}-2)\sum_{d\in\mathcal{D}} x_{di} - \sum_{d'\in\mathcal{D}'} x_{id'} \le \overline{\mathsf{u}}-1 \qquad \forall\, i\in C \tag{12b}$$

$$u_i + \sum_{d\in\mathcal{D}} x_{di} + (2-\underline{\mathsf{u}})\sum_{d'\in\mathcal{D}'} x_{id'} \ge 2 \qquad \forall\, i\in C \tag{12c}$$

ensure that each salesman will be assigned between $\underline{\mathsf{u}}$ and $\overline{\mathsf{u}}$ cities to visit, and city i will be the u_i-th city a salesman visits[7]. Inequalities (12a) provide cycle elimination constraints. In both (12b) and (12c) the first summation is 1 if and only if node i represents the first city

[5] The number of salesman returning to depot d' is less or equal to the capacity q_d minus the number of salesmen m_d present at the start plus the number of salesmen that leave depot d.

[6] Loop conditions representation with workload bound were first proposed in [24] for the mTSP (single depot), and were extended in [4] for the MmTSP (multidepot)

[7] In [4, 24] it is stated that these inequality constraints are only valid for $\underline{\mathsf{u}} \ge 4$; this is only under the restriction that salesmen should at least visit two cities, and hence $x_{di} = x_{id'} = 1$ is not allowed. Lifting this restriction by allowing salesmen to visit zero or one city these inequality constraints are valid for all $\underline{\mathsf{u}} \ge 0$.

of a tour (and it is 0 otherwise), and the second summation is 1 if and only if node i represents the last city of a tour (and is 0 otherwise). Therefore, (12b) and (12c) ensure that $u_i \leq \overline{\mathsf{u}}$ and $u_i \geq \underline{\mathsf{u}}$ for the last cities in a tour, thereby setting the desired upper and lower bound respectively on the number of visited cities per salesman.

For the TSP with time windows [2] the constraints

$$t_i - t_j + \tau_{ij} + \mathsf{T} x_{ij} \leq \mathsf{T} \tag{13}$$

can be seen as a variant of the node potential approach, where t_i is the time instant city i is visited, τ_{ij} is the potential difference between the nodes, and T is a sufficiently large constant.

3.3.3. Commodity flows

Gavish and Graves [20] introduced commodity flows as a means for eliminating undesired cycles. The L'^2 continuous variables f_{ij} should satisfy the constraints

$$\sum_{j \in \mathcal{L}'} f_{ij} - \sum_{j \in \mathcal{L}'} f_{ji} = 1 \qquad \forall\, i \in C \tag{14a}$$

$$f_{ij} \leq \mathsf{C} x_{ij} \qquad \forall\, i \in C,\, j \in \mathcal{L}' \tag{14b}$$

$$f_{ij} \geq 0 \qquad \forall\, i, j \in \mathcal{L}' \tag{14c}$$

Extensions to two-commodity flows [18] and multi-commodity flows [11, 47] have been proposed, resulting in stronger linear programming relaxations [26, 35].

3.3.4. Time periods

For a graph with L' nodes, Fox et al. [19] present a cycle elimination formulation using a 3-index binary variable representation x_{ijt}, where

$$x_{ijt} = \begin{cases} 1 & \text{if } i \text{ precedes } j \text{ as the } t\text{-th node in the tour,} \\ 0 & \text{otherwise.} \end{cases} \tag{15}$$

The index $t \in \mathcal{T}$ represents the time period in which the salesman travels from city i to city j. To ensure that all cities are visited in some time period, and that in each time period only one city is visited, the $\mathrm{O}(\mathsf{L}'^3)$ binary variables x_{ijt} should satisfy

$$\sum_{i \in \mathcal{L}'} \sum_{j \in \mathcal{L}'} \sum_{t \in \mathcal{T}} x_{ijt} = \mathsf{L}' \tag{16a}$$

$$\sum_{j \in \mathcal{L}'} \sum_{t \in \mathcal{T} \setminus \{1\}} t x_{ijt} - \sum_{j \in \mathcal{L}'} \sum_{t \in \mathcal{T}} t x_{jit} = 1 \qquad \forall\, i \in \mathcal{L}'_{-} \tag{16b}$$

$$x_{ijt} \in \{0, 1\} \qquad \forall\, i, j \in \mathcal{L}' \quad t \in \mathcal{T} \tag{16c}$$

Constraints (16a) and (16c) replace the assignment constraints (9). Equality constraints (16b) ensure that in each time period t exactly one city i is visited.

3.4. Cycle Imposement Constraints

To assure that *each salesman returns to the original depot*, additional constraints are needed to enforce cycles that start and end in the same depot (or paths leading from one start depot towards the associated end depot). Opposite to the cycle elimination constraints, the constraints that enforce the existence of a certain amount of cycles in a graph can be seen as *cycle imposement constraints*. To the authors' best knowledge, currently only three approaches exist for obtaining fixed-destination solutions; using 3-index binary variables, or using 2-index binary variables plus commodity flow variables [4, 24], or using the path elimination constraints [5], which introduces no new variables, but modifies the definition of the 2-index binary variable associated with each arc. We will introduce a fourth approach in Section 4 that is based on node currents, which can be seen as the dual of the node potentials introduced by Miller et al. [30] presented in (11).

3.4.1. 3-Index formulation

Using decision variables x_{ijd} that satisfy

$$x_{ijd} = \begin{cases} 1 & \text{if } i \text{ precedes } j \text{ directly in the tour of depot } d, \\ 0 & \text{otherwise,} \end{cases} \tag{17}$$

the existence of D cycles can be enforced [24] using

$$\sum_{j \in C} x_{djd} = m_d \qquad \forall\, d \in \mathcal{D} \tag{18a}$$

$$\sum_{d \in \mathcal{D}} \left\{ x_{djd} + \sum_{i \in C} x_{ijd} \right\} = 1 \qquad \forall\, j \in C \tag{18b}$$

$$x_{djd} + \sum_{i \in C} x_{ijd} = x_{jdd} + \sum_{i \in C} x_{jid} \qquad \forall\, d \in \mathcal{D},\, j \in C \tag{18c}$$

$$\sum_{j \in C} x_{djd} = \sum_{j \in C} x_{jdd} \qquad \forall\, d \in \mathcal{D} \tag{18d}$$

Constraints (18a) and (18d) ensure that exactly m_d salesmen depart and return to depot d. Constraints (18b) guarantee that each city are visited exactly once. Constraints (18c) ensure the path continuity. Together with the degree constraints (18b), constraints (18c) ensure that a salesman starts at depot d and visits city j first will either continue to another city i or return to the same depot. Note that this formulation uses $\mathrm{O}(\mathsf{L}^2\mathsf{D})$ binary variables, and the number of binary variables increases cubically with the number of depots (as opposed to the quadratic increase for 2-index formulations).

3.4.2. Multi-commodity flow formulation

A multi-commodity flow problem is a network flow problem with multiple flows [38]. Besides applications for cycle elimination (as discussed in Section 3.3.3) it was shown by Bektaş [4] that this concept can also be used for enforcing fixed-destination solutions to the mTSP. In this context a commodity f^d represents the number of salesmen originating from depot d. The constraints based on commodity flows [4] are given by

$$\sum_{j \in C \cup \mathcal{D}'} f^d_{dj} - \sum_{j \in \mathcal{D} \cup C} f^d_{jd} = m_d \qquad \forall\, d \in \mathcal{D} \tag{19a}$$

$$\sum_{j \in C \cup \mathcal{D}'} f^d_{ij} - \sum_{j \in \mathcal{D} \cup C} f^d_{ji} = 0 \qquad \forall\, i \in C, d \in \mathcal{D} \tag{19b}$$

$$\sum_{j \in \mathcal{D} \cup C} f^d_{jd'} - \sum_{j \in C \cup \mathcal{D}'} f^d_{d'j} = m_d \qquad \forall\, d' \in \mathcal{D}' \tag{19c}$$

$$0 \le f^d_{ij} \le x_{ij} \qquad \forall\, i, j \in \mathcal{L}', d \in \mathcal{D} \tag{19d}$$

In this formulation each depot d in $\mathcal{D}$ acts as a source of commodity f^d, while each depot d' in $\mathcal{D}'$ acts as a sink where only commodity $d = d' - \mathsf{L}$ is accepted. By (19a) exactly m_d units of commodity f^d will leave depot d (meaning that m_d salesmen will leave the depot). Constraints (19b) are flow-conservation constraints that guarantee that the same amount of commodity f^d entering a node i will also leave node i (meaning that each salesman that enters a city will also leave the city). By (19c) exactly m_d units of commodity f^d will reach depot d' (meaning that m_d salesmen will arrive at the duplicate depot node d'). Combined with the assignment constraints (9), the inequality constraints (19d) restrict the commodities to only flow along arcs that are part of the selected routes; if $x_{ij} = 0$ no commodity can flow from city i to city j. This formulation uses $\mathsf{L}'^2\mathsf{D}$ commodity flow variables f^d_{ij}, where $\mathsf{L}' = 2\mathsf{D} + \mathsf{C}$ is the number of nodes in the graph.

3.4.3. Path elimination constraints

The path elimination constraints, first proposed in [5], fix the destination of each salesman by eliminating paths that start and end in two different depots. The idea is inspired by the chain-baring constraints introduced in [28]. Although originally designed for location routing problems, path elimination constraints have been applied to many fixed-destination mTSP variants [6, 13, 44]. The decision variables are

$$x_{ij} = \begin{cases} 1 & \text{if } (i, j) \text{ is traversed exactly once,} \\ 0 & \text{otherwise,} \end{cases} \tag{20}$$

for $i, j \in \mathcal{L}$ and

$$w_{ij} = \begin{cases} 1 & \text{if } (i, j) \text{ is in a return trip,} \\ 0 & \text{otherwise,} \end{cases} \tag{21}$$

for all $i \in \mathcal{D}$, $j \in C$. The path elimination constraints given by [5] are

$$\sum_{p,q \in \mathcal{S} \cup \{i,j\}} x_{pq} + \sum_{d \in \mathcal{D}^\circ} x_{jd} + \sum_{d \in \mathcal{D} \setminus \mathcal{D}^\circ} x_{id} \le |\mathcal{S}| + 2 \tag{22}$$
$$\forall i, j \in C, \forall \mathcal{S} \subseteq C \setminus \{i, j\}, \forall \mathcal{D}^\circ \subset \mathcal{D}$$

If all cities in $\mathcal{S} \cup \{i, j\}$ are in a consecutive path, then the loop conditions (10) are satisfied with equality, i.e., $\sum_{p,q \in \mathcal{S} \cup \{i,j\}} = |S| + 1$. Because of constraints (22) we have $\sum_{d \in \mathcal{D}^\circ} x_{jd} + \sum_{d \in \mathcal{D} \setminus \mathcal{D}^\circ} x_{id} \le 1$. This indicates that a path connected to a depot in $\mathcal{D}^\circ$ cannot be connected to another depot in $\mathcal{D} \setminus \mathcal{D}^\circ$. Constraints (22) can eliminate all unwanted paths that visit at least two cities and start and end in different depots. The constraints

$$\sum_{d \in \mathcal{D}} x_{dj} + w_{dj} \le 1 \qquad \forall\, j \in C \tag{23}$$

are needed to also eliminate undesired paths that visit only one city (and start and end in different depots).This formulation requires $O(\mathsf{L}^2 + \mathsf{DC})$ binary variables, and the number of constraints ($O(2^{\mathsf{C}}\mathsf{C}^2 2^{\mathsf{D}})$ grows exponentially with the number of cities and depots. Just as for the loop conditions (10) these constraints are suitable for branch-and-cut implementations, but not to formulate the complete problem and use a MILP solver to obtain optimal solutions.

4. NOVEL FMmTSP FORMULATION

In the previous section it has been discussed that the FMmTSP can be formulated using four components (as provided in (4)). The *cost* that needs to be minimised is the total travel distance of the salesmen, for which a standard formulation is given in (5). For the *assignment constraints* the conventional constraints are given in (9), but variations can be used to e.g. allow some salesmen to be idle or to limit the number of salesmen that may end at a depot, as discussed in Section 3.2.2. The component that has the most variants in literature provides the *cycle elimination constraints* (or subtour elimination constraints), discussed in Section 3.3.

The component that has received the least attention provides the *cycle imposement constraints*. Until recently, fixed-destination solutions for mTSPs and its variants have been ensured by using 3-index

formulations of the decision variables as discussed in Section 3.4.1. The formulation of Bektaş proposed in [4] is the first formulation that ensures fixed-destination solutions using 2-index binary variables and the multi-commodity flow constraints presented in Section 3.4.2. In this section we will introduce a novel approach for cycle imposement. This approach is also based on 2-index binary variables, where one continuous variable per node is added to the formulation to ensure a fixed-destination solution. The new formulation needs a few less binary variables than the formulation of [4]; more importantly, it uses DL times less continuous variables than the multi-commodity flow approach.

4.1. Cycle Imposement through Node Currents

Inspired by the node potentials of Miller, Tucker, and Zemlin [30] we propose an alternative formulation of the FMmTSP using *node currents* [8]. Similar to the commodity flow transported between cities over the arcs, the current in an electric circuit can also be considered as a flow in a directed graph. With the depots representing current sources, a proper electric circuit contains only cycles (if not, there would be an open circuit or short circuit), which corresponds to the cycle imposement constraint stating that every salesman must return to his departing depot. A flow conservation law combined with assignment constraints forces the current flowing into a node to be equal to the current flowing out of the node, so that nodes in the same cycle must have the same current. Thus, we can view the current k_i as a property of node i (instead of a property of the arc x_{ij}).

4.1.1. Node current formulation

For the newly proposed node current formulation there is no need to use copies of the depot nodes. Therefore, this formulation will use less binary variables as for the copy-based formulation, since there are D less nodes to represent the graph. Fixed-destination solutions can be obtained by using $\mathsf{L} = \mathsf{D} + \mathsf{C}$ continuous variables k_i satisfying the cycle imposement constraints

$$k_d = d \quad \forall\, d \in \mathcal{D} \tag{24a}$$

$$k_i - k_j \le (\mathsf{D}-1)(1-x_{ij}) \;\forall\, i, j \in \mathcal{L} \tag{24b}$$

resulting in $k_i \le k_j$ if $x_{ij} = 1$ using $(\mathsf{D}+\mathsf{C})^2 - \mathsf{C}$ constraints. Additionally, one can obtain the tighter constraint $k_i = k_j$ if $x_{ij} = 1$ by adding

$$k_j - k_i \le (\mathsf{D}-1)(1-x_{ij}) \;\forall\, i, j \in \mathcal{L}. \tag{24c}$$

to the constraints (24a)–(24b), resulting in stronger linear relaxations at the cost of using more constraints. If the minimum number of cities to visit is set to be at least two one can substitute (24b)–(24c) with

$$k_i - k_j \le (\mathsf{D}-1)(1-x_{ij}-x_{ji}) \;\forall\, i, j \in \mathcal{L}. \tag{24d}$$

This enforces the equality $k_i = k_j$ using half the amount of inequality constraints. Notice that constraints (24d) exclude solutions where a salesman visits only one node, since $x_{ij} + x_{ji} = 2$ is infeasible by (24d).

Theorem 4.1 (Cycle imposement)**.** *The MILP consisting of* (5)*,* (9)*, any of the cycle elimination constraints, and the cycle imposement constraints* (24) *will result in a graph with exactly* $\sum_{d\in\mathcal{D}} m_d$ *cycles, where each node* $d \in \mathcal{D}$ *is contained in exactly* m_d *cycles.*

Proof. Let the directed graph $G = (\mathcal{L}, \mathcal{A})$ be the graph associated with a feasible solution of the given MILP. The node set of G coincides with the set of locations of the FMmTSP, and the arc set $\mathcal{A}$ is defined as

$$\forall i, j \in \mathcal{L},\ (i,j) \in \mathcal{A} \text{ if and only if } x_{ij} = 1$$

Define a cut $(\mathcal{D}, C)$ on G, and denote the subset of forward and backward arcs in the cut set as

$$\delta_+ = \{(i,j) \in \mathcal{A} \,|\, i \in \mathcal{D},\ j \in C\}$$
$$\delta_- = \{(i,j) \in \mathcal{A} \,|\, i \in C,\ j \in \mathcal{D}\}$$

which represents all salesmen leaving the depots and all salesmen returning from the cities, respectively. By assignment constraints on the city nodes (9b)–(9c), the in- and out- degree of each node in C is one, and the cycle elimination constraints ensure that no cycles exist in C, so no path can start or end in C. Therefore $|\delta_+| = |\delta_-|$, indicating that any salesman leaving a depot must also return to a depot (see Figure 2).

The above arguments actually show that the graph associated with a solution of the non-fixed destination MmTSP contains exactly $\sum_{d\in\mathcal{D}} m_d$ distinct paths starting and ending in $\mathcal{D}$. Now we need to prove that by the additional cycle imposement constraints for the fixed-destination setting, the $\sum_{d\in\mathcal{D}} m_d$ distinct paths are all cycles, and each node $d \in \mathcal{D}$ is contained in exactly m_d cycles, i.e., each path starting in a depot node $d \in \mathcal{D}$ must also end in the same depot d. We prove this statement by induction.

For any path $P = \{d, c_{i1}\, c_{i2} \dots d^*\}$, where $d, d^* \in \mathcal{D}$ and $c_{i1}, c_{i2} \cdots \in C$, by constraint (24b) that the node current k_i is non-decreasing along a path, one has

$$k_d \le k_{c_{i1}} \le k_{c_{i2}} \le \cdots \le k_{d^*}$$

In addition, by (24a) each depot node is assigned a unique node current, and hence

$$1 \leq k_d \leq D \; \forall d \in \mathcal{D}.$$

We use the following inductive steps to prove that any path starting in depot d must also end in the same depot.

i) By (9a) there will be m_D paths leaving depot D. For any path P starting in depot $d = D$, one has $D = k_d \leq k_{d^*} \leq D$, where the upper limit follows from the fact that a path must return to a depot, for which D is the highest value. Thus $k_{d^*} = D$, indicating $d^* = D = d$. Therefore, a path starting in node D can only end in node D. By (9a)–(9d) exactly m_D cycles start and end in depot D, i.e., depot D is contained in exactly m_D cycles, and can accept no more incoming arcs because the constraint (9d) on its in-degree is already satisfied.

ii) For any path P starting in depot $d = D - 1$, similarly one has $D - 1 = k_d \leq k_{d^*} \leq D$. So k_{d^*} can only take the value D or $D - 1$, i.e., path P can only end in depot D or $D - 1$. By the previous argument P cannot end in D because (9d) is already satisfied for $d = D$, so the m_{D-1} paths starting at depot $D - 1$ can only end in depot $D - 1$. Similarly, by the assignment constraints, depot $D - 1$ is contained in exactly m_{D-1} cycles, and can accept no more incoming arcs.

iii) Continuing this argumentation for any path P starting in depot d, P can only end in the same depot d since the constraint (9d) is already satisfied for depots $d + 1$ to D, and by the assignment constraints depot d is contained in exactly m_d cycles.

iv) Finally, it follows that any path P starting in depot 1 must end in depot 1.

□

By assigning a unique value to the node currents of the depots through (24a) and adding constraints (24b)–(24c) or (24d)—such that $k_i = k_j$ if there is a connection between nodes i and j— it is guaranteed that a tour starting at depot d will return to depot d without visiting another depot by Theorem 4.1.

Note. For the optimal solution the node current variables will implicitly satisfy

$$1 \leq k_i \leq D \; \forall i \in \mathcal{L}, \tag{25}$$

and these bounds can be set explicitly in the MILP formulation without affecting the result.

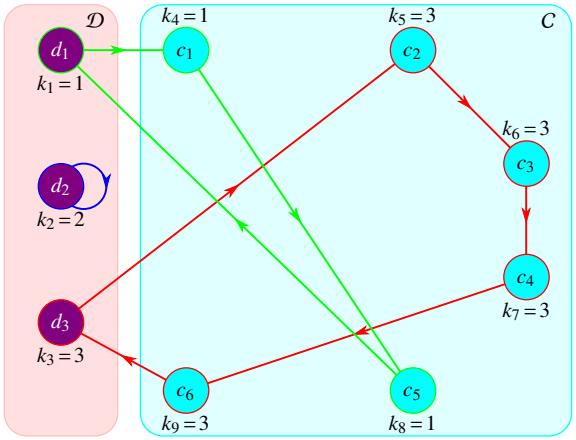

Figure 2: *A solution without a copy of the set $\mathcal{D}$, using node currents to ensure a fixed-destination solution. Each of the three cycles has a unique 'current': the depot nodes act as current sources of 1, 2 and 3 Ampère respectively, and this current is flowing through the arcs and nodes. Since each node has exactly one incoming arc and one outgoing arc (due to the assignment constraints) this 'node current' uniquely defines to which depot a city is assigned.*

Figure 2 shows an example of a feasible solution for $\mathsf{D} = 3$ depots and $\mathsf{C} = 6$ cities. Note that within the set $\mathcal{L} = \mathcal{D} \cup C$ the existence of three cycles has been imposed, whereas in the set C no cycles exist due to the cycle elimination constraints.

4.1.2. MILP formulation using node currents

As an alternative to the FMmTSP formulation presented in [4] we propose a novel formulation of the problem based on node currents as cycle imposement constraints. It is based on 2-index decision variables using the cost function given by (5). The formulation will be presented using the standard assignment constraint given in (9), but the variant with idle salesmen may also be used. For a comparison with the formulation in [4] (which excludes the possibility of idle salesmen) it should be possible to set workload bounds for the salesmen, therefore the cycle elimination constraints (12) are chosen. For the computational comparison in the next section the minimum number of cities to visit per salesman will be $\underline{u} = 1$, therefore we use constraints (24a–24c) to impose $\sum_{d \in \mathcal{D}} m_d$ cycles in the set $\mathcal{L}$; if $\underline{u} \geq 2$ it would be more efficient to use (24a) and (24d). The maximum number of cities per salesman can be set to

$$\overline{u} = \mathsf{C} + \underline{u}(1 - \sum_{d \in \mathcal{D}} m_d) \tag{26}$$

where $\sum_{d \in \mathcal{D}} m_d$ gives the total number of salesmen, such that $\underline{u}(1 - \sum_{d \in \mathcal{D}} m_d)$ becomes the minimum number of

cities that need to be visited by other salesmen; a single salesman can visit at most all cities minus the minimum number of cities visited by the others. The MILP formulation of the FMmTSP using workload bounds and node currents then becomes

$$
\begin{aligned}
\text{min.} \quad & \sum_{i\in\mathcal{L}}\sum_{j\in\mathcal{L}} \mathsf{c}_{ij}x_{ij} && (27)\\
\text{s.t.} \quad & \sum_{i\in\mathcal{L}} x_{ic} = \sum_{j\in\mathcal{L}} x_{cj} = 1 && \forall\, c\in C\\
& \sum_{i\in\mathcal{L}} x_{id} = \sum_{j\in\mathcal{L}} x_{dj} = m_d && \forall\, d\in\mathcal{D}\\
& u_i - u_j + \overline{\mathsf{u}}x_{ij} + (\overline{\mathsf{u}}-2)x_{ji} \le \overline{\mathsf{u}}-1 && \forall\, i,j\in C\\
& u_i + (\overline{\mathsf{u}}-2)\sum_{d\in\mathcal{D}} x_{di} - \sum_{d\in\mathcal{D}} x_{id} \le \overline{\mathsf{u}}-1 && \forall\, i\in C\\
& u_i + \sum_{d\in\mathcal{D}} x_{di} + (2-\underline{\mathsf{u}})\sum_{d\in\mathcal{D}} x_{id} \ge 2 && \forall i\in C\\
& k_d = d && \forall\, d\in\mathcal{D}\\
& k_i - k_j \le (\mathsf{D}-1)(1-x_{ij}) && \forall\, i,j\in\mathcal{L}\\
& k_j - k_i \le (\mathsf{D}-1)(1-x_{ij}) && \forall\, i,j\in\mathcal{L}\\
& x_{ij}\in\{0,1\},\ 1\le u_i\le \mathsf{C},\ 1\le k_i\le \mathsf{D} && \forall\, i,j\in\mathcal{L}
\end{aligned}
$$

Note. As opposed to the two alternative FMmTSP formulations [4, 31], this novel formulation does not use copies of the depot nodes, therefore only $\mathsf{L} = \mathsf{D} + \mathsf{C}$ nodes are needed for this formulation instead of $2\mathsf{D}+\mathsf{C}$. Furthermore, unlike the other two formulations, the costs of travelling between the nodes remains the same as for the nonfixed-destination problem, hence there is no need to build a new cost matrix; the original cost matrix $\mathbf{C}$ can be used without any modification.

4.2. Properties of Fixed-Destination Formulations

Adding more variables to an optimisation problem in general results in larger computation times and a higher memory usage. Compared to continuous variables the number of binary variables used in a programming problem can significantly influence the computation times. Therefore, reducing the number of binary variables to represent a problem can result in a noticeable performance gain. Although in general the addition of a few continuous variables has little influence on the computation times, using many continuous variables can cause problems due to the larger memory use, and would also result in larger computation times.

Table 3 shows the number of nodes, binary variables, continuous variables, equality constraints, and inequality constraints that are needed to represent the FMmTSP per formulation. In the following 'I' denotes the novel MILP formulation (27), 'II' denotes the extended formulation based on [32], and 'III' denotes the formulation from [4].

Table 3: **Properties of the three formulation types, divided into the number of nodes (N), binary variables (BV), continuous variables (CV), equality constraints (EC), and inequality constraints (IC).**

	I	II	III
N	L=C+D	L′=C+2D	L′=C+2D
BV	$(\mathsf{C}+\mathsf{D})^2$	$(\mathsf{C}+\mathsf{D})^2$	$(\mathsf{C}+\mathsf{D})^2$
CV	2C+2D	2C+4D	(DL′+1)L′
EC	2C+3D	2C+4D	2C+ (4+L′)D
IC	C^2+2C+2L^2	C^2+2C+2L'^2	C^2+2C+DL'^2

Notice that besides less binary variables, the newly proposed formulation also uses less continuous variables compared to the formulation of [4]; there are $\mathsf{D}+\mathsf{C}$ node currents necessary to solve the fixed-destination problem, compared to $\mathsf{D}(2\mathsf{D}+\mathsf{C})^2$ commodity flow parameters needed to represent the D different commodities that could move along the $(2\mathsf{D}+\mathsf{C})^2$ arcs in the extended network.

5. COMPUTATIONAL COMPARISON

The three aforementioned formulations for solving FMmTSPs are compared by solving a large number of test cases. First we describe the benchmark that we use, followed by a discussion on the results. The formulations provide optimal solutions for the FMmTSP, and all computation times give the time it took to reach this optimum. When the optimum was not reached within 3 hours wall-clock time the test was marked as failed.

5.1. Description of Test Instances

To compare the three formulations of the FMmTSP we have chosen 32 symmetric and asymmetric TSP test cases with size ranging from 14 to 170 nodes from the library TSPLIB [40], where the numbers in the name of the test instance (e.g. `dantzig42`) represent the number of locations L in the problem. For each test case we have selected D cities to represent depots. Since e.g. the cities in `dantzig42` are given in the order of the optimal tour (and hence subsequent cities are close to each other), the depot nodes are selected as the i-th cities satisfying

$$i = 1 + (d-1)\left\lfloor \frac{\mathsf{C}}{\mathsf{D}} \right\rfloor \quad \forall d\in\mathcal{D}, \qquad (28)$$

where $\lfloor a \rfloor$ represents the operator that returns the largest integer smaller than or equal to a. This approach is used to reduce the chance that the depots are close to

each other. The number of depots D varies from 2 to 6 for each test case. We consider two scenarios, namely, one-salesman-at-each-depot and multiple-salesmen-at-each-depot (The number of salesmen at each depot for the second scenario is fairly distributed using the procedure described in Appendix A.) In this way we create a benchmark of $32 \times 5 \times 2 = 320$ FMmTSP test instances, which are solved using three different formulations, and three MILP solvers.

Since the formulation using the commodity flows according to (19) requires that each salesman visits at least one city (due to (19a)), we set $\underline{\mathsf{u}} = 1$ and $\overline{\mathsf{u}} = \mathsf{C}$ to obtain the same problem for each formulation. All computations are performed on a desktop computer with an Intel Xeon E5-1620 Quad Core CPU and 64 GB of RAM, running 64-bit versions of SUSE Linux Enterprise Desktop 11, and Matlab R2014b. Three state-of-the-art commercial and free MILP solvers are used, namely, CPLEX 12.5 (called via Tomlab 8.0), Gurobi Optimizer 5.6, and CBC 2.9.4 from COIN-OR (called via the OPTI-Toolbox).

The results for the one-salesman-at-each-depot problem are reported in Tables B.7–B.9, followed by the results for the multiple-salesmen-at-each-depot problem in Tables B.10–B.12 of Appendix B. To reduce the chance that the outcome is affected by random events, we chose to run each test case a few times and take the average value of the computation times. Tables B.10–B.12 of Appendix B contain the average CPU time to find the optimal solution over 10 runs for each small test case, and over 5 runs for each large test case, for each number of depots D. A time limit of 3 hours is imposed on each test run, and all reported times are in seconds.

5.2. Comparison of Problem Formulations

When a test case is solved to optimality within 3 hours wall-clock time, we register this time. Otherwise, we mark the test case as failed. A comparison is made between the three formulations on both the average CPU times and the number of failed cases.

5.2.1. Comparison of average CPU times

To compare the three problem formulations, we have split the benchmark into four sets:

- Small problems with a single salesman per depot
- Large problems with a single salesman per depot
- Small problems with multiple salesmen per depot
- Large problems with multiple salesmen per depot

The first 16 test cases (`burma14` up to `ry48p`) are considered small problems, while the last 16 test cases (`hk48` up to `ftv170`) are included in the large problems. Table 4 shows the relative increase in CPU time needed to compute the solution compared to formulation I. For the FMmTSP with a single salesman per depot, formulation I was the fastest on average; for the variant with multiple salesmen per depot formulation II outperformed the other two. Although formulation II uses a few more binary values than formulation I, *it cannot be concluded from our results that the use of more binary variables results in larger computation times*.

Table 4: **The average relative difference for the mean CPU times compared with formulation I. The average is taken over all test instances that are successfully computed by the corresponding formulation and solver. A positive result means longer computation time, indicating worse performance.**

	Solver	Small & single	Large & single	Small & multi	Large & multi
II	CPLEX	7%	192%	−27%	455%
	Gurobi	51%	7%	−13%	−9%
	CBC	46%	33%	−18%	-
III	CPLEX	535%	1880%	104%	720%
	Gurobi	113%	78%	37%	24%
	CBC	621%	1676%	130%	-

Notice that the difference between formulation I and II is small (I is less than 1.5 times faster than II for all averages), but formulation III is significantly slower on average when using CPLEX or CBC, even for the small instances (where memory use is not yet expected to be a problem); for Gurobi the differences are smaller. Nevertheless, we conclude that the use of *node current formulations are expected to be faster than multi-commodity-based formulations for fixed-destination problems*.

5.2.2. Comparison of failed test cases

Next we compare how often a test case did not reach an optimal solution in time. We distinguish between the results for a single salesman per depot and for multiple salesmen per depot. For each formulation we provide the number of failed cases (per solver) in Table 5.

From Table 5, it is clear that formulation II demonstrates stronger ability to solve large test cases. Formulation III also performs rather well in solving large test cases when there are multiple salesmen at each depot, but it has problems for cases with a single salesman per depot. CPLEX and Gurobi seem to perform equally well, but also here it becomes clear that CBC cannot match the other two solvers.

Table 5: **The number of failed test instances ('failed') and the size (i.e., the number of nodes) of the largest instance successfully solved ('largest') for each formulation (I, II, and III) solved per solver type. 'Single' means one-salesman-at-each-depot, and 'multiple' means multiple-salesmen-at-each-depot. The number before '/' is the number of failed test instances out of the 160 that were performed.**

	Solver	Single		Multiple	
		Failed	Largest	Failed	Largest
I	CPLEX	9/160	124	37/160	124
	Gurobi	12/160	170	40/160	124
	CBC	49/160	76	92/160	42
II	CPLEX	14/160	124	8/160	124
	Gurobi	15/160	124	11/160	124
	CBC	51/160	64	90/160	42
III	CPLEX	30/160	124	25/160	124
	Gurobi	24/160	124	19/160	124
	CBC	52/160	64	90/160	42

6. CONCLUDING REMARKS

In this article we have provided a brief overview of cycle elimination and imposement constraints, and 2-index formulations for the fixed-destination multi-depot travelling salesman problem. A novel cycle imposement constraint formulation has been proposed based on node currents, which can be seen as the dual of the node potentials of Miller, Tucker, and Zemlin [30]. The main advantage of the novel formulation over the existing formulations is the reduced number of binary and continuous variables needed to formulate the problem. Furthermore, the novel formulation can be used to find solutions where several salesmen can be idle.

The comparisons of the formulations have been performed using three state-of-the-art MILP solvers. Similar to the node potential constraints (11), the node current constraints can be used to easily formulate (variants of) fixed-destination multi-depot problems. This approach is suitable for the initial development of formulations; once it is confirmed that the formulation provides the desired solutions, one can use more sophisticated techniques, e.g. Benders' decomposition [7], to reformulate the problems and solve them faster. The proposed formulation was able to solve problems up to 170 nodes using general MILP solvers, and [4] found optimal solutions up to 170 nodes using Benders' decomposition. By using a branch-and-cut algorithm [6] even managed to solve (symmetric) problems up to 255 cities and 25 depot to optimality within reasonable time. It would be interesting to see whether improvements can be obtained by using the node currents combined with e.g. Benders' decomposition or user-specified cuts. Furthermore, the proposed formulation for FMmTSP can be applied to other scheduling and routing problems.

Appendix A. Allocation of Salesmen over Depots

A three-step procedure is described to allocate salesmen for the multiple-salesmen-at-each-depot scenario.

Step 1: Compute the number of cities $C = L - D$, and generate the total number of salesmen to be assigned to the D depots

$$S = \min\left(\max\left(D+1, \left\lfloor \frac{L}{3} \right\rfloor\right), C-1\right)$$

We choose $\left\lfloor \frac{L}{3} \right\rfloor$ for the total number of salesmen (as long as it lies in the interval $[D+1, C-1]$), since too few salesmen are insufficient to consider the multiple-salesmen-at-each-depot scenario, and too many salesmen can lead to idle salesmen in the solution.

Step 2: Assign $x = \left\lfloor \frac{S}{D} \right\rfloor$ salesmen to each depot, and calculate the number of the unassigned salesmen

$$r = S - x \cdot D$$

Step 3: Assign one salesman to the depots with index

$$i = 1 + (k-1)\left\lfloor \frac{D}{r} \right\rfloor \quad \forall k \in \{1, 2, \cdots r\}$$

Since the number of remaining salesmen calculated at *Step 2* is always less than D, all salesmen have been assigned to a depot after performing the three-step procedure. Moreover, the last step also ensures a fair allocation of the remaining salesmen.

Appendix B. List of Results

The optimal values for the benchmark described in Appendix A are provided in Table B.6. These values are obtained by summarising all instances successfully solved by three MILP solvers (CPLEX, Gurobi, and CBC) and three formulations under a 3-hour time limit. A complete list of tables with the average CPU time (in seconds) for all test instances solved by the three afore mentioned formulations, and three different MILP solvers is presented in Table B.7-B.12. The fastest instances are indicated by the bold-faced numbers, and the symbol "-" is used to denote the failed test instances.

Table B.6: **Summary of optimal values obtained using three solvers and three formulations.**

	one-salesman-at-each-depot					multiple-salesmen-at-each-depot				
test case	D=2	D=3	D=4	D=5	D=6	D=2	D=3	D=4	D=5	D=6
burma14	3098	3033	2993	3480	3728	3253	3079	3039	3696	3944
ulysses16	6986	6326	6097	5809	8862	8324	6873	6101	5816	9774
gr17	2054	1819	1945	1684	1815	2374	2056	2182	1818	1953
br17	36	23	16	6	34	39	23	16	6	34
gr21	2716	2662	2674	2684	3228	3623	3454	3932	3004	3360
ulysses22	6445	6282	6435	6147	6855	12520	6769	7701	6386	6888
fri26	930	939	940	932	903	1442	1385	1143	1093	1061
bayg29	1596	1598	1641	1583	1608	1955	1972	1962	1867	1851
bays29	1988	1993	2018	1972	1966	2471	2469	2476	2369	2351
ftv33	1302	1291	1292	1237	1214	1831	1732	1536	1402	1490
ftv35	1457	1453	1429	1396	1421	2157	1999	1801	1918	1735
ftv38	1521	1510	1518	1455	1512	2297	2158	1962	1941	1966
dantzig42	661	633	645	611	631	1202	1016	977	795	813
swiss42	1272	1262	1274	1277	1257	2054	1982	1640	1583	1604
ftv44	1611	1602	1608	1569	1582	2744	2232	2546	2205	2171
ry48p	14097	14318	14366	14306	14146	23925	22637	19560	18506	19378
hk48	11439	11358	11222	11285	11310	18259	19258	14989	14697	13717
eil51	426.358	423.013	424.452	431.966	423.28	712.039	624.878	554.267	549.326	558.837
berlin52	7464.36	7591.94	7528.97	7501.83	7733.97	11896.5	14113.6	9754.51	9460.11	9444.27
ft53	6926	7029	6880	6842	6896	12211	12012	9376	8857	8943
ftv55	1590	1585	1594	1623	1584	2905	2715	2865	2666	2279
ftv64	1782	1835	1798	1770	1846	3052	3067	2722	2935	2601
st70	671.792	667.264	659.917	654.026	655.046	1423.55	1166.02	1023.21	1029.61	884.627
eil76	542.325	541.004	540.436	555.114	551.761	941.31	898.858	866.489	864.664	795.141
gr96	54795	55076	54047	54653	54999	130169	126797	101320	88968	92425
kroB100	21954.8	21698.8	21695.2	21680.6	21415.8	47224.9	40190.7	38024.8	32726.8	33430.7
kroC100	20504.7	20400.2	20308.2	20321.8	20434	49947.6	39621.9	40213.4	35719.9	32695.2
kroD100	21493	-	-	20792.4	-	60888.7	47247.5	36873.3	36597.8	37458.2
kroE100	21896.9	21932.2	21700.5	21865	21386.3	46588.1	42585.9	35477.3	33493	37063.9
eil101	641.711	637.213	636.541	641.933	640.554	1353.47	1414.26	1098.31	973.447	973.366
kro124p	36316	36244	36252	36041	36427	69844	67146	-	58912	53206
ftv170	2755	2740	-	2744	-	-	-	-	-	-

Table B.7: **Mean CPU time (in seconds) obtained from the CPLEX solver, for scenario one-salesman-at-each-depot.**

type	test case	D=2	D=3	D=4	D=5	D=6	test case	D=2	D=3	D=4	D=5	D=6
I	burma14	0.50	0.19	0.19	0.31	0.40	hk48	**4.30**	32.14	**1.34**	**0.45**	1.97
II		**0.21**	0.25	**0.16**	**0.23**	**0.15**		8.06	33.00	2.26	0.79	**1.54**
III		0.34	**0.18**	0.34	0.26	0.24		114.81	**25.21**	10.05	4.00	36.04
I	ulysses16	1.16	**0.30**	**0.17**	**0.23**	392.31	eil51	**1.86**	2.25	**1.88**	13.28	2.18
II		**0.57**	0.31	0.23	0.23	**0.32**		2.48	**1.69**	2.49	**9.05**	**1.54**
III		21.30	0.37	0.35	0.37	0.43		70.44	29.06	15.39	152.15	26.55
I	gr17	0.36	0.36	0.33	**0.20**	0.25	berlin52	1.71	**5.58**	**1.22**	**1.65**	28.17
II		**0.32**	**0.27**	**0.29**	0.29	**0.21**		**1.58**	5.98	1.85	2.68	**9.55**
III		1.39	0.35	0.32	0.26	0.34		264.08	27.24	14.69	12.67	110.63
I	br17	**0.39**	**0.35**	0.25	0.30	0.30	ft53	**2.31**	67.92	**2.04**	**2.17**	**1.36**
II		0.43	0.39	0.26	**0.24**	**0.28**		4.54	**20.50**	3.84	2.72	1.67
III		1.09	0.61	**0.25**	0.34	0.31		866.05	1024.84	52.29	46.70	18.25
I	gr21	0.23	0.25	0.24	0.25	1.79	ftv55	**1.19**	**1.08**	**1.51**	5.87	12.48
II		**0.23**	**0.20**	**0.19**	**0.21**	**0.22**		1.93	1.37	2.34	**2.37**	**0.85**
III		0.41	0.28	0.27	0.34	0.38		4.13	9.30	17.74	7.31	28.34
I	ulysses22	**0.44**	**0.28**	**0.51**	**0.30**	3.62	ftv64	1.68	**7.01**	2.51	**1.30**	14.36
II		0.55	0.37	0.59	0.41	**0.38**		**1.56**	13.07	**1.67**	3.09	**3.09**
III		1.86	0.40	0.92	0.64	0.73		21.04	59.80	22.06	9.54	1107.09
I	fri26	**0.54**	**0.45**	**0.41**	0.51	**0.33**	st70	**107.87**	**45.59**	**146.17**	**106.91**	**25.82**
II		0.63	0.77	0.55	**0.48**	0.52		253.25	82.69	171.05	122.70	35.93
III		4.68	0.92	1.17	1.34	1.35		527.36	92.28	698.98	448.71	131.62
I	bayg29	**0.38**	**0.57**	3.01	0.36	1.15	eil76	**10.83**	**11.52**	**5.57**	3034.21	1060.30
II		0.45	0.67	**2.72**	**0.33**	**0.86**		14.81	15.08	8.32	**25.54**	**6.06**
III		2.62	1.18	9.71	0.85	2.98		71.22	126.64	120.74	444.12	98.20
I	bays29	0.45	**0.39**	**1.30**	**0.30**	**0.46**	gr96	**52.71**	232.55	**23.88**	158.68	405.34
II		**0.40**	0.53	1.46	0.30	0.75		126.06	**129.70**	51.32	**97.39**	**181.88**
III		1.87	0.73	9.85	0.75	1.77		366.28	2506.87	342.34	-	-
I	ftv33	**0.71**	2.76	**0.53**	**0.38**	**0.32**	kroB100	**715.08**	253.79	**10249.36**	**159.69**	**418.96**
II		0.91	**1.04**	0.73	0.40	0.32		1100.80	**223.31**	-	265.20	915.75
III		73.28	1.52	1.43	1.77	1.33		-	2192.25	-	-	-
I	ftv35	**0.49**	0.54	**0.37**	0.34	**0.31**	kroC100	**1079.32**	**198.89**	**175.56**	759.76	2653.58
II		0.57	**0.51**	0.44	**0.29**	0.37		2032.30	299.03	199.45	**632.67**	**1007.79**
III		2.18	1.24	1.61	1.68	1.70		6054.43	7025.49	-	-	-
I	ftv38	0.77	0.60	**0.47**	0.48	**0.62**	kroD100	-	**-**	**-**	**8739.53**	**-**
II		**0.62**	**0.59**	0.70	**0.34**	0.67		**2995.81**	-	-	-	-
III		3.86	1.63	1.70	1.83	3.15		6718.39	-	-	-	-
I	dantzig42	**3.25**	**0.75**	**1.80**	**0.87**	**0.61**	kroE100	**440.11**	**222.39**	254.65	**676.93**	**44.51**
II		3.96	1.40	2.25	1.55	0.66		798.82	238.56	**163.79**	946.49	-
III		36.54	2.06	6.93	6.10	5.23		-	-	-	-	-
I	swiss42	**1.50**	**1.13**	**1.42**	**1.75**	**1.27**	eil101	**78.90**	**26.07**	**133.80**	**71.34**	71.21
II		1.84	1.65	2.41	1.92	1.87		338.65	33.88	143.56	117.90	**58.03**
III		33.78	6.40	6.61	16.46	9.48		6377.94	-	-	-	-
I	ftv44	1.17	**0.67**	1.27	0.58	0.95	kro124p	**25.41**	**34.76**	**22.02**	**11.13**	**33.28**
II		**0.99**	0.96	**0.82**	**0.58**	**0.62**		1154.00	2196.56	-	-	-
III		3.26	1.99	6.16	4.47	3.79		1151.96	2195.11	-	-	-
I	ry48p	**1.52**	**7.30**	**8.02**	**5.60**	6.46	ftv170	-	-	-	-	-
II		2.32	13.40	11.75	9.48	**1.23**		-	-	-	-	-
III		37.00	32.92	36.70	44.33	11.96		-	-	-	-	-

Table B.8: **Mean CPU time (in seconds) obtained from the Gurobi Optimizer, for the scenario one-salesman-at-each-depot.**

type	test case	D=2	D=3	D=4	D=5	D=6	test case	D=2	D=3	D=4	D=5	D=6
I	burma14	**0.07**	0.07	**0.04**	0.22	0.68	hk48	7.17	**86.31**	2.55	**0.23**	**3.10**
II		0.13	**0.06**	0.09	**0.10**	**0.04**		**3.66**	107.00	**2.22**	0.41	5.32
III		0.14	0.08	0.08	0.14	0.09		4.27	105.61	5.51	0.59	4.86
I	ulysses16	2.05	**0.15**	0.05	**0.07**	461.46	eil51	2.69	3.97	**0.99**	32.10	4.03
II		**1.02**	0.26	**0.04**	0.08	0.27		**1.86**	2.89	1.53	38.29	**2.56**
III		1.59	0.27	0.11	0.24	**0.26**		3.86	**2.78**	4.56	**21.79**	3.32
I	gr17	**0.19**	**0.14**	**0.18**	**0.03**	0.11	berlin52	2.06	10.88	**1.94**	**1.84**	49.24
II		0.35	0.17	0.28	0.05	**0.10**		**1.10**	**8.64**	2.86	3.90	**21.73**
III		1.08	0.27	0.40	0.08	0.18		3.50	29.66	6.74	5.95	29.06
I	br17	**1.14**	0.78	**0.21**	**0.11**	0.33	ft53	28.29	56.11	**22.43**	19.16	36.85
II		2.71	**0.43**	0.24	0.22	**0.32**		**7.13**	46.61	43.77	**11.04**	**6.97**
III		2.52	0.70	0.32	0.19	0.43		28.82	**39.73**	35.95	111.07	11.81
I	gr21	**0.13**	**0.06**	**0.16**	**0.07**	3.04	ftv55	1.84	**1.69**	3.10	10.94	18.37
II		0.21	0.09	0.23	0.10	**0.07**		**1.60**	2.28	**2.12**	**2.32**	**1.59**
III		0.20	0.20	0.29	0.45	0.11		4.15	3.98	4.17	4.12	4.33
I	ulysses22	**0.46**	**0.18**	**0.42**	**0.23**	3.19	ftv64	1.84	**4.22**	2.56	**1.46**	21.49
II		0.67	0.22	0.47	0.55	**0.46**		**1.62**	4.96	**1.68**	3.17	**5.62**
III		0.70	0.31	1.26	0.47	0.48		1.95	12.61	4.02	2.82	8.09
I	fri26	**0.40**	**0.72**	**0.97**	**0.51**	**0.38**	st70	**636.50**	**241.88**	**1133.48**	**379.68**	**29.23**
II		0.76	0.98	1.01	0.67	0.48		-	261.64	1506.71	475.47	84.12
III		0.87	1.24	1.30	1.36	0.80		1595.30	307.17	4683.77	1098.81	189.27
I	bayg29	**0.73**	**0.55**	4.45	**0.47**	2.27	eil76	9.48	11.43	4.68	10547.85	2205.36
II		0.88	0.99	**3.41**	1.23	**1.03**		**6.85**	**10.49**	**2.50**	**14.06**	**2.51**
III		0.95	1.13	5.70	1.18	2.07		13.21	27.92	4.30	48.19	8.06
I	bays29	**0.38**	**0.57**	**2.81**	0.42	1.32	gr96	349.41	957.70	**97.39**	**362.25**	**329.48**
II		0.54	1.09	2.92	0.40	**1.00**		**209.83**	**376.86**	128.38	443.95	666.62
III		0.69	0.96	3.84	**0.30**	1.26		259.31	1196.67	124.34	439.49	-
I	ftv33	**0.55**	**0.85**	**0.91**	**0.37**	**0.17**	kroB100	6013.72	**420.09**	**-**	1580.18	**2098.60**
II		2.12	1.17	1.18	0.87	0.27		**5341.60**	1115.09	-	**1071.68**	4772.04
III		1.85	1.24	3.19	0.65	0.46		-	1648.63	-	-	-
I	ftv35	**0.64**	**0.63**	0.31	**0.23**	**0.18**	kroC100	-	-	319.52	**2720.50**	**-**
II		1.17	0.97	**0.26**	0.37	0.24		-	**2370.29**	**308.45**	5469.71	-
III		1.27	0.71	0.54	0.63	0.49		**5304.34**	-	2094.30	-	-
I	ftv38	**0.94**	0.79	0.62	0.27	**0.71**	kroD100	-	**-**	**-**	**-**	**-**
II		1.31	1.22	**0.61**	**0.26**	1.55		**8377.61**	-	-	-	-
III		1.00	**0.64**	1.09	0.51	0.98		10581.25	-	-	-	-
I	dantzig42	**4.67**	**0.77**	2.39	1.84	**0.56**	kroE100	-	1189.62	800.29	**6460.41**	**303.35**
II		4.98	2.17	**1.27**	**1.55**	0.63		**10475.87**	**1143.61**	**675.25**	8845.54	525.64
III		4.77	2.44	1.52	1.59	0.98		-	3482.50	1186.41	-	-
I	swiss42	2.37	**1.45**	**1.70**	**2.80**	**1.49**	eil101	229.58	317.55	242.76	**140.20**	189.96
II		**2.33**	1.77	3.20	3.63	2.89		**125.33**	26.50	**126.40**	157.08	**47.72**
III		5.27	5.63	4.97	5.00	5.09		421.55	**17.71**	356.45	-	-
I	ftv44	**0.61**	**0.81**	1.51	**0.32**	0.83	kro124p	**30.17**	**254.38**	**24.80**	**16.82**	**269.64**
II		1.23	0.97	**0.81**	0.85	**0.62**		74.38	336.59	77.14	-	-
III		0.76	1.25	1.48	1.02	1.02		74.46	337.20	77.08	-	-
I	ry48p	3.35	**8.76**	**14.18**	**6.82**	7.52	ftv170	**32.24**	**26.90**	-	24.86	-
II		**2.84**	14.54	77.48	14.98	**2.16**		-	-	-	-	-
III		3.89	18.89	17.72	29.67	3.86		-	-	-	-	-

Table B.9: **Mean CPU time (in seconds) obtained from the CBC solver, for scenario one-salesman-at-each-depot. As the largest test case that CBC can solve for this scenario is `eil76`, we have truncated the table to make it more concise.**

type	test case	D=2	D=3	D=4	D=5	D=6	test case	D=2	D=3	D=4	D=5	D=6
I	burma14	0.79	**0.41**	**0.39**	1.55	49.48	dantzig42	**133.87**	19.19	**30.06**	**23.48**	**4.92**
II		**0.78**	0.44	0.57	0.70	0.75		356.50	**15.07**	89.85	128.51	5.17
III		2.55	0.72	0.99	**0.55**	**0.60**		-	75.43	339.87	126.71	70.08
I	ulysses16	33.17	1.97	**0.32**	**0.51**	6772.37	swiss42	**27.77**	79.93	**21.82**	**41.73**	**16.61**
II		**11.18**	**0.86**	0.42	0.72	**1.35**		45.40	**25.81**	60.22	48.75	27.37
III		28.21	1.67	0.35	1.78	2.42		598.24	68.51	72.12	132.23	102.62
I	gr17	**1.19**	**0.65**	**0.52**	**0.32**	**0.45**	ftv44	23.38	**5.13**	237.46	**8.41**	20.77
II		1.68	1.04	1.05	0.50	0.92		**9.08**	9.95	**24.70**	13.49	**11.77**
III		3.56	1.36	0.82	1.31	1.04		65.52	38.73	30.67	106.26	104.84
I	br17	**599.43**	46.45	**0.54**	**0.58**	33.90	ry48p	**20.75**	1775.21	**729.87**	**222.86**	1302.83
II		618.82	**23.23**	0.83	0.89	**1.00**		216.58	**321.03**	4956.45	348.75	**17.26**
III		1995.82	31.05	2.50	1.91	1.10		577.59	875.94	1188.08	1306.19	220.15
I	gr21	**0.64**	**0.54**	0.99	1.05	25.46	hk48	**333.89**	-	75.05	**6.73**	43.39
II		0.77	0.65	1.89	**1.00**	0.84		406.06	**7452.41**	152.95	11.88	**11.87**
III		1.32	4.39	**0.96**	3.52	**0.79**		2509.90	7621.84	**51.88**	33.55	174.17
I	ulysses22	**5.87**	**0.93**	5.31	**1.64**	70.65	eil51	51.23	34.59	44.43	935.99	247.47
II		6.23	1.34	**2.27**	2.25	**5.73**		**47.27**	**21.94**	**36.56**	853.15	**14.74**
III		31.50	4.18	16.83	7.09	6.54		2116.50	70.25	76.43	**490.93**	101.66
I	fri26	**1.99**	**6.20**	**7.97**	9.51	**3.93**	berlin52	13.95	**129.52**	**27.39**	106.86	3666.44
II		7.10	8.37	14.70	**7.04**	7.99		**11.10**	362.80	53.40	**86.09**	548.08
III		140.05	11.24	11.38	12.23	16.64		9137.46	1977.97	135.74	355.46	**423.07**
I	bayg29	**5.31**	**8.11**	67.46	3.75	**15.35**	ft53	159.03	-	9468.16	**121.40**	85.57
II		8.51	15.64	**67.00**	**2.06**	23.53		**152.91**	**527.96**	-	130.30	**24.68**
III		29.43	13.70	104.49	8.47	24.26		-	1926.62	**1004.57**	439.18	85.58
I	bays29	**4.46**	11.88	42.83	3.24	78.02	ftv55	**16.37**	**14.44**	36.75	259.52	614.58
II		7.22	16.14	**22.97**	**2.20**	11.73		116.94	27.70	**22.57**	**31.31**	**15.38**
III		24.90	**10.34**	37.78	12.36	**9.43**		2539.82	86.93	86.52	337.31	314.63
I	ftv33	**12.61**	**10.90**	**7.60**	**2.83**	**1.52**	ftv64	**36.75**	**192.80**	627.11	**27.18**	879.96
II		14.44	15.71	17.40	9.78	3.04		93.52	225.63	**104.97**	34.65	**98.39**
III		319.31	15.58	22.57	59.12	64.11		2311.47	211.36	318.15	650.50	770.92
I	ftv35	**6.44**	**3.34**	**4.68**	**2.33**	**3.18**	st70	**-**	**-**	**-**	**-**	**-**
II		21.41	5.17	5.55	3.45	3.83		-	-	-	-	-
III		177.13	13.57	17.46	17.31	30.32		-	-	-	-	-
I	ftv38	**7.81**	**4.58**	**6.90**	**3.13**	9.90	eil76	**1246.47**	**402.44**	**69.14**	**-**	**-**
II		15.28	8.56	8.37	4.55	**6.24**		-	-	-	-	-
III		39.18	39.36	8.82	40.58	12.55		-	-	-	-	-

Table B.10: **Mean CPU time (in seconds) obtained from the CPLEX solver, for the scenario multiple-salesmen-at-each-depot.**

type	test case	D=2	D=3	D=4	D=5	D=6	test case	D=2	D=3	D=4	D=5	D=6
I		0.26	**0.16**	**0.21**	0.48	0.54		**1.92**	-	**0.66**	1.28	1.97
II	burma14	**0.22**	0.29	0.23	**0.22**	**0.16**	hk48	2.06	**3.36**	1.10	**1.16**	**1.85**
III		0.29	0.27	0.26	0.28	0.21		4.26	9.11	5.00	11.19	19.29
I		6.91	**0.23**	**0.14**	0.33	1037.06		**1.01**	**0.82**	0.49	7.12	83.13
II	ulysses16	**0.20**	0.27	0.17	**0.24**	**0.25**	eil51	2.22	2.01	**0.43**	**3.84**	**1.45**
III		0.26	0.33	0.23	0.32	0.39		4.44	11.14	2.04	27.18	14.75
I		**0.36**	0.33	**0.33**	**0.18**	**0.23**		1.15	3.85	**2.05**	**1.63**	588.83
II	gr17	0.42	**0.29**	0.35	0.23	0.23	berlin52	**0.83**	**1.69**	2.57	1.65	**2.68**
III		0.44	0.56	0.40	0.37	0.31		6.12	16.82	13.62	11.57	37.65
I		0.35	**0.28**	0.27	**0.20**	0.28		-	2305.90	**2.07**	96.97	**2.40**
II	br17	**0.32**	0.29	**0.21**	0.24	**0.25**	ft53	**1.93**	336.30	4.82	**5.75**	2.66
III		0.45	0.33	0.27	0.37	0.39		44.95	**31.36**	49.45	16.60	10.19
I		0.40	0.45	91.74	0.29	1.30		4.16	-	2627.80	-	1158.37
II	gr21	**0.22**	**0.27**	**0.21**	**0.20**	**0.17**	ftv55	**0.98**	**2.57**	**5.96**	**19.94**	**1.93**
III		0.35	0.27	0.25	0.28	0.28		10.74	23.80	7.58	21.99	9.09
I		0.31	**0.25**	**0.42**	0.49	2.12		1.20	**1.10**	91.39	2.79	-
II	ulysses22	**0.29**	0.31	0.44	**0.36**	**0.31**	ftv64	**0.56**	1.69	**1.14**	**0.78**	**4.01**
III		0.43	0.31	0.82	0.64	0.63		1.70	5.86	4.56	5.50	38.52
I		**0.38**	0.73	**0.42**	**0.45**	0.44		-	-	**27.42**	-	120.53
II	fri26	0.40	**0.34**	0.44	0.59	**0.42**	st70	**22.42**	**10.68**	27.66	**8.85**	**49.54**
III		0.59	0.69	1.06	0.89	1.04		50.52	101.39	159.51	144.87	343.32
I		**0.28**	**0.27**	**0.38**	0.66	**0.49**		**3.66**	51.39	**2.50**	-	-
II	bayg29	0.40	0.37	0.64	**0.31**	0.65	eil76	4.72	**1.38**	2.74	**2.37**	**2.96**
III		0.42	0.41	1.28	0.82	1.45		32.96	7.01	67.98	45.81	43.07
I		0.25	0.33	**0.55**	1.04	0.55		136.12	-	-	-	-
II	bays29	**0.21**	**0.32**	0.61	**0.46**	**0.41**	gr96	**70.37**	**32.37**	5271.15	**216.89**	**400.03**
III		0.33	0.62	1.36	0.74	1.54		430.93	739.18	**3902.65**	-	-
I		5.32	**0.55**	**0.39**	**0.19**	0.67		**11.80**	42.46	-	**154.68**	-
II	ftv33	**0.38**	0.74	0.44	0.27	**0.42**	kroB100	19.18	**29.77**	**324.27**	232.42	**1647.09**
III		0.88	1.19	0.98	0.53	2.17		85.15	477.31	-	-	-
I		**0.34**	0.69	0.43	1.38	0.28		35.14	-	5360.70	-	-
II	ftv35	0.43	**0.48**	**0.41**	**0.34**	**0.26**	kroC100	**32.06**	**575.64**	**178.00**	**70.52**	**253.15**
III		0.51	0.88	1.03	1.11	0.79		340.68	1199.03	-	-	-
I		88.85	**0.64**	5.76	0.55	3.32		-	-	33.37	-	-
II	ftv38	**0.28**	0.73	**0.47**	**0.54**	**0.55**	kroD100	**62.94**	**226.68**	**18.18**	**6993.22**	**157.58**
III		0.59	1.95	2.90	1.27	3.07		312.03	923.29	-	-	-
I		0.47	0.43	**1.23**	**0.86**	0.72		271.89	-	6675.83	-	-
II	dantzig42	**0.41**	**0.34**	2.64	1.26	**0.57**	kroE100	**143.93**	**138.07**	**242.76**	**8398.49**	**58.45**
III		1.13	1.21	11.39	2.73	11.66		479.70	858.16	-	-	-
I		1.41	554.87	1.73	10.63	13.95		-	-	23.64	-	-
II	swiss42	**0.49**	**2.11**	**1.61**	**1.05**	**1.49**	eil101	**45.14**	**19.24**	**16.01**	**52.81**	**18.28**
III		1.20	11.93	10.22	2.41	21.49		350.63	357.35	-	-	-
I		1233.89	1.32	1968.36	**0.54**	1.11		**4.03**	**5.51**	-	**2768.97**	**39.11**
II	ftv44	**0.35**	**0.51**	**1.06**	0.64	**0.40**	kro124p	185.08	1410.72	-	-	-
III		1.29	1.21	4.36	2.84	1.44		184.42	1412.88	-	-	-
I		**1.25**	3691.25	16.56	52.54	-		-	-	-	-	-
II	ry48p	2.06	**3.36**	**4.29**	**1.34**	**1.21**	ftv170	-	-	-	-	-
III		7.43	12.42	28.10	8.19	11.64		-	-	-	-	-

Table B.11: **Mean CPU time (in seconds) obtained from Gurobi Optimizer, for the scenario multiple-salesmen-at-each-depot.**

type	test case	D=2	D=3	D=4	D=5	D=6	test case	D=2	D=3	D=4	D=5	D=6
I		**0.04**	**0.04**	**0.04**	0.52	0.61		2.21	-	**0.48**	**0.99**	**2.71**
II	burma14	0.04	0.05	0.05	**0.05**	**0.04**	hk48	**1.23**	**1.52**	0.68	2.05	2.84
III		0.05	0.07	0.09	0.09	0.08		3.14	5.46	1.74	2.10	3.04
I		6.14	0.14	**0.02**	**0.11**	1054.08		0.83	2.58	0.50	17.94	164.84
II	ulysses16	0.08	**0.09**	0.04	0.17	**0.24**	eil51	**0.79**	**0.95**	**0.30**	**2.86**	**1.70**
III		**0.06**	0.23	0.07	0.27	0.30		2.27	2.03	0.63	6.36	2.04
I		**0.16**	**0.13**	**0.21**	**0.03**	0.12		1.70	3.96	3.41	**2.09**	1013.67
II	gr17	0.25	0.25	0.30	0.05	**0.07**	berlin52	**1.48**	**1.61**	**3.19**	2.79	**4.77**
III		0.79	0.39	0.37	0.12	0.16		13.82	2.04	8.20	3.48	14.19
I		**0.30**	**0.07**	**0.11**	**0.03**	0.27		6739.40	-	**20.23**	133.20	**16.10**
II	br17	0.52	0.07	0.13	0.04	**0.11**	ft53	6.66	65.09	55.45	**9.69**	21.73
III		1.20	0.11	0.17	0.10	0.26		**3.40**	**33.89**	106.06	64.09	16.45
I		**0.12**	0.36	105.00	0.18	1.38		6.28	-	3558.18	-	5136.91
II	gr21	0.22	**0.31**	0.17	**0.11**	**0.07**	ftv55	**1.04**	**3.00**	6.88	32.69	5.11
III		0.35	0.36	**0.14**	0.22	0.17		1.88	3.56	**2.48**	**6.14**	**2.65**
I		0.26	0.13	**0.33**	**0.34**	3.75		0.31	1.59	67.29	3.23	-
II	ulysses22	**0.14**	**0.08**	0.65	0.35	**0.29**	ftv64	**0.26**	**0.68**	**1.10**	2.24	**5.07**
III		0.20	0.12	0.83	0.55	0.51		0.79	1.35	1.89	**1.27**	6.26
I		**0.31**	1.55	0.72	**0.48**	**0.35**		-	-	**97.57**	-	**878.29**
II	fri26	0.43	**0.63**	**0.56**	0.78	0.91	st70	56.77	**16.84**	223.73	**29.98**	2060.49
III		0.41	1.00	0.68	0.65	0.51		**54.24**	45.69	161.56	135.73	1274.02
I		**0.07**	**0.21**	**0.45**	0.57	**0.77**		**1.82**	50.45	7.30	-	-
II	bayg29	0.11	0.22	0.90	**0.49**	1.23	eil76	2.82	1.30	**1.68**	**2.06**	**1.40**
III		0.21	0.39	1.45	0.81	0.91		5.60	**0.79**	6.57	5.28	2.98
I		0.14	**0.24**	**0.60**	1.67	1.35		**266.89**	-	-	-	-
II	bays29	**0.13**	0.35	1.01	0.71	1.01	gr96	560.08	**32.93**	-	1685.22	**344.50**
III		0.29	0.43	1.15	**0.68**	**0.92**		277.99	50.03	**658.95**	**1055.28**	-
I		3.85	**0.57**	0.57	**0.11**	0.99		40.80	**68.72**	-	**546.46**	**-**
II	ftv33	**0.65**	0.87	**0.46**	0.19	1.08	kroB100	**17.19**	147.15	**805.84**	1425.71	-
III		1.51	1.37	0.75	0.29	**0.69**		39.94	256.84	-	-	-
I		0.52	0.74	0.68	1.86	0.30		141.21	-	-	-	-
II	ftv35	0.65	**0.49**	**0.47**	**0.25**	**0.23**	kroC100	**46.60**	**675.18**	**697.69**	**245.39**	**2182.16**
III		**0.39**	0.58	0.83	0.50	0.33		103.36	933.91	848.71	-	-
I		89.25	1.20	11.33	0.57	3.15		-	-	57.62	**-**	-
II	ftv38	**0.59**	**0.66**	1.43	**0.26**	1.01	kroD100	**125.71**	1042.29	**33.18**	-	**262.00**
III		0.66	0.81	**1.14**	0.46	**0.66**		388.16	**903.60**	67.51	-	-
I		**0.33**	**0.25**	**1.03**	1.62	**1.38**		810.50	-	-	**-**	-
II	dantzig42	0.39	0.68	1.68	**0.90**	1.94	kroE100	**322.32**	**373.75**	732.20	-	**84.27**
III		0.64	0.89	1.06	1.44	1.40		526.81	421.44	**387.38**	-	-
I		1.00	750.93	**2.02**	17.79	15.17		-	-	19.28	-	-
II	swiss42	**0.40**	**3.11**	2.93	**0.78**	**3.01**	eil101	**31.80**	**13.56**	**3.77**	**48.96**	**11.67**
III		1.15	9.78	4.58	2.55	4.63		41.80	66.04	5.72	-	-
I		2981.10	3.50	2019.71	1.48	3.72		10.60	11.34	-	-	**192.54**
II	ftv44	**0.34**	0.84	1.86	1.21	**0.47**	kro124p	**9.69**	10.08	81.48	-	-
III		0.91	**0.61**	**1.05**	**0.93**	0.55		9.73	**10.02**	**81.07**	-	-
I		**1.85**	3081.24	22.57	112.09	-		-	-	-	-	-
II	ry48p	3.87	**2.89**	**3.07**	**1.71**	**1.08**	ftv170	-	-	-	-	-
III		2.57	3.14	4.28	2.26	1.80		-	-	-	-	-

Table B.12: **Mean CPU time (in seconds) obtained from the CBC solver, for the scenario multiple-salesmen-at-each-depot. As the largest test case that CBC can solve for this scenario is `swiss42`, we have truncated the table to make it more concise.**

type	test case	D=2	D=3	D=4	D=5	D=6	test case	D=2	D=3	D=4	D=5	D=6
I		0.40	**0.42**	**0.40**	2.45	15.72		**0.60**	**1.03**	**5.67**	76.38	26.24
II	burma14	**0.37**	0.45	0.53	**0.52**	**0.43**	bayg29	0.70	1.29	7.53	**2.39**	**10.80**
III		0.80	0.48	0.95	0.54	0.71		5.24	8.82	8.73	19.08	22.07
I		96.08	**0.64**	**0.32**	**0.46**	-		**0.64**	1.89	**3.03**	66.69	12.45
II	ulysses16	**0.71**	0.80	0.44	0.60	1.62	bays29	1.58	**1.42**	6.97	**1.70**	**3.29**
III		0.92	1.38	0.48	0.74	**0.93**		4.26	2.09	10.62	11.07	17.07
I		**1.63**	**0.82**	6.61	**0.33**	**0.50**		121.84	**3.58**	8.75	1.61	39.19
II	gr17	2.60	0.93	2.42	0.54	0.86	ftv33	**2.57**	5.49	**3.86**	2.17	**7.21**
III		13.86	2.66	**1.29**	1.28	1.30		17.29	16.61	9.76	**1.39**	29.50
I		**4.84**	0.79	**0.53**	**0.33**	6.96		**1.97**	67.43	8.85	86.78	1.45
II	br17	5.41	**0.68**	1.31	0.50	**0.75**	ftv35	3.51	**4.45**	**6.10**	**4.80**	2.28
III		39.25	1.35	1.62	0.50	1.83		3.03	20.41	26.28	52.22	**1.21**
I		**0.87**	2.79	2147.21	1.95	50.15		2896.58	**15.63**	277.51	3.42	439.83
II	gr21	1.07	1.35	1.30	**0.92**	0.76	ftv38	**5.05**	18.83	**12.75**	4.86	**6.74**
III		3.46	**1.18**	**1.19**	2.35	**0.49**		9.85	25.90	62.36	**2.68**	25.37
I		1.26	**1.18**	4.80	**1.47**	96.28		**3.84**	6.43	**20.49**	44.62	**15.31**
II	ulysses22	**1.00**	1.71	**3.96**	1.51	**6.83**	dantzig42	7.19	**5.12**	24.19	**13.97**	18.96
III		6.85	3.00	6.94	2.65	12.51		17.89	5.83	54.73	120.24	82.66
I		3.18	17.85	11.20	**5.32**	5.82		35.59	-	73.03	765.07	305.13
II	fri26	**1.88**	**5.37**	11.99	6.44	**2.94**	swiss42	**6.65**	**78.95**	**37.61**	**8.37**	239.81
III		37.25	18.56	**8.25**	8.83	11.83		30.72	96.86	85.50	10.98	**98.13**

References

[1] Applegate, D. L., Bixby, R. E., Chvatal, V., Cook, W. J., 2006. The Traveling Salesman Problem - A Computational Study. Princeton University Press.

[2] Ascheuer, N., Fischetti, M., Grötschel, M., 2001. Solving the asymmetric travelling salesman problem with time windows by branch-and-cut. Mathematical Programming 90 (3), 475–506.

[3] Bektaş, T., 2006. The multiple traveling salesman problem: an overview of formulations and solution procedures. Omega 34 (3), 209 – 219.

[4] Bektaş, T., 2012. Formulations and Benders decomposition algorithms for multidepot salesmen problems with load balancing. European Journal of Operational Research 216 (1), 83 – 93.

[5] Belenguer, J.-M., Benavent, E., Prins, C., Prodhon, C., Calvo, R. W., 2011. A branch-and-cut method for the capacitated location-routing problem. Computers & Operations Research 38 (6), 931 – 941.

[6] Benavent, E., Martínez, A., 2013. Multi-depot multiple TSP: a polyhedral study and computational results. Annals of Operations Research 207 (1), 7–25.

[7] Benders, J. F., 1962. Partitioning procedures for solving mixed-variables programming problems. Numerische Mathematik 4 (1), 238–252.

[8] Burger, M., 2014. Exact and compact formulation of the fixed-destination travelling salesman problem by cycle imposement through node currents. In: Huisman, D., Louwerse, I., Wagelmans, A. P. (Eds.), Operations Research Proceedings 2013: Selected Papers of the International Conference on Operations Research, OR2013, organized by the German Operations Research Society (GOR), the Dutch Society of Operations Research (NGB) and Erasmus University Rotterdam, September 3-6, 2013. Springer International Publishing, Cham, pp. 83–88.

[9] Burger, M., Huiskamp, M., Keviczky, T., Apr. 2013. Complete field coverage as a multiple harvester routing problem. In: Proc. of the 4th IFAC Conference on Modelling and Control in Agriculture, Horticulture and Post Harvest Industry. Espoo, Finland, pp. 97–102.

[10] Burger, M., Schutter, B. D., Hellendoorn, J., June 2012. Micro-ferry scheduling problem with time windows. In: 2012 American Control Conference (ACC). pp. 3998–4003.

[11] Claus, A., 1984. A new formulation for the travelling salesman problem. SIAM Journal on Algebraic Discrete Methods 5 (1), 21–25.

[12] Coelho, L. C., Laporte, G., 2013. The exact solution of several classes of inventory-routing problems. Computers & Operations Research 40 (2), 558 – 565.

[13] Contardo, C., Martinelli, R., 2014. A new exact algorithm for the multi-depot vehicle routing problem under capacity and route length constraints. Discrete Optimization 12, 129 – 146.

[14] Cook, W. J., 2012. In Pursuit of the Traveling Salesman: Mathematics at the Limits of Computation. Princeton University Press.

[15] Dantzig, G., Fulkerson, R., Johnson, S., 1954. Solution of a large-scale traveling-salesman problem. Journal of the Operations Research Society of America 2 (4), 393–410.

[16] Desrochers, M., Laporte, G., 1991. Improvements and extensions to the Miller-Tucker-Zemlin subtour elimination constraints. Operations Research Letters 10 (1), 27 – 36.

[17] Doppstadt, C., Koberstein, A., Vigo, D., 2016. The hybrid electric vehicle–traveling salesman problem. European Journal of Operational Research 253 (3), 825–842.

[18] Finke, G., Claus, A., Gunn, E., May 1984. A two-commodity network flow arroach to the traveling salesman problem. Congressus Numerantium 41, 167–178.

[19] Fox, K. R., Gavish, B., Graves, S. C., Aug. 1980. An n-constraint formulation of the time-dependent traveling salesman problem. Operations Research 28 (4), 1018–1021.

[20] Gavish, B., Graves, S. C., 1978. The travelling salesman problem and related problems. Tech. rep., Massachusetts Institute of Technology.

[21] Harris, J., Hirst, J. L., Mossinghoff, M., 2008. Combinatorics and Graph Theory, 2nd Edition. Undergraduate Texts in Mathematics. Springer-Verlag New York.

[22] Helsgaun, K., 2000. An effective implementation of the Lin-Kernighan traveling salesman heuristic. European Journal of Operational Research 126, 106–130.

[23] Irawan, C., Ouelhadj, D., Jones, D., Stålhane, M., Sperstad, I., 2017. Optimisation of maintenance routing and scheduling for offshore wind farms. European Journal of Operational Research 256 (1), 76–89.

[24] Kara, I., Bektaş, T., 2006. Integer linear programming formulations of multiple salesman problems and its variations. European Journal of Operational Research 174 (3), 1449–1458.

[25] Kulkarni, R., Bhave, P., 1985. Integer programming formulations of vehicle routing problems. European Journal of Operational Research 20 (1), 58 – 67.

[26] Langevin, A., Soumis, F., Desrosiers, J., 1990. Classification of travelling salesman problem formulations. Operations Research Letters 9 (2), 127 – 132.

[27] Laporte, G., Jun. 1992. The traveling salesman problem: An overview of exact and approximate algorithms. European Journal of Operational Research 59 (2), 231–247.

[28] Laporte, G., Nobert, Y., Arpin, D., 1986. An exact algorithm for solving a capacitated location-routing problem. Annals of Operations Research 6 (9), 291–310.

[29] Lin, S., Kernighan, B. W., 1973. An effective heuristic algorithm for the traveling-salesman problem. Operations Research 21 (2), 498–516.

[30] Miller, C. E., Tucker, A. W., Zemlin, R. A., Oct. 1960. Integer programming formulation of traveling salesman problems. Journal of the Association for Computing Machinery 7 (4), 326–329.

[31] Oberlin, P., Rathinam, S., Darbha, S., June 2009. A transformation for a heterogeneous, multiple depot, multiple traveling salesman problem. In: 2009 American Control Conference. pp. 1292–1297.

[32] Oberlin, P., Rathinam, S., Darbha, S., June 2009. A transformation for a multiple depot, multiple traveling salesman problem. In: 2009 American Control Conference. pp. 2636–2641.

[33] Öncan, T., Altinel, I. K., Laporte, G., 2009. A comparative analysis of several asymmetric traveling salesman problem formulations. Computers & Operations Research 36 (3), 637 – 654.

[34] Orman, A., Williams, H., 2007. A survey of different integer programming formulations of the travelling salesman problem. In: Kon-

toghiorghes, E. J., Gatu, C. (Eds.), Optimisation, Econometric and Financial Analysis. Vol. 9 of Advances in Computational Management Science. Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 91–104.
[35] Padberg, M., Sung, T.-Y., May 1991. An analytical comparison of different formulations of the travelling salesman problem. Mathematical Programming 52 (1), 315–357.
[36] Parragh, S., Doerner, K., Hartl, R., 2008. A survey on pickup and delivery problems part I: Transportation between customers and depot. Journal für Betriebswirtschaft 58 (1), 21–51.
[37] Parragh, S., Doerner, K., Hartl, R., 2008. A survey on pickup and delivery problems part II: Transportation between pickup and delivery locations. Journal für Betriebswirtschaft.
[38] Rader, D. J., 2010. Deterministic Operations Research: Models and Methods in Linear Optimization. John Wiley & Sons, Inc.
[39] Ramkumar, N., Subramanian, P., Narendran, T. T., Ganesh, K., Dec. 2012. Mixed integer linear programming model for multi-commodity multi-depot inventory routing problem. OPSEARCH 49 (4), 413–429.
[40] Reinelt, G., 1991. TSPLIB - a traveling salesman problem library. ORSA Journal on Computing 3 (4), 376–384.
[41] Roberti, R., Toth, P., 2012. Models and algorithms for the asymmetric traveling salesman problem: an experimental comparison. EURO Journal on Transportation and Logistics 1 (1), 113–133.
[42] Røpke, S., Cordeau, J.-F., 2009. Branch and cut and price for the pickup and delivery problem with time windows. Transportation Science 43 (3), 267–286.
[43] Savelsbergh, M. W. P., Sol, M., Feb. 1995. The general pickup and delivery problem. Transportation Science 29 (1), 17–29.
[44] Sundar, K., Rathinam, S., 2016. Generalized multiple depot traveling salesmen problem—polyhedral study and exact algorithm. Computers & Operations Research 70, 39 – 55.
[45] Toth, P., Vigo, D., nov 2002. Models, relaxations and exact approaches for the capacitated vehicle routing problem. Discrete Applied Mathematics 123 (1–3), 487–512.
[46] Toth, P., Vigo, D. (Eds.), 2002. The Vehicle Routing Problem. Monographs on Discrete Methematics and Applications. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA.
[47] Wong, R., 1980. Integer programming formulations of the traveling salesman problem. In: IEEE International Conference of Cirtuits and Computers. New York, USA, pp. 149–152.