sustainability

MDPI

Article

A Flow-Based Formulation of the Travelling Salesman Problem with Penalties on Nodes

Przemysław Kowalik [1,*], Grzegorz Sobecki [2], Piotr Bawoł [2] and Paweł Muzolf [3]

[1] Department of Quantitative Methods in Management, Faculty of Management, Lublin University of Technology, ul. Nadbystrzycka 38, 20-618 Lublin, Poland
[2] General Military Department, Military University of Technology, ul. Gen. Sylwestra Kaliskiego 2, 00-908 Warszawa, Poland
[3] Faculty of Civil Engineering and Geodesy, Military University of Technology, ul. Gen. Sylwestra Kaliskiego 2, 00-908 Warszawa, Poland
* Correspondence: p.kowalik@pollub.pl

Abstract: The travelling salesman problem (TSP) is one of combinatorial optimization problems of huge importance to practical applications. However, the TSP in its "pure" form may lack some essential issues for a decision maker—e.g., time-dependent travelling conditions. Among those shortcomings, there is also a lack of possibility of not visiting some nodes in the network—e.g., thanks to the existence of some more cost-efficient means of transportation. In this article, an extension of the TSP in which some nodes can be skipped at the cost of penalties for skipping those nodes is presented under a new name and in a new mathematical formulation. Such an extension can be applied as a model for transportation cost reduction due to the possibility of outsourcing deliveries to some nodes in a TSP route. An integer linear programming formulation of such a problem based on the Gavish–Graves-flow-based TSP formulation is introduced. This formulation makes it possible to solve the considered problem by using any integer linear programming optimization software. Numerical examples and opportunities for further research are presented.

Keywords: travelling salesman problem; integer linear programming; penalties

Citation: Kowalik, P.; Sobecki, G.; Bawoł, P.; Muzolf, P. A Flow-Based Formulation of the Travelling Salesman Problem with Penalties on Nodes. *Sustainability* **2023**, *15*, 4330. https://doi.org/10.3390/su15054330

Academic Editor: Armando Cartenì

Received: 17 November 2022
Revised: 22 February 2023
Accepted: 23 February 2023
Published: 28 February 2023

1. Introduction

The flow of goods is one of pillars of the modern economy [1–3]. In today's world, the role of transport is extremely important, and effective supply chain management is more and more often recognized as a key factor in gaining a competitive advantage on the market [4,5]. The transport needs of enterprises result from many different reasons. These are mainly geographical differences in the locations of production plants or sales points, resulting from production or product specialization, economies of scale, or the political and social situation [6,7]. The territorial gap between the seller and the buyer can only be filled efficiently thanks to appropriate transport organization, so that the demand for goods can be met.

Costs play a key role in shaping the transport policy, mainly due to the fact that they constitute a significant share of the company's total logistic costs [8,9] and thus have a direct impact on its efficiency [10,11]. Moreover, the paradigm of sustainable transport development additionally places emphasis on issues related to environmental protection [12], exhaust emissions [13], road safety [14–16], sustainable use of natural resources [17], etc. These factors also affect transport costs because of possible environmental and material losses [18].

Such a significant share of transport costs in the logistics costs of enterprises forces the optimization of this area, and the search for solutions that may, on the one hand, reduce the financial burden, and on the other hand, maintain the desired quality of transport services [19–21]. Their value is influenced by many factors, not all of which are modifiable.

Fixed costs are independent of the volume of the shipping service. They include the costs of insurance, licenses and permits, leasing and some taxes [22,23]. On the other hand, the value of the variable costs is directly related to the volume of the transport service and generally increases as the distance travelled increases. The variable costs include the costs of fuel, energy, tires, repairs, overhauls, inspections, spare parts and road or rail tolls [24,25]. Therefore, they are influenced by the distance travelled (between the locations of loading and unloading), the length and type of transport route, the type of means of transport used and the duration of the transport service. These are the factors that the company can modify, and the effectiveness of these treatments may be of key importance in shaping the efficient logistics strategy [22–25].

The search for the optimal route is one of the key problems of logistic management in every enterprise [26,27]. Its difficulty is related to the necessity of taking into account many different criteria, often ones that are mutually contradictory, in order to find the best solution for the enterprise, including satisfaction of its customers. Finding the right solution requires a thorough analysis of factors related to transport costs, transport limitations and travel time, but also customer requirements [28]. The correct selection of the route gives the company the opportunity to save time and reduce costs, and also to ensure timely delivery, which should result finally in customer satisfaction.

Therefore, the search for the best transport routes is one of the basic goals of logistics activities. Routing methods are systematically developed and improved. They include, for example, the classical Hitchcock transportation problem, the travelling salesman problem, the vehicle routing problem, etc. However, whereas many problems of that kind can be formulated relatively simply, they may be very hard to solve, since they may even belong to the class of NP-hard problems. Due to the computational complexity of the abovementioned problems, usually some optimization methods must be used in order to determine optimal solutions. Among the optimization techniques used, we distinguish exact algorithms that calculate an exact, optimal solution for a given instance of a problem, and approximation algorithms that may calculate an exact solution but may also calculate an approximate, suboptimal solution with a given accuracy. Approximation methods can be divided into heuristic and metaheuristic ones [29,30]. Heuristics is a specialized method of solving a specific problem that finds good (i.e., not far from optimal ones) solutions with an acceptable computational effort, but without the guarantee of finding an optimal solution. Metaheuristics is a general method that serves as a framework for constructing a heuristic that solves any problem that can be described by certain concepts defined by this method. Such methods are not used to solve specific problems, but only provide a way to create an appropriate heuristic algorithm [31,32].

Determining the routes for delivery (or pickup) of some goods is generally always possible, if enough necessary resources (vehicles, staff, time for completing the service) are available. However, if the set of delivery points is somehow "unusual"—for example, if one or a few particular delivery locations are relatively very distant from the most of the other delivery locations—then the optimization in terms of total costs leads to the conclusion that skipping such locations may result in a significant decrease in the cost. However, due to contractual liabilities, usually, no location can be skipped. If one could be, it usually would mean that either the customer must be rewarded for not receiving the contracted service, or the supply to this location should be carried out using other transport options (e.g., outsourcing the service or using an alternative, more economical means of transport) when constructing the entire route. Any of the abovementioned options is obviously connected with some cost (the reward for not performing the contracted service or the cost of outsourcing the service), which can be commonly called a penalty. If the value of the penalty associated with skipping some location is low enough, it may be profitable to skip that location. As has been already mentioned, the opportunity to skip some locations initially scheduled to be visited obviously will be the most advantageous when some very-distant, isolated locations simple cost savings, planning shorter routes can

result in further advantages, such as less usage of vehicles or shorter work hours for drivers. The latter, in turn, could help in providing better safety and resilience to unexpected delays.

The problem considered in this article is determining a closed route (i.e., the route for which the start locations and the end locations are the same). Moreover, locations along the route can be skipped if positive penalties are charged for skipping. It is an extension of the traveling salesman problem (TSP)—a problem of calculating the shortest cycle connecting all the nodes in a graph (the shortest Hamiltonian cycle), where "node" is a graph theory term which stands for "delivery location". From a practical point of view, Hamiltonian cycles are routes for trips in which a person starts from some chosen point, e.g., a city, visits some other cities, each once only, and finally returns to the first city.

The history of the TSP as an optimization problem, which is important for its practical logistic application, dates back to 1930's, when it was formulated under the name the "messenger problem" [33]. The name "traveling salesman problem" appeared for the first time in 1949 [34]. In the 1950s, linear programming started to be used to solve the TSP [35–37]. Other approaches to solving the TSP which appeared already in the early years of its investigation were heuristic algorithms [38,39] and dynamic programming [40–43].

During the development of studies on TSP, it turned out that computational efficiency was not the only important issue for practical applications. Real-world route planning must often include such factors as the variability of parameters over time (non-constant cost/travelling time parameters, the limited availability of nodes, forced sequences of visiting some nodes), which resulted in creating such extensions of the TSP as the time-dependent travelling salesman problem (TDTSP) [44–47] and the travelling salesman problem with time windows (TSPTW) [48–51]. Necessities related to real-world application even led to a "relaxation" of probably the most distinguishing and apparently unchangeable assumption of the TSP, namely, visiting each node in the network exactly once. Such a "relaxation" can include visiting some nodes possibly more than once (TSP with multiple visits—TSPM) [52] or not visiting some nodes at all, which is a case investigated in this article, and it is described in more detailed later on. Another reason for extending the TSP is related to cases for which there are necessities of distinguishing some groups of nodes and defining specific rules of visiting them. Examples of such extensions are the generalized travelling salesman problem (GTSP)—calculating the shortest cycle which visits one node only in each of many disjoint groups of nodes [53]; and the clustered travelling salesman problem (CTSP)—calculating the shortest cycle in which all nodes of a group (cluster) must be visited before moving to another group (cluster) [54]. A "fusion" of the above problems is the clustered generalized travelling salesman problem (CGTSP), in which groups—clusters of nodes—are divided into subclusters, and exactly one node in each subcluster must be visited before moving to another cluster. The CGSTP is an example that TSP extensions appear with development of new technologies because its practical applications are optimizing automated storage and retrieval [55]. The assumption of the availability of one salesman only can also be dropped what results in formulating a multiple traveling salesman problem (m-TSP) [56,57]. It turns out that m-TSP can be applied to an optimization problem which is not a route planning but a production planning problem, namely, unrelated parallel machine scheduling problem with machine and job sequence setup times (UPMS) [58].

An example of investigating the node skipping in TSP came from a real-world problem which appeared in a M.Sc. thesis [59] supervised by one of the authors (P. Kowalik). The author of the thesis considered route planning for a wholesale company based in Lublin, Poland which supplied professional cleaning chemicals and appliances. Some deliveries were contracted to multiple destinations in adjacent regions (voivodships). The delivery options were to use the company's own van with fuel cost estimated to be 0.40 PLN/km or to hire a courier company which could deliver a normalized package of up to 31.5 kg at a price of 19.52 PLN to any address in Poland (PLN stands for Polish zloty, 2009 prices). The above data mean that delivering one package to a destination via a courier company instead of using the company's van was cheaper if the total length of the route was shortened by at

least 48.8 km. This "threshold" distance was valid if the fuel cost only was considered and could be lower if other costs were also taken into account. It turned out that other costs may also matter. In the considered case, some TSP routes were so long that they could not be completed in one day, which resulted in the extra cost of a hotel stay for the driver. On the other hand, eliminating/skipping one or a few nodes at the cost of courier deliveries shortened the driven route and made it possible to avoid the hotel costs and to cut down the total cost significantly (much more than the fuel cost itself). Planning the routes was not, however, based on any formal optimization model handling skipping the nodes (and no specialized software supporting such model was used either). Instead, it was performed by testing various route variants using TSP routing software. More precisely, removing and adding selected nodes was performed "manually" by a manager who was studying the layout of the calculated route on the map of the area.

The abovementioned case became the inspiration for creating its mathematical model in a formal way. The research of the investigation of the subject showed that the extension of the TSP considered in this article has been known since the 1980s. More precisely, we consider a problem formulated in 1987 in [60] under the name, the generalized travelling salesman problem (GTSP). However, as it has been mentioned earlier, in the literature on TSP extensions, the same name is used for a different modification of the TSP—calculating the shortest cycle which visits one node only in each of many disjoint groups of nodes [53]. Indeed, the GTSP, as defined in [53], is also a modification of the standard TSP in which some nodes some are skipped, but where there is special rule of visiting and skipping nodes described above and no penalties are imposed on skipping the nodes.

The considered problem was also formulated in 1993 in [61] as a variant of so-called prize collecting traveling salesman problem (PCTSP). The PCTSP is an extension of the TSP defined in [62], in which there are not only penalties for skipping nodes, but also prizes for visiting the remaining nodes, and there is an additional constraint: "the total prize collected on all the visited nodes must be not less than the predefined level". In [61] the prize collecting component of the PCTSP was removed. The name "PCTSP" applied to the considered problem is also incorrect, since this problem does not involve prize collecting in any sense. The existing names introduced in [60,61] are misleading, and the above naming inconsistences occurring in the literature are the reason for introducing a new name, the traveling salesman problem with penalties on nodes (TSPPN), for the problem under consideration.

A novelty is its formulation as an integer linear programming (ILP) problem which is an extension of one of ILP formulations of the TSP. Such a formulation allows solving the TSPPN by using standard ILP optimization software, which provides an exact optimal solution. Obviously, when solving large instances of such complex problems (larger than the one considered in this article), the duration of computations may become unacceptably long if an exact algorithm is used. In such cases, the condition that the algorithm should provide an exact optimal solution can be abandoned in favor of a heuristic optimization algorithm, which can often reduce the computation time with a slight loss of optimality of the approximate solution. This is why we do not claim that the presented model is the best one because of its computational properties, but we just add it to the palette of the solving methods for the considered optimization problem.

Let us introduce some necessary notations. Formally speaking, let $G = (V, E)$ be a complete graph with an n-element vertex or node set V and edge set E. A positive number c_{ij}, $i \neq j$, $i = 1, 2, \ldots, n$, $j = 1, 2, \ldots, n$ called a weight (which can denote the distance or, alternatively, the cost/time of trip between the nodes) is associated with each edge connecting node i with j. As travelling between any pair of nodes generally can occur in either of the two possible directions, c_{ij} does not need to be equal to c_{ji} (e.g., because of different time of trip downhill and uphill). If $c_{ij} = c_{ji}$ for all the edges in the graph, the problem is called the symmetric travelling salesman problem (STSP), and if not, the problem is called the asymmetric travelling salesman problem (ATSP).

The considered generalization assumes that the necessity of visiting each node in the graph exactly once is changed to possibility of visiting each node at most once (or, which is equivalent, the possibility of skipping some nodes). Moreover, penalties imposed for skipping nodes, or in short, penalties on nodes, exist. They are positive numbers p_i—fixed costs specific for each node i, charged if the node is skipped.

Obviously, the possibility of skipping nodes without penalties would lead to a "zero solution". Namely, in case of minimization of the standard TSP objective function formulated as the sum of weights of the edges connecting the nodes visited in the cycle, skipping all the nodes results in making no trip at all with the total weight equal to zero. This is not true, however, if penalties on nodes exist. The definition of the penalties as the costs implies that, for consistence of units, all the weights of the edges in the graph must be expressed as the costs of travelling through the edges and not the distances or trip durations. If a node is skipped, then the cost of travelling to and from that node is zero, but the penalty "attached" to the node is added to the total cost of the trip instead. The objective function is the sum of weights of all the edges connecting the visited nodes and of penalties "attached" to the skipped nodes. If the penalties are relatively low compared with the costs of travelling through the edges, it may be cheaper to skip some, or in extreme cases, even all the nodes instead of visiting them.

A practical application of the considered TSPPN can be straightforward, namely, including in TSP route planning for the total value of penalties resulting from not fulfilling contractual obligations which require visiting all the nodes in the graph. Another example of an application can be delivery outsourcing. If a TSP trip is used to provide deliveries of some goods, then there may exist a possibility of using an external service (e.g., a courier company) for deliveries to some nodes instead of using one's own means of transportation. If a delivery to some node is performed by the external service, then it is no longer necessary to visit that node. The cost of delivery to the node is then the penalty for skipping the node.

The definition presented above is simple and easy to understand. However, as in the case of the standard TSP, it does not suggest any idea of finding an optimal except for a "brute force" approach, i.e., checking the value of the objective function for all the feasible solutions. Namely, it means checking the cost for all the cycles with all the possible combinations of skipped nodes. The previous statement is equivalent to solving by "brute force" a series of the TSPs: with no skipped node, with each node skipped, with all pairs of the nodes skipped and so on. Such an approach is useless, however, when applied to real-world problems of with even a few nodes, only because the "brute force" approach is very inefficient even in the case of a single TSP. The TSPPN formulated in [60] as an optimization model was transformed into a standard TSP problem by adding n fictional nodes and edges. In [61], the TSSPN was solved by using a special algorithm, valid only for the case when the weights of all the edges satisfy the triangle inequality.

This article presents a different approach using integer linear programming (ILP). It is based on one of the so-called flow-based LP formulations of the TSP and named after its creators: the Gavish–Graves formulation [63]. The presented formulation of the TSPPN makes it possible to obtain optimal solutions by using basically any ILP optimization software (ILP solvers). Numerical examples of the TSPPN are also presented, and results of the calculations are discussed.

2. Integer Linear Programming Formulations of the TSP

Integer linear programming formulations of the TSP share some common features. They include binary variables representing making a trip between each pair of nodes; the objective function which is minimized and represents the total distance, cost or duration of a trip through all the nodes; and constraints which are all the sums of the double-indexed binary variables grouped by the number of the first node and the second node equal to one [64]. Those common features are basically identical to a binary assignment problem. The mathematical model of the binary assignment problem is not sufficient to solve each TSP, however, because it can result in obtaining a collection of separate subcycles

or subtours instead of a single Hamiltonian cycle as an optimal solution. That is why some more constraints, so-called subtour elimination constraints (SECs), are necessary to create a linear programming model whose optimal solution is a Hamiltonian cycle. SECs also usually require some extra auxiliary variable, except for the oldest linear programming formulation of the TSP, so called Dantzig–Fulkerson–Johnson (DFJ) formulation [35]. This formulation is the only known linear programming formulation of the TSP which uses binary variables reflecting the usage of edges only and does not require auxiliary variables. However, the main disadvantage of this formulation is the number of constraints, which increases exponentially along with the number of nodes. This is why other formulations were created. They do not need as many constraints as the DFJ formulation does, but they require auxiliary variables instead. A review of such formulations can be found in [64]. They include [35] a sequential formulation by Miller, Tucker and Zemlin [65]; flow-based formulations [63,66–68]; and time-dependent formulations [46,69]. A modification of [65] (not listed) was presented in [70]. In this article, we consider one of linear programming formulations of the TSP—the Gavish–Graves formulation [63]. It is also called a flow-based formulation because auxiliary variables used in SECs correspond to flows of some fictional commodity through the graph.

The Gavish–Graves formulation of the TSP is the following.

Let us consider a graph with n nodes. Satisfying the triangle inequality by the weights of edges is not required. The used notation assumes that the trip through the graph starts and ends at node 1 (the base node). Obviously, it is a purely technical assumption because nodes can be arbitrarily renumbered without affecting optimal solutions.

Variables:

- $x_{ij}, i = 1, \ldots, n;\ j = 1, \ldots, n$—binary: equals 1 if a trip from node i to node j is done and 0 if not;
- $y_{ij}, i = 1, \ldots, n;\ j = 1, \ldots, n$—amount of fictional commodity transported from node i to node j.

Parameters:

- $c_{ij}, i = 1, \ldots, n;\ j = 1, \ldots, n$—cost of a trip from node i to node j.

Objective function—the total cost of a trip through all the visited nodes:

$$\begin{array}{c} c_{11}x_{11} + c_{12}x_{12} + \ldots + c_{1n}x_{1n} + \\ c_{21}x_{21} + c_{22}x_{22} + \ldots + c_{2n}x_{2n} + \\ \vdots \\ c_{n1}x_{n1} + c_{n2}x_{n2} + \ldots + c_{nn}x_{nn} \rightarrow \min \end{array} \quad (1)$$

subject to the constraints

$$\begin{array}{c} x_{11} + x_{12} + \ldots + x_{1n} = 1 \\ x_{21} + x_{22} + \ldots + x_{2n} = 1 \\ \vdots \\ x_{n1} + x_{n2} + \ldots + x_{nn} = 1 \end{array} \quad (2)$$

$$\begin{array}{c} x_{11} + x_{21} + \ldots + x_{n1} = 1 \\ x_{12} + x_{22} + \ldots + x_{n2} = 1 \\ \vdots \\ x_{1n} + x_{2n} + \ldots + x_{nn} = 1 \end{array} \quad (3)$$

$$y_{ij} \geq 0, i = 1, \ldots, n;\ j = 1, \ldots, n \quad (4)$$

$$y_{ij} \leq (n-1)x_{ij},\ i = 1, \ldots, n;\ j = 1, \ldots, n \quad (5)$$

$$\begin{array}{c} y_{11} + y_{21} + \ldots + y_{n1} - (y_{11} + y_{12} + \ldots + y_{1n}) = n - 1 \\ y_{12} + y_{22} + \ldots + y_{n2} - (y_{21} + y_{22} + \ldots + y_{2n}) = -1 \\ y_{13} + y_{23} + \ldots + y_{n3} - (y_{31} + y_{32} + \ldots + y_{3n}) = -1 \\ \vdots \\ y_{1n} + y_{2n} + \ldots + y_{nn} - (y_{n1} + y_{n2} + \ldots + y_{nn}) = -1 \end{array} \quad (6)$$

(2) and (3) are standard TSP constraints which guarantee visiting each node exactly once. (4) (5) and (6) are flow-based subtour elimination constraints.

A short explanation of how the above SECs act is the following. In each node (except the base node), one unit of a fictional commodity is added to the total flow, and finally, $n - 1$ units are "collected" in the base node. Flows leaving the nodes are expressed as negative numbers and flows entering the nodes as positive numbers. Constraints (4), (5) and (6) make all the values of x_{ij} which do not represent Hamiltonian cycles infeasible.

SECs in the Gavish–Graves formulation play an additional role. Namely, they guarantee zero optimal values for the variables x_{kk}, $k = 1, 2, \ldots, n$ corresponding to non-existing "edges" connecting each node to itself, known in graph theory as loops.

A more detailed explanation of subtour elimination in the Gavish–Graves formulation with an example is presented in Appendix A.

The formulation has n^2 binary variables, n^2 continuous variables, $n^2 + 3n$ linear constraints and n^2 nonnegatitvity constraints on continuous variables. Out of the above, n binary variables x_{kk}, n continuous variables y_{kk} and n linear constraints $y_{kk} \leq (n-1)x_{kk}$ can be removed from the model, as they correspond to loops on all the nodes, but removing them may be inconvenient because of a feature of the optimization software (see Appendix B for details).

The above formulation was used as a "base" for an analogical flow-based ILP formulation of the TSPPN.

3. A Flow-Based ILP Formulation of the TSPPN

The TSPPN differs from the TSP in two features: possibility of skipping nodes and existence of penalties for skipping nodes. As skipping nodes in the TSP is a matter of decision of a decision-maker, this means that it is necessary to introduce new variables indicating whether the nodes are skipped or not and new parameters equal to penalties for skipping the nodes. Obviously, those variables are binary. We assume that node 1 (base node) is never skipped. All the variables and parameters of the Gavish–Graves formulation of the TSP remain valid, but new variables and parameters related to nodes are also added. The new variables are:

- z_k, $k = 2, \ldots, n$—binary: 1 if node k is skipped, 0, if not (i.e., if it is visited).

The new parameters are:

- p_{ik}, $k = 2, \ldots, n$—penalty for skipping node k.

Moreover, $c_{11} = 0$ (the reason for this value is explained in Appendix A).

The optimization problem is formulated as follows.

Objective function—the total cost of a trip through all the visited nodes and of the penalties associated with all the skipped nodes:

$$\begin{array}{c} c_{11}x_{11} + c_{12}x_{12} + \ldots + c_{1n}x_{1n} + \\ c_{21}x_{21} + c_{22}x_{22} + \ldots + c_{2n}x_{2n} + \\ \vdots \\ c_{n1}x_{n1} + c_{n2}x_{n2} + \ldots + c_{nn}x_{nn} + \\ p_2z_2 + p_3z_3 + \ldots + p_nz_n \rightarrow \min \end{array} \quad (7)$$

subject to the constraints

$$\begin{array}{c} x_{11} + x_{12} + \ldots + x_{1n} = 1 \\ x_{21} + x_{22} + \ldots + x_{2n} = 1 - z_2 \\ \vdots \\ x_{n1} + x_{n2} + \ldots + x_{nn} = 1 - z_n \end{array} \tag{8}$$

$$\begin{array}{c} x_{11} + x_{21} + \ldots + x_{n1} = 1 \\ x_{12} + x_{22} + \ldots + x_{n2} = 1 - z_2 \\ \vdots \\ x_{1n} + x_{2n} + \ldots + x_{nn} = 1 - z_n \end{array} \tag{9}$$

$$y_{ij} \geq 0,\ i = 1, \ldots, n;\ j = 1, \ldots, n \tag{10}$$

$$y_{ij} \leq (n-1)x_{ij},\ i = 1, \ldots, n;\ j = 1, \ldots, n \tag{11}$$

$$\begin{array}{c} y_{11} + y_{21} + \ldots + y_{n1} - (y_{11} + y_{12} + \ldots + y_{1n}) + z_2 + \ldots + z_n = n - 1 \\ y_{12} + y_{22} + \ldots + y_{n2} - (y_{21} + y_{22} + \ldots + y_{2n}) = z_2 - 1 \\ y_{13} + y_{23} + \ldots + y_{n3} - (y_{31} + y_{32} + \ldots + y_{3n}) = z_3 - 1 \\ \vdots \\ y_{1n} + y_{2n} + \ldots + y_{nn} - (y_{n1} + y_{n2} + \ldots + y_{nn}) = z_n - 1 \end{array} \tag{12}$$

(8) and (9) are reformulated standard TSP constraints (2) and (3), respectively, which guarantee visiting each node at most once (node 1 exactly one). (10), (11) and (12) are flow-based subtour elimination constraints which guarantee that feasible, and in what follows, optimal solutions are single (but, if at least one node is skipped, not necessarily Hamiltonian) cycles. (10) and (11) are identical to (4) and (5), respectively. Constraints (12) are reformulated versions of (6), in the way which excludes the flow of the fictional commodity through skipped nodes.

A more detailed explanation of subtour elimination and skipping the nodes in the Gavish–Graves-based formulation is presented in Appendix A.

If it is needed, some optional extensions of the model of TSSPPN can be easily introduced. They are connected with possible restrictions on the number of skipped nodes.

Optional parameters are:

- $S_{min}, 0 \leq S_{min} \leq n-2$—minimal number of skipped nodes.
- $S_{max}, 1 \leq S_{max} \leq n-1,\ S_{min} \leq S_{max}$—maximal number of skipped nodes.

Optional constraints are:

$$S_{min} \leq z_2 + \ldots + z_n \leq S_{max} \tag{13}$$

Obviously, if not specified explicitly, the default values are

$$S_{min} = 0, S_{max} = n-1. \text{ If } S_{min} = S_{max} = S, \text{ then } z_2 + \ldots + z_n = S. \tag{14}$$

The formulation has $n^2 + n - 1$ binary variables, n^2 continuous variables, $n^2 + 3n$ linear constraints and n^2 nonnegativity constraints on continuous variables. Just like in case of the Gavish–Graves formulation of the TSP, n binary variables x_{kk}, n continuous variables y_{kk} and n linear constraints $y_{kk} \leq (n-1)x_{kk}$ can be removed from the model, as they correspond to loops on all the nodes, but removing them may be inconvenient because of feature of the optimization software (see Appendix B for details).

Introducing binary variables $z_2, z_3, \ldots, z_n$ do not increase the number of constraints because those variables are used in modification of formulas in the existing constraints. The only exception is if there are optional limits of the numbers of skipped nodes. In this case, one (14) or two linear constraints (13) are added.

Introducing the explicit parameters S_{min} and S_{max} (just as a single value S_{min} or S_{max} or $S = S_{min} = S_{max}$; or as a pair of values S_{min}, S_{max}, $S_{min} < S_{max}$) and the constraints related to them (either (13) or (14)) is motivated by adding more flexibility to route planning. More precisely, the reason for the above optional extension of the mathematical model is to enable analysis of scenarios depending on a specific number S of skipped nodes. Such scenarios take into account skipping more nodes than results from the minimization of the total cost of the trip plus the penalties. Even if the total cost is not minimal for a specific number S of skipped nodes, there may exist essential advantages resulting from the trip time and distance reduction. These could be extra cost reductions, such as avoiding the hotel cost for the driver due to the trip-time reduction (as mentioned in [59]), but also reducing the driver's tiredness. Reducing the trip distance decreases not only the fuel cost, but also other exploitation costs resulting from the usage of tires, brake pads, engine oil, etc. As it can be hard to include the abovementioned factors in the objective function expressed in currency units, an alternative for the manager is to study the scenarios depending on the number of skipped nodes. Next, a decision can be made on whether a suboptimal solution (in the sense of the cost minimization) is worth choosing anyway because it can give additional advantages resulting from the trip time and distance reductions. The parameters S_{min}, S_{max}, $S_{min} < S_{max}$ (or only one of them) are essential if there are some technical or organizational restrictions on the number of skipped nodes.

4. Numerical Examples

The problems were solved by using Microsoft Excel 2019 with an optimization add-in, OpenSolver 2.9.3, using the COIN-OR linear optimization engine [71]. The software ran on an HP G6 250 laptop with Intel Core i3-6006U 2 GHz CPU, 8 GB RAM, 1 TB HDD and Windows 10 Home 64 bit. Details of the implementation are presented in Appendix B with the data used to solve the first of the example problems.

The choice of the optimization software was motivated by its availability and price. Obviously, Microsoft Excel is not free, but OpenSolver replaces the built-in Solver at no cost. Moreover, OpenSolver offers not only declared, but also actually much better computational efficiency than the built-in Solver, including no predefined, very-low limits on the numbers of variables and constraints.

The data used in example problems partially came from of a well-known TSPLib library of instances of the TSP. The word "partially" pertains to distance matrices from TSBLib, which were used as cost matrices containing parameters c_{ij}.

The data for the first problem, i.e., the matrix of the edge weights, were downloaded from [72] (it is named br17.atsp). It is an asymmetrical TSP with 17 nodes. The original problem was modified because it contained 36 edges with zero weights. Since zero distances, costs or time usually make no sense in real-world applications, zero-weight edges were replaced with 36 excluded edges, i.e., because it was assumed that connections between 18 pairs of nodes (in both directions, hence 36 edges) just do not exist. Excluding the 36 abovementioned edges was achieved by assigning to them fictional, relatively very large weights (equal to 1000, whereas weights of the existing edges varied from 3 to 74), which provided their absence in the optimal solution. The penalties associated with nodes 2 to 17 were arbitrarily chosen by the authors. Two sets of penalties were tested. They are presented in Table 1. The cost matrix can be found in Figure A4 in Appendix B (a screenshot of the spreadsheet).

The original problem, br17.atsp, from TSPLib, was solved along with a problem with two sets of penalties. Results are collected in Table 2.

Table 1. Two sets of penalties used for solving the problem b17.atsp as a TSPPN.

Node	Penalties Set 1	Penalties Set 2
2	5	7.5
3	6	9
4	4	6
5	5	7.5
6	7	10.5
7	6	9
8	4	6
9	5	7.5
10	5	7.5
11	6	9
12	3	4.5
13	7	10.5
14	4	6
15	6	9
16	6	8
17	4	5
Total	83	122.5

Table 2. Solutions of the original TSP br17.atsp and its extension to TSPPN with two sets of penalties presented as sequences of visited nodes.

	Original TSP	TSSPN Penalty Set 1	TSSPN Penalty Set 2
	2	10	10
	3	12	3
	11	2	11
	14	14	8
	10	11	7
	12	3	4
	13	13	15
	9	1	5
	16		6
	8		9
	6		12
	4		13
	15		14
	5		2
	7		1
	17		
	1		
Skipped nodes	0	9	2
Trip cost	87	24	71
Penalties	0	47	13
Total	87	71	84

The minimal cost of a Hamiltonian cycle for the "standard" TSP br17.atsp is 87. In penalty set 1, the sum of penalties is smaller—83, which would suggest at a first glance that the best solution is to skip all the nodes and pay the penalties. Obviously, in a case where the sum of penalties is smaller than the cost of the Hamiltonian cycle, it is better in terms of cost to skip the nodes. However, as the obtained solution shows, this does not mean that skipping all the nodes is optimal. The optimal solution for penalty set 1 requires skipping nine nodes and visiting the remaining eight in the sequence shown in Table 2. The minimal total cost is 71, which is composed of the total of the weights of the used edges, equal to 24, and the sum of penalties associated with the skipped nodes, equal to 47. The total cost reduction compared with the cheapest Hamiltonian cycle is 16 (18.39%).

Visualizations of the solution of the "standard" TSP br17.atsp and its extension to the TSPPN with penalty set 2 are presented in Figures 1 and 2.

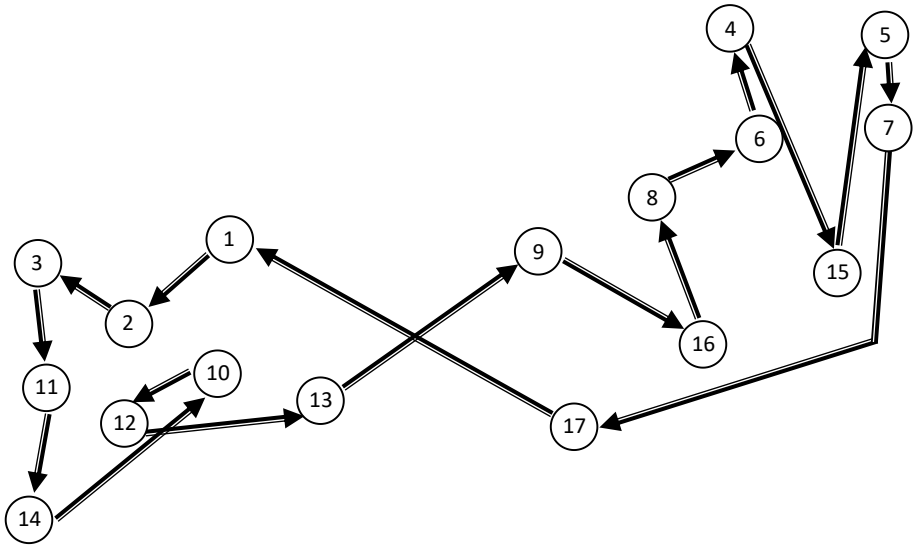

Figure 1. Visualization of the solution of the "standard" TSP br17.atsp presented in Table 2.

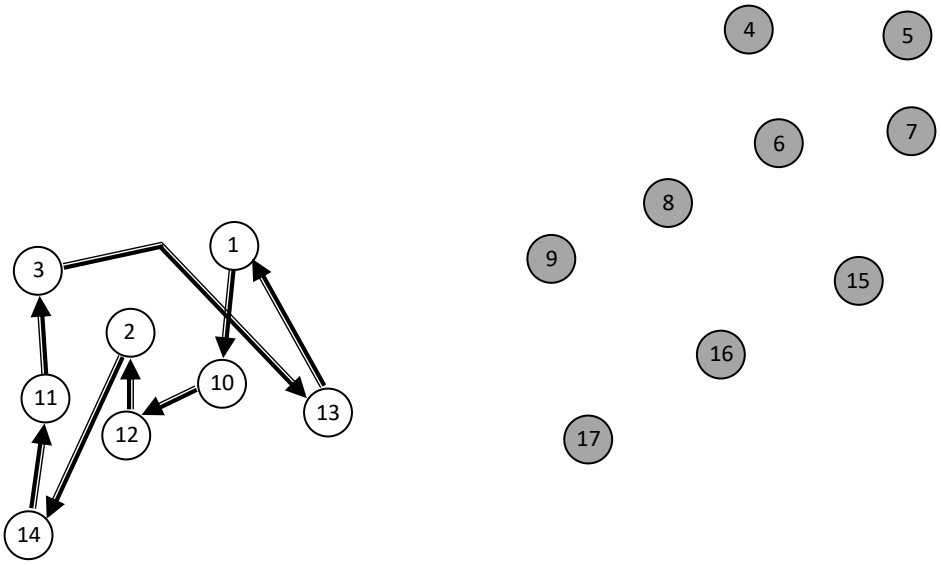

Figure 2. Visualization of the solution of the TSPPN based on br17.atsp with penalty set 1 (presented in Table 2).

Penalty set 2 contains elements which are 50% larger than those of penalty set 29 (with exceptions for nodes 16 and 17). The sum of penalties is now 122.5. One can expect that larger penalties result in larger contributions of the cost of the trip through the network compared with the penalties paid. Since the sum of penalties is larger than the minimal cost of the Hamiltonian cycle, it is not obvious that skipping any nodes will result in a total cost reduction. The optimal solution for penalty set 2 requires skipping 2 nodes and visiting the remaining 15 in the sequence shown in Table 2. The minimal total cost is 84, which is composed of the total of the weights of used edges, equal to 71, and the sum of penalties associated with the skipped nodes, equal to 13. The total cost reduction compared with the cheapest Hamiltonian cycle is three (3.45%).

Table 3 shows how the minimal total cost changes for all the possible pairs of skipped and visited nodes. The calculations were performed for both the penalty sets according to the TSPPN model (7) to (12) with an extra constraint (14), where $S = 0, 1, 2, \ldots, 16$.

Table 3. The minimal total costs of trip and penalties for all combinations of visited and skipped nodes.

Skipped Nodes (S)	Visited Nodes ($n - S$)	Penalty Set 1			Penalty Set 2		
		Trip	Penalties	Total	Trip	Penalties	Total
0	17	87	0	87	87	0	87
1	16	78	6	84	78	8	86
2	15	71	10	81	71	13	84
3	14	66	14	80	66	19	85
4	13	59	20	79	63	23.5	86.5
5	12	54	24	78	54	34	88
6	11	49	29	78	51	38.5	89.5
7	10	38	38	76	46	46	92
8	9	31	42	73	31	61	92
9	8	24	47	71	24	68.5	92.5
10	7	23	52	75	25	73	98
11	6	18	55	73	18	80.5	98.5
12	5	17	60	77	17	88	105
13	4	12	64	76	12	94	106
14	3	11	70	81	11	103	114
15	2	6	76	82	6	112	118
16	1	0	83	83	0	122.5	122.5

Data collected in Table 3 provide some more insights into the TSPPN. As the number of skipped nodes S increases, the total value of penalties imposed on those nodes also increases. On the other hand, the total trip cost generally decreases; however, exceptions may happen. An example of such an exception is skipping 10 nodes compared with skipping 9 nodes for penalty set 2. When 10 nodes are skipped, the trip cost of visiting seven nodes is larger (25) than the trip cost of visiting eight nodes (24) while skipping nine nodes.

However, more valuable conclusions can be drawn from the relationship between the number of skipped nodes and the total cost. If there exists a positive number of skipped nodes, for which the total cost of trip and penalties is smaller than the lowest cost of a Hamiltonian cycle, then along with increasing S, the total cost should decrease to the minimum and then increase. In fact, for penalty set 1, as S changes from 5 to 6, the total cost remains unchanged (78), as is the case with S changing from 7 to 8 for penalty set 2 (cost 92). However, even weak monotonicity with respect to S does not need to hold. For penalty set 1, as S changes from 10 to 11 and from 12 to 13, the total cost decreases from 75 to 73 and from 77 to 76, respectively. Such results may suggest rethinking of the optimality of the model from the point of view of possible applications. Let us return to a practical problem which was the inspiration for creating this article, i.e., optimization of delivery costs as a "mix" of deliveries of goods performed by the company's own transport and outsourcing. The main source of cost reductions was not connected with the fuel cost but with the shorter trip time. The cost reduction consisted in avoiding the cost of the hotel for the driver thanks to decreasing the trip time. However, even without such opportunities,

such as saving the relatively large cost of a hotel stay, there are other reasons for decreasing the trip time. The shorter the planned trip time is, the lesser is the risk of accidents due to the driver's tiredness and the larger is the immunity to delays caused by unexpected disturbances in traffic conditions. Thus, remembering that the considered problem uses fictional data, we can suppose that if enough time is saved thanks to that, the solution for penalty set 1 with the cost 73 for $S = 11$ or even with the cost 76 for $S = 13$ could be better than the strictly cost-effective solution with the cost 71 for $S = 9$. However, there may be no obvious way of recalculating the time-for-money parameters, especially when considering the risk resulting from accidental events disturbing the traffic. The above conclusions are worth developing, but such considerations are outside the scope of this article.

Analysis of how values of penalties affect an optimal solution of the TSPPN is difficult in the general case, as the penalty imposed on each node can be a number independent of $n - 2$ remaining ones. However, some general properties can be noticed if all the penalties have the same value: $p > 0$. In this case, p can be considered as a parameter with respect to which a solution is found. Tests were performed on three problems from TSPLib: the already investigated br17.atsp (17 nodes, asymmetrical [72]), bayg29.tsp (29 nodes, symmetrical [73]) and ftv44.atsp (45 nodes, asymmetrical [74]). TSPPN was solved for data for each of the three problems for various values of p. For br17, $p = 1, 2, \ldots, 10$. For bayg29.tsp and ftv44.atsp, values of p were selected in the following way. First, the largest integer value of p for which all the nodes are skipped and the smallest integer value of p for which no node is skipped were calculated. This was performed by trial-and-error until correct values were found. Next, the TSPPNs were solved with p being multiples of five between the two abovementioned extreme values. The results are presented in Tables 4–6. The columns contain values of p, and for a given p, numbers of skipped nodes, the trip cost, the value of penalties, the total cost and the percentage of saved expenses (as "% saved"). The values in the last column are just percentages referring to differences between the minimal cost of a Hamiltonian cycle (the closed path without skipping the nodes) and the total cost for a given p divided by the minimal cost of a Hamiltonian cycle.

Table 4. The minimal total costs of trip and penalties for various values of p (edge weights from problem br17.atsp).

p	Skipped Nodes	Trip Cost	Penalties	Total Cost	% Saved
1	16	0	16	16	81.61%
2	16	0	32	32	63.22%
3	16	0	48	48	44.83%
4	9	24	36	60	31.03%
5	9	24	45	69	20.69%
6	9	24	54	78	10.34%
7	2	71	14	85	2.30%
8	1	78	8	86	1.15%
9	1	78	9	87	0.00%
10	0	87	0	87	0.00%

It turned out that the same number of nodes in an optimal solution is skipped for various values of p. Obviously, the total cost is increasing function of p until the value of p for which skipping even one node is not optimal is reached. In the problem, ftv44-specific alternative solutions exist—the same minimal total cost 1613 is attained for $p = 74$, where two nodes are skipped, and $p = 75$, where no node is skipped.

Table 5. The minimal total costs of trip and penalties for various values of p (edge weights from problem bayg29.tsp).

p	Skipped Nodes	Trip	Penalties	Total	% Saved
48	28	0	1344	1344	17.39%
49	14	673	686	1359	16.47%
50	13	722	650	1372	15.67%
55	9	934	495	1429	12.17%
60	9	934	540	1473	9.47%
65	5	1183	325	1508	7.31%
70	4	1250	280	1530	5.96%
75	4	1250	300	1550	4.73%
80	4	1250	320	1570	3.50%
85	2	1413	170	1583	2.70%
90	2	1413	180	1593	2.09%
95	2	1413	190	1603	1.48%
100	2	1413	200	1613	0.86%
105	2	1413	210	1623	0.25%
107	2	1413	214	1627	0.00%
108	0	1627	0	1627	0.00%

Table 6. The minimal total costs of trip and penalties for various values of p (edge weights from problem ftv44.atsp).

p	Skipped Nodes	Trip	Penalties	Total	% Saved
23	44	0	1012	1012	37.26%
24	37	167	888	1055	34.59%
25	37	167	925	1092	32.30%
30	24	513	720	1233	23.56%
35	16	781	560	1341	16.86%
40	12	931	480	1411	12.52%
45	8	1102	360	1462	9.36%
50	8	1102	400	1502	6.88%
55	6	1208	330	1538	4.65%
60	5	1265	300	1565	2.98%
65	5	1265	325	1590	1.43%
70	2	1465	140	1605	0.50%
74	2	1465	148	1613	0.00%
75	0	1613	0	1613	0.00%

5. Conclusions

Mathematical optimization problems which result from the necessity of optimizing real-world decision problems can be quite easy to formulate, just like the one considered in this article. Nevertheless, the simplicity of a formulation does not need to mean that a simple way of finding an optimal solution exists, and this applies to the problem under consideration. The problem considered in this article comes from a question posed in [59], which could be expressed as: Is it reasonable from the financial point of view for a wholesale trade company to replace some deliveries to remote customers performed by one's own transport with outsourcing transport services? The attempt how to make such a decision correctly led to a formulation of an extension of the travelling salesman problem with the possibility of skipping some nodes in the network and paying some price for it.

The main result of this article is a formulation of the TSPPN which can be solved, from a purely technical point of view, by any optimization software which solves integer linear programming problems. The computational efficiency of solving a particular problem may obviously depend on the problem's size, the values of the parameters, the hardware configuration and the quality of the software used. However, it may happen that the default

user interface of general purpose optimization software will not be convenient for end users—logistics practitioners—and will require some improvements.

The presented formulation of the TSPPN seems to be promising as a starting point for more sophisticated extensions of the TSP. The research performed in [59] and numerical simulations performed for this article show that similar values of the total cost (the trip cost plus the penalties) may result for different numbers of skipped nodes. However, the more nodes skipped, the shorter the trip time. A shorter trip time is a factor which improves driving safety and provides immunity to unexpected delays. Reducing the number of visited nodes means that the trip distance also decreases. As has already been mentioned, a shorter driving distance affects not only the fuel cost, but also other exploitation costs resulting from the usage of tires, brake pads, engine oil, etc. These results suggest that in practical applications, it may be preferred to outsource more deliveries than what is suggested by simple cost minimization. A possible way to take into account effects of the decreases in the trip time and distance resulting from skipping the nodes is a scenario analysis performed for various possible numbers of skipped nodes by using optional constraints (13) or (14), and possibly in part by an arbitrary decision of the manager. The above goal can also be achieved by reformulation of TSSPN towards implementing time- and distance-related constraints or modifications to the objective function. Other possible extensions of the TSPPN may be introducing predefined sequence(s) of visiting some nodes or considering pickup/delivery specific amounts of some real commodity under capacity restrictions.

Author Contributions: Conceptualization, P.K. and P.M.; methodology, P.K. and P.M.; software, P.K.; validation, G.S., P.B. and P.M.; formal analysis, P.K.; investigation, P.K., G.S. and P.B.; resources, P.K.; data curation, P.K. and G.S.; writing—original draft preparation, P.K.; writing—review and editing, P.K. and P.M.; visualization, P.M.; supervision, P.K.; project administration, G.S.; funding acquisition, G.S. and P.M. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Used data were downloaded from web addresses specified in [72–74].

Conflicts of Interest: The authors declare no conflict of interest.

Appendix A

The appendix contains examples of two feasible solutions: one of the Gavish–Graves formulation of the TSP and the other of its extension to a variant of the one named in this article as TSPPN. The example is used to explain how the Gavish–Graves formulation is adopted to TSPPN works.

Let us consider two feasible solutions of the TSPPN with 7 nodes (one without skipping node, and another with one node skipped). For simplicity of notation, only those parameters c_{ij} which are involved in the feasible solutions will be used with their values expressed explicitly as numbers. From now on, the feasible solution without skipping nodes (sequence of nodes 1,6,7,2,3,4,5,1) will be called "feasible solution 1" (FS1 in short), and the feasible solution with node 6 skipped (sequence of nodes 1,7,2,3,4,5,1) will be called "feasible solution 2" (FS2 in short). As FS1 is in fact also a feasible solution of a standard TSP, all the considerations pertaining to it refer to the formulas of the standard TSP model—i.e., (1) to (6), and not (7) to (12) of the TSPPN. All the numerical data of both feasible solutions are presented in Tables A1 and A2. The two feasible solutions are also visualized in Figure A1.

Table A1. Examples of feasible solutions without skipping nodes and with node 6 skipped, respectively—data for nodes.

Node No.	Penalties p_i	Feasible Solution 1 z_i	Feasible Solution 2 z_i
2	10	0	0
3	8	0	0
4	10	0	0
5	11	0	0
6	9	0	1
7	8	0	0

Table A2. Examples of feasible solutions without skipping nodes and with node 6 skipped, respectively—data for edges.

Edges (*i*,*j*)	Edge Weights c_{ij}	Feasible Solution 1		Feasible Solution 2	
		x_{ij}	y_{ij}	x_{ij}	y_{ij}
(1,6)	13	1	0	0	0
(6,7)	14	1	1	0	0
(1,7)	12	0	0	1	0
(7,2)	10	1	2	1	1
(2,3)	11	1	3	1	2
(3,4)	15	1	4	1	3
(4,5)	12	1	5	1	4
(5,1)	10	1	6	1	5

All the values of x_{ij} and y_{ij} not specified in Table A2 are equal to zero.

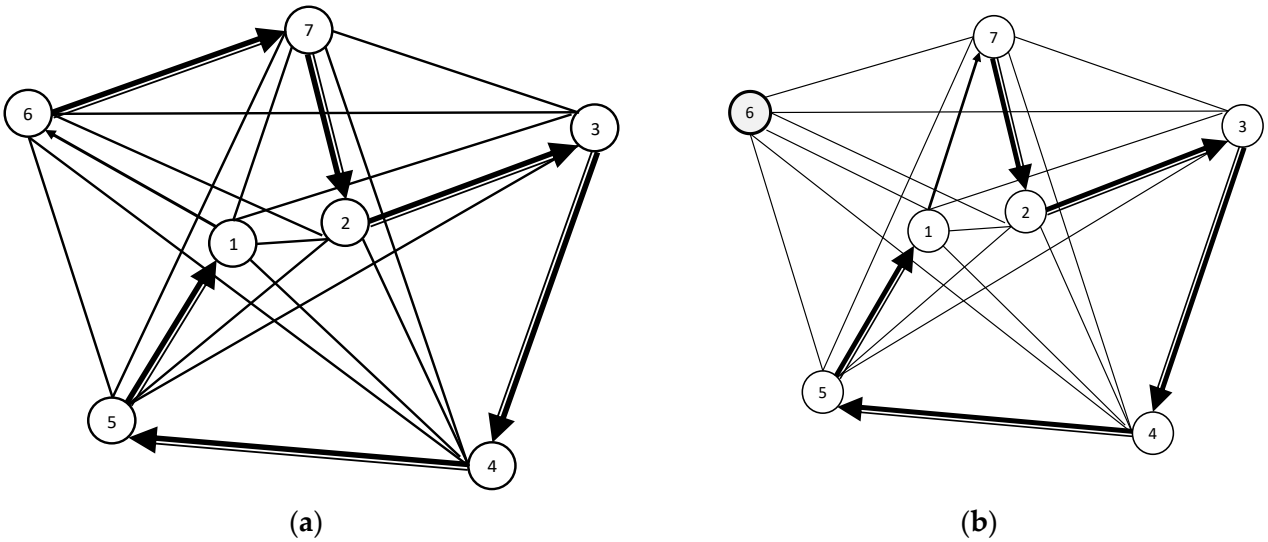

Figure A1. Visualization of two feasible solutions of the TSPPN with 7 nodes. (**a**) A Hamiltonian cycle connecting all 7 nodes—FS1. (**b**) A non-Hamiltonian cycle which does not contain (skips) node 6—FS2.

Circles with numbers inside represent nodes, and lines connecting circles represent edges. Thin arrows represent connections from node 1 to the next node in the cycle (node 6 in FS1 and node 7 in FS2, respectively), for which the flow y_{1j} of a virtual commodity is zero. Thick arrows represent remaining fragments of the cycles, for which the flow y_{ij} of the virtual commodity is positive. The grey background in node 6 in Figure A2b represents a skipped node.

Let us start with analyzing FS1, which is considered just as a feasible solution of a standard TSP in the Gavish–Graves formulation.

Below all the values of FS1 are presented as matrices.

$$[x_{ij}] = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}, \quad [y_{ij}] = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 3 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 4 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 5 & 0 & 0 \\ 6 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 2 & 0 & 0 & 0 & 0 & 0 \end{bmatrix},$$

The value of the objective function (1) is 85. It is a sum of products of the weights of edges and the corresponding x_{ij} binary variables. Alternatively, it can be described as a sum of the weights of used edges:

$$c_{16} + c_{67} + c_{72} + c_{23} + c_{34} + c_{45} + c_{51} = 13 + 14 + 10 + 11 + 15 + 12 + 10 = 85.$$

The binary matrix $[x_{ij}]$ is a numerical representation of the Hamiltonian cycle (1,6,7,2,3,4,5,1). Each element, $x_{ij} = 1$, represents a used (travelled) edge of the network—a connection from node i to node j. Essential features of this matrix are the following.

1. The matrix contains exactly one non-zero element in each row and one in each column.
2. All the elements on the main diagonal, i.e., x_{ii}, are zeroes.
3. Due to item 1, sums of all the elements of each row and each column are equal to one.

Elements of $[x_{ij}]$, and in general, of all the matrices representing Hamiltonian cycles, satisfy the constraints (2) and (3), which are sums (equal to 1) of binary variables grouped by the first (number of the row) and the second (number of the column) index, respectively. The constraints (2) and (3) occur in all the linear programming formulations of the TSP. However, binary matrices satisfying constraints (2) and (3) can represent not only all the Hamiltonian cycles but also "collections" of separate subcycles/subtours and connections from a node to itself ($x_{ii} = 1$), called loops. It is easy to eliminate loops from being a part of an optimal solution. It can be achieved be either by assigning fictional "big M" values to c_{ii} instead of zeroes or by excluding x_{ii} from the model. Eliminating the subtours is much harder. The method considered in this article is known as the Gavish–Graves formulation, and it is based on the simulated flow of a fictional commodity. This formulation eliminates loops, too.

An example of a matrix whose elements represent two separate cycles but still satisfy (2) and (3) is presented. Let us slightly modify input data of FS1 (it is named " x-feasible solution 1", or FS1x for short, for its feasibility with respect to constraints containing x_{ij} variables only). Let the non-zero values of x_{ij} be

$$x_{16} = x_{67} = x_{72} = x_{21} = x_{34} = x_{45} = x_{53} = 1.$$

The elements of the matrix of values of x_{ij}

$$[x_{ij}]^* = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

obviously satisfy (2) and (3). A visualization of FS1x is presented in Figure A2.

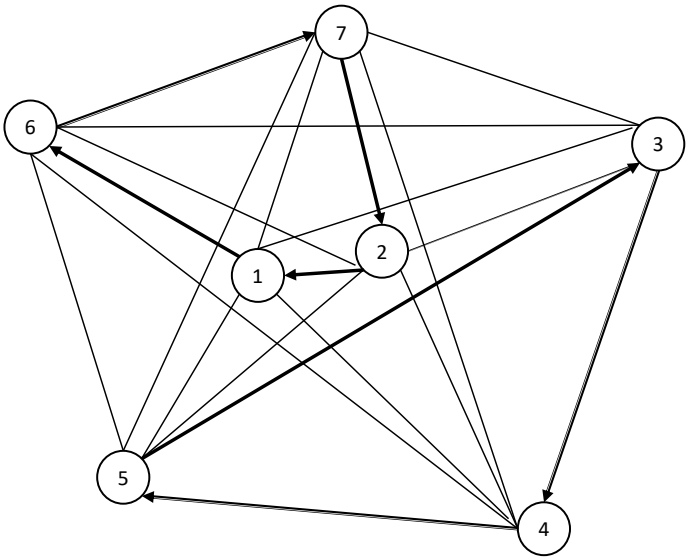

Figure A2. A visualization of FS1x—two separate subtours.

It is obvious that FS1x is not acceptable as a TSP solution because it does not correspond to any Hamiltonian cycle, but to two disjoint subcycles/subtours (1,6,7,2,1) and (3,4,5,3). The main issue associated with non-Hamiltonian matrices satisfying (2) and (3) is that the values of the objective function corresponding to some of them may be lower than those for Hamiltonian cycles. If we explicitly specify two more parameters of the objective function, namely, $c_{21} = 4$ and $c_{53} = 16$, then the value of the objective function for FS1x is 84. Thus, if the problem were solved with constraints (2) and (3) only, FS1x would be preferred to feasible solution 1, for which the value of the objective function is 85.

Now we are going to analyze flow-related constraints (6). In short, they also represent Hamiltonian cycles. For feasible solution 1, they are the following:

$$\begin{aligned}
y_{11} + y_{21} + \ldots + y_{71} - (y_{11} + y_{12} + \ldots + y_{17}) &= 7 - 1 = 6 \\
y_{12} + y_{22} + \ldots + y_{72} - (y_{21} + y_{22} + \ldots + y_{27}) &= -1 \\
y_{13} + y_{23} + \ldots + y_{73} - (y_{31} + y_{32} + \ldots + y_{37}) &= -1 \\
y_{14} + y_{24} + \ldots + y_{74} - (y_{41} + y_{42} + \ldots + y_{47}) &= -1 \\
y_{15} + y_{25} + \ldots + y_{75} - (y_{51} + y_{52} + \ldots + y_{57}) &= -1 \\
y_{16} + y_{26} + \ldots + y_{76} - (y_{61} + y_{62} + \ldots + y_{67}) &= -1 \\
y_{17} + y_{27} + \ldots + y_{77} - (y_{71} + y_{72} + \ldots + y_{77}) &= -1
\end{aligned} \tag{A1}$$

Let us list the values of y_{ij} specified in the 4th column of Table A2 (i.e., corresponding to edges with $x_{ij} = 1$) in accordance with the sequence of nodes in FS1:

$$y_{16} = 0,\ y_{67} = 1,\ y_{72} = 2,\ y_{23} = 3,\ y_{34} = 4, y_{45} = 5,\ y_{51} = 6. \tag{A2}$$

Equation (A1) shows balance constraints for the flow of some fictional commodity. Each node except node 1 is a source with a capacity of one unit of the commodity. Outflows from nodes are denoted by negative values and inflows to nodes by positive values. If we replace y_{ij} in (A1) by the values (A2) (where all other $y_{ij} = 0$), we have

$$\begin{aligned}
y_{51} - y_{16} &= 6 - 0 = 6 \\
y_{72} - y_{23} &= 2 - 3 = -1 \\
y_{23} - y_{34} &= 3 - 4 = -1 \\
y_{34} - y_{45} &= 4 - 5 = -1 \\
y_{45} - y_{51} &= 5 - 6 = -1 \\
y_{16} - y_{67} &= 0 - 1 = -1 \\
y_{67} - y_{72} &= 1 - 2 = -1.
\end{aligned}$$

The meaning of the above equations is the following.

- No unit of a fictional commodity flows to node 6. One unit of a fictional commodity flows from node 6 to node 7. The difference between inflow and outflow for node 6 is -1.
- Node 7 is also a source for another unit of commodity. Thus, 2 units (one which flows from node 6 and one for which node 7 is a source) flowed from node 7 to node 2. The difference between inflow and outflow for node 7 was -1.
- Node 2 is another source of one unit of the commodity. Thus, a total of three units flow from node 2 to node 3. The difference between inflow and outflow for node 2 is -1.
- Analogical operations were performed for nodes 3 and 4.
- Finally, 5 units flow from node 4 to node 5. Node 5 is another source of one unit of the commodity. Thus, 6 units flow from node 5 to node 1. The difference between inflow and outflow for node 5 is -1.
- Six units flow from node 5 to node 1. No commodity flows from node 1, but for consistency of formulas, we can express explicitly $y_{16} = 0$. The difference between inflow and outflow for node 1 is 6 (which is $n - 1$ where $n = 7$).

The sequence of flows considered above (0,1.2, ... 6) (in general case, any sequence of values of flows 0,1.2, ... , $n - 1$ starting with outflow equal to 1 from any node other than 1, and ending with inflow equal to $n - 1$ to node 1) represents a Hamiltonian cycle of corresponding nodes and satisfies (6).

The subtour elimination is achieved by simultaneous usage of (2), (3) and (6), "joined" by (5) and completed by (4)—the flow nonnegativity constraints. By (2) and (3), exactly n out of n^2 x_{ij} variables are equal to one. By (5) and (6), at most n values of y_{ij} can be positive (and not larger than $n - 1$), and the remaining y_{ij} are equal to zero.

In the example, FS1 is

$$x_{16} = x_{67} = x_{72} = x_{23} = x_{34} = x_{45} = x_{51} = 1, \text{ all other } x_{ij} = 0$$

Then, the inequalities $y_{ij} \geq 0$ and $y_{ij} \leq (n-1)x_{ij}$ (in this example $y_{ij} \leq 6x_{ij}$) imply that at most 7 values of y_{ij} (with the same indices as $x_{ij} = 1$ listed above) can be positive and not larger than 6:

$$0 \leq y_{16} = y_{67} = y_{72} = y_{23} = y_{34} = y_{45} = y_{51} \leq 6.$$

The constraints (6) "force" feasible values of y_{ij} for corresponding $x_{ij} = 1$. Obviously, $y_{51} \leq 6$. Let us assume that $y_{51} = 6$. Then, by (6)

$$\begin{aligned} y_{45} - y_{51} = -1 &\Longrightarrow y_{45} = 5 \\ y_{34} - y_{45} = -1 &\Longrightarrow y_{34} = 4 \\ y_{23} - y_{34} = -1 &\Longrightarrow y_{23} = 3 \\ y_{72} - y_{23} = -1 &\Longrightarrow y_{72} = 2 \\ y_{67} - y_{72} = -1 &\Longrightarrow y_{67} = 1 \\ y_{16} - y_{67} = -1 &\Longrightarrow y_{16} = 0 \end{aligned}$$

Thus, the assumption $y_{51} = 6$ results in calculating feasible values of y_{ij} identical to those in (16). Moreover, it turns out that there can be 6 (i.e., $n - 1$) positive feasible values of y_{ij} only. However, by (5) it is also possible that $y_{51} < 6$. Is inflow to node 1 of less than 6 (in general, less than $n - 1$) also feasible? Let, for example, $y_{51} = 5.5$. Then, by (6)

$$\begin{aligned} y_{45} - y_{51} = -1 &\Longrightarrow y_{45} = 4.5 \\ y_{34} - y_{45} = -1 &\Longrightarrow y_{34} = 3.5 \\ y_{23} - y_{34} = -1 &\Longrightarrow y_{23} = 2.5 \\ y_{72} - y_{23} = -1 &\Longrightarrow y_{72} = 1.5 \\ y_{67} - y_{72} = -1 &\Longrightarrow y_{67} = 0.5 \\ y_{16} - y_{67} = -1 &\Longrightarrow y_{16} = -0.5. \end{aligned}$$

Obviously, because $y_{16} \geq 0$, $y_{51} = 5.5$ (and any $0 \leq y_{51} < 6$) cannot be feasible. Finally, it can be concluded that constraints (4), (5) and (6) calculated for a given binary representation $x_{ij} = 1$ of a Hamiltonian cycle are equivalent to the numbers $0, 1, \ldots, n-1$, which are the values of y_{ij} with the same indices as x_{ij}, $x_{ij} = 1$.

What will be the values of y_{ij} if $x_{ij} = 1$ are not a binary representation of a Hamiltonian cycle (they just satisfy (2) and (3)). Let us try to calculate the values of y_{ij} for FS1x. More precisely, for

$$x_{16} = x_{67} = x_{72} = x_{21} = x_{34} = x_{45} = x_{53} = 1$$

we need to calculate the following values of y_{ij} satisfying (6)

$$0 \leq y_{16} = y_{67} = y_{72} = y_{21} = y_{34} = y_{45} = y_{53} \leq 6.$$

Let us start with $y_{21} = 6$. Then, by (6)

$$\begin{aligned} y_{72} - y_{21} = -1 &\Longrightarrow y_{72} = 5 \\ y_{67} - y_{72} = -1 &\Longrightarrow y_{62} = 4 \\ y_{16} - y_{67} = -1 &\Longrightarrow y_{16} = 3. \end{aligned}$$

However, if $y_{21} = 6$ and $y_{16} = 3$, then the constraint for the flow balance for node 1, i.e., $y_{21} - y_{16} = 6$, cannot be satisfied. This example shows that if there is a subcycle/subtour represented by some $x_{ij} = 1$, then there are no values of y_{ij}, which could be feasible with respect to (4), (5) and (6).

It is worth mentioning that constraints (4), (5) and (6) prevent loops from occurring. A loop for node k, $k = 1, 2, \ldots, n$ is mathematically expressed by $x_{kk} = 1$. By (2) and (3), all the variables x_{ij}, where $i \neq j$ and either $i = k$ or $j = k$ (exactly one index is k), must be then equal to zero. By (4) and (5), all the variables y_{ij} with exactly one index equal to k must be equal to zero. Hence, out of all y_{ij}, where at least one index is k, y_{kk} can be positive only. The corresponding constraint for flow for node k is then

$$y_{11} - y_{11} = n - 1 \text{ for } k = 1$$

or

$$y_{kk} - y_{kk} = -1 \text{ for } k = 2, 3, \ldots, n$$

Both variants of the constraint are obviously contradictory, so occurring as $x_{kk} = 1$ in a feasible, and, in what follows, possibly an optimal solution is prevented no matter what the values of c_{kk} are. This is why there is no necessity of eliminating possible $x_{kk} = 1$ feasible solutions (resulting from $c_{kk} = 0$) by replacing $c_{kk} = 0$ with $c_{kk} = M$, where M is "big M", much larger than all the c_{ij}, where $i \neq j$ (parameters of the objective function (1) corresponding to "true" connections between pairs of different nodes).

The next step is to analyze an extension of the Gavish–Graves formulation of the TSP with regard to handling node skipping at the cost of paying penalty per each skipped node. This extension in the general form is expressed by (7) to (12) and—if applicable—optional constraints (13) or (14). It is assumed that node 1 cannot be skipped. A binary variable $z_k, k = 2, 3, \ldots, n$ is associated with each node. A value of 1 means that the node is skipped, whereas the value 0 means that the node is visited. Variables z_k play two roles. The first one is including the penalty cost for node i in the objective function. The second one is "eliminating" all the variables x_{ij} (and, what follows, also y_{ij}), where $i = k$ or $j = k$ (at least one index of two is k), by forcing them to be equal to zero. It must be performed because if node k is skipped, the mathematical model must reflect the fact that no connections to and from that node exist. Obviously, if at least one node is skipped, a cycle containing all the visited nodes is no longer a Hamiltonian cycle.

Including the penalty costs for skipping nodes in the objective function is obvious—it is performed by adding a sum of products of penalties and variables z_k, $k = 2, 3, \ldots, n$ in (7).

Forcing all the variables x_{ij} and y_{ij} with at least one index k to be equal to zero for variables x_{ij} is performed by constraints (8) and (9) (which are modified versions of (2) and (3), respectively) and for variables y_{ij} by (12)—a modified version of (6).

Below all the feasible values in FS2 are presented as matrices.

$$[x_{ij}] = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}, \quad [y_{ij}] = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 3 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 4 & 0 & 0 \\ 5 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Skipping node 6 means that in both matrices, row 6 and column 6 contain zeroes only.

The value of the objective function (7) is 79. It is a sum of products of the weights of edges and the corresponding x_{ij} binary variables added to a sum of products of the penalties and the corresponding z_k binary variables. Alternatively, it can be described as a sum of the weights of used edges added to a sum of penalties on skipped nodes:

$$c_{17} + c_{72} + c_{23} + c_{34} + c_{45} + c_{51} + p_6 = 12 + 10 + 11 + 15 + 12 + 10 + 9 = 70 + 9 = 79.$$

It is less than the value of the objective function (1) for FS1, so skipping node 6 is preferable to visiting it (and it would be also for other values of p_6 as long as $p_6 < 15$).

In feasible solution 2, node 6 is skipped; hence, $z_6 = 1$. As node 6 will not be included in any cycle connecting all other nodes, then x_{i6}, x_{6j}, y_{i6}, y_{6j} for $i = 1, 2, \ldots, n$, $j = 1, 2, \ldots, n$ all must be zeroes.

For variables x_{i6}, x_{6j}, we have

$$x_{61} + x_{62} + x_{63} + x_{64} + x_{65} + x_{66} = 1 - z_6 = 0$$
$$x_{16} + x_{26} + x_{36} + x_{46} + x_{56} + x_{66} = 1 - z_6 = 0$$

which (since x_{i6}, x_{6j} are binary variables) results in $x_{i6} = x_{6j} = 0$.

Whenever node k is skipped, there cannot exist the flow of the fictional commodity via that node. Obviously, constraints involving flows of the fictional commodity (10) and (11) (identical to (4) and (5), respectively) and (12) (a modified version of (6)) are still necessary because they allow only y_{ij} corresponding to $x_{ij} = 1$ to represent a flow through a single cycle connecting 6 visited (not skipped) nodes. In the case of FS2, if $x_{i6} = x_{6j} = 0$, then by (10) and (11).

$$0 \le y_{i6} \le 6x_{i6},\ 0 \le\ y_{6j}\ \le 6x_{6j} \text{ for } i = 1, 2, \ldots, n,\ j = 1, 2, \ldots, n$$

and hence $y_{i6} = y_{6j} = 0$. Next,

$$y_{61} + y_{62} + y_{63} + y_{64} + y_{65} + y_{66} - (y_{61} + y_{62} + y_{63} + y_{64} + y_{65} + y_{66}) = 0.$$

As such a constraint in the Gavish–Graves formulation of the TSP has a right-hand side equal to -1, so it must be adjusted—in the case of no flow ($z_6 = 1$)—such that the RHS of the constraint must be zero. This is performed in the following way:

$$y_{61} + y_{62} + y_{63} + y_{64} + y_{65} + y_{66} - (y_{61} + y_{62} + y_{63} + y_{64} + y_{65} + y_{66}) = z_6 - 1$$

Analogical adjustments are performed for flow constraints for all the nodes except node 1.

Node 1 is a special case. As it is never skipped, it cannot be "eliminated" from any feasible solution. However, the flow balance constraint for node 1 must be adjusted to the value of the flow depending on the number of visited/skipped nodes. This constraint in

the Gavish–Graves formulation of the TSP (6) reflects the fact that inflow of the fictional commodity to node 1 is equal to $n-1$. The total value of inflow to node 1 is the sum of units of the fictional commodity added at each node $2, 3, \ldots, n-1$. However, if a node is skipped, then obviously it is not a source of one unit of fictional commodity and does not contribute to the total flow. In short, each skipped node decreases the inflow to node 1 by one unit. This is why the value of that inflow must be "compensated" by the number of skipped nodes, which is simply the sum of $z_2, z_3, \ldots, z_n$.

The constraint

$$y_{11} + y_{21} + \ldots + y_{n1} - (y_{11} + y_{12} + \ldots + y_{1n}) + z_2 + \ldots + z_n = n - 1$$

for FS2 takes the form

$$y_{51} + z_6 = 5 + 1 = 6 = 7 - 1.$$

An extreme case of skipping $n-1$ nodes (all but node 1) additionally requires one more adjustment—however, not in the constraints but in the objective function. In this case, node 1 is the only visited node, which means that among all the x_{ij} variables, only $x_{11} = 1$ (and, obviously $z_2 = z_3 = \ldots = z_n = 1$). However, such a value of x_{11} means that there is a loop from node 1 to itself. Whereas loops must be eliminated from set of feasible solutions, because they would "break" a single cycle, the case of $n-1$ skipped nodes is different. This means that the cycle along which "the salesman" would travel reduces to "no trip" (no leaving node 1 at all), which is just a loop. Subtour elimination constraints (10), (11) and (12) do not work in this case, and the loop from node 1 to itself remains in the feasible solution. This fact does not affect the correctness of the feasible solution. The only requirement is to set $c_{11} = 0$. Then, the total cost expressed by the objective function is $c_{11}x_{11} + p_2z_2 + p_3z_3 + \ldots + p_nz_n = p_2 + p_3 + \ldots + p_n$; i.e., c_{11} does not affect it. Except this case, at was shown earlier, the parameters c_{kk} can take any values, because the constraints force variables x_{kk} to be zeroes.

Appendix B

The appendix contains details of an implementation of the example problem in Microsoft Excel 2019 with OpenSolver 2.9.3.

The following formatting rules were applied:

- default formatting—values of the parameters and text descriptions;
- bold centered—indices of the nodes;
- italic—values of the variables (bold italic—positive values);
- grey background—formulas.

The cell references pertaining to the particular parts of the considered mathematical model are the following.

Variables:

- B22:R38— x_{ij}, $i = 1, 2, \ldots, 17$; $j = 1, 2, \ldots, 17$.
- B45:R61— y_{ij}, $i = 1, 2, \ldots, 17$; $j = 1, 2, \ldots, 17$,
- V3:V18— z_i, $i = 2, 3, \ldots, 17$.

Parameters:

- B2:R18— c_{ij}, $i = 1, 2 \ldots, 17$; $j = 1, 2, \ldots, 17$.
- W2:W18— p_i, $i = 2, 3, \ldots, 17$.
- B40, T22—1 (RHS's for the constraints $x_{11} + x_{12} + \ldots + x_{1,17} = 1$ and $x_{11} + x_{21} + \ldots + x_{n,17} = 17$, respectively).
- C41— $n-1$ (16 in this model).

Formulas:

- S22:S38—sums of x_{ij} variables by index j ($x_{i1} + x_{i2} + \ldots + x_{1,17}$, $i = 1, 2, \ldots, 17$); in S22: =SUM(B22:R22), S22 copied to S23:S38.

- B39:R39—sums of x_{ij} variables by index i ($x_{1j} + x_{2j} + \ldots + x_{17,j}$, $j = 1, 2, \ldots, 17$); in B39: =SUM(B22:B38), B39 copied to C39:R39;
- T23:T38—differences $1 - z_i$ (RHS's for sums of x_{ij} variables by index j, $i = 2, 3, \ldots, 17$); in T23: =1-V3, T23 copied to T23:T38;
- C40:R40—differences $1 - z_j$, (RHS's for sums of x_{ij} variables by index i, $j = 2, 3, \ldots, 17$); in C40: =1- INDEX($V2:$V18,C21,1), C40 copied to D40:R40 (the function INDEX provides references to consecutive cells of the one-column range V2:V18 when copying of C40 is performed horizontally).
- S40—trip cost (the first part of the objective function): =SUMPRODUCT(B2:R18,B22:R38).
- U40—sum of the values of penalties (the second part of the objective function): =SUMPRODUCT(W2:W18,V2:V18).
- W40—total cost (the objective function) = S40 + U40.
- B62:R62—sums of y_{ij} variables by index j ($y_{i1} + y_{i2} + \ldots + y_{I,17}$, $i = 1, 2, \ldots, 17$) —inflows of fictional commodity; in B62 = SUM(B45:B61), B62 copied to C62:R62;
- S45:S61—sums of y_{ij} variables by index i ($y_{1j} + y_{2j} + \ldots + y_{17,j}$, $j = 1, 2, \ldots, 17$) —outflows of fictional commodity; in S45 = SUM(B45:R45), S45 copied to S46:S61.
- U45:U61—inflows of fictional commodity—a transpose of B62:R62, introduced for technical reasons; in U45 = INDEX(B$62:R$62,1,A45), U45 copied to U46:U61.
- V45—the formula for the LHS of the balance formula of fictional commodity flow for node 1 ($y_{11} + y_{21} + \ldots + y_{17,1} - (y_{11} + y_{12} + \ldots + y_{1,17n}) + z_2 + \ldots + z_{17}$): =U45-S45 + SUM(V2:V18).
- V46:V61—differences in the inflow and the outflow for each node, i.e., LHSs of the balance formulas of fictional commodity flows for nodes 2,3, ... 17: in V46: =U46-S46. V46 copied to V47:V61.
- W45—RHS of the balance formula of fictional commodity flow for node 1 ($n - 1 = 16$): =C42
- W46:W61—differences $z_i - 1$ (RHS's of the balance formulas of fictional commodity flows for nodes 2,3, ... 17); in W46: =V3-1, W46 copied to W47:W61.
- B66:B82—products $(n - 1)x_{ij}$, i.e., RHS's of the formulas limiting fictional commodity flows between pairs of nodes to the values equal at most $n - 1$; in B66: =B22*C42, B66 copied to B66:B82.

Auxiliary formulas introduced for better readability of the solution (not a part of the mathematical model).

- V22:V38—indices of nodes which are connected with nodes specified in U22:U38 (considered pairwise, e.g., U22 with V22 and U23 with V23; if a node is skipped then there is no connection, and 0 is displayed instead); in V22: =ROUND(SUMPRODUCT(B$21:R$21,B22:R22),0), V22 is copied to V23:V38.
- W22:W38—sequence of nodes in the cycle starting from the node visited after leaving node 1 and ending in node 1 (if not all nodes are visited, then the N/A errors display in the lower part of the range), in W22: =V22, in W23: =VLOOKUP(W22,U$23:V$38,2), W23 copied to W24:W38.

As optimal values of the variables in B22:R38 may slightly differ from zero or one due to rounding errors during calculations, the ROUND function is used to provide integer results. Integer values in W22:W38 are necessary to obtain correct results returned by the function VLOOKUP, which calculates the sequence of visited nodes.

The exact layout for data entered into a spreadsheet are presented in Figures A3–A6. Rows 19, 41 and 62 are not shown because they were empty. The settings entered into OpenSolver's model window are presented in Figure A7.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W
1	costs	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**			node#	skipped?	penalties
2	**1**	0	3	5	48	48	8	8	5	5	3	3	1000	3	5	8	8	5			**1**	*0*	
3	**2**	3	0	3	48	48	8	8	5	5	1000	1000	3	1000	3	8	8	5			**2**	*0*	5
4	**3**	5	3	0	72	72	48	48	24	24	3	3	5	3	1000	48	48	24			**3**	*0*	6
5	**4**	48	48	74	0	1000	6	6	12	12	48	48	48	48	74	6	6	12			**4**	***1***	4
6	**5**	48	48	74	1000	0	6	6	12	12	48	48	48	48	74	6	6	12			**5**	***1***	5
7	**6**	8	8	50	6	6	0	1000	8	8	8	8	8	8	50	1000	1000	8			**6**	***1***	7
8	**7**	8	8	50	6	6	1000	0	8	8	8	8	8	8	50	1000	1000	8			**7**	***1***	6
9	**8**	5	5	26	12	12	8	8	0	1000	5	5	5	5	26	8	8	1000			**8**	***1***	4
10	**9**	5	5	26	12	12	8	8	1000	0	5	5	5	5	26	8	8	1000			**9**	***1***	5
11	**10**	3	1000	3	48	48	8	8	5	5	0	1000	3	1000	3	8	8	5			**10**	*0*	5
12	**11**	3	1000	3	48	48	8	8	5	5	1000	0	3	1000	3	8	8	5			**11**	*0*	6
13	**12**	1000	3	5	48	48	8	8	5	5	3	3	0	3	5	8	8	5			**12**	*0*	3
14	**13**	3	1000	3	48	48	8	8	5	5	1000	1000	3	0	3	8	8	5			**13**	*0*	7
15	**14**	5	3	1000	72	72	48	48	24	24	3	3	5	3	0	48	48	24			**14**	*0*	4
16	**15**	8	8	50	6	6	1000	1000	8	8	8	8	8	8	50	0	1000	8			**15**	***1***	6
17	**16**	8	8	50	6	6	1000	1000	8	8	8	8	8	8	50	1000	0	8			**16**	***1***	6
18	**17**	5	5	26	12	12	8	8	1000	1000	5	5	5	5	26	8	8	0			**17**	***1***	4

Figure A3. Data entered into the spreadsheet—parameters c_{ij}, variables z_i, parameters p_i.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W
20																						optimal cycle	
21	x_{ij}	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	sums by j	$1-z_j$	from node	to node	sequence
22	**1**	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	***1***	*0*	*0*	*0*	*0*	1	1	**1**	13	13
23	**2**	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	***1***	*0*	*0*	*0*	1	1	**2**	14	3
24	**3**	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	***1***	*0*	*0*	*0*	*0*	*0*	*0*	1	1	**3**	11	11
25	**4**	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	0	0	**4**	0	12
26	**5**	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	0	0	**5**	0	2
27	**6**	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	0	0	**6**	0	14
28	**7**	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	0	0	**7**	0	10
29	**8**	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	0	0	**8**	0	1
30	**9**	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	0	0	**9**	0	#N/D
31	**10**	***1***	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	1	1	**10**	1	#N/D
32	**11**	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	***1***	*0*	*0*	*0*	*0*	*0*	1	1	**11**	12	#N/D
33	**12**	*0*	***1***	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	1	1	**12**	2	#N/D
34	**13**	*0*	*0*	***1***	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	1	1	**13**	3	#N/D
35	**14**	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	***1***	*0*	*0*	*0*	*0*	*0*	*0*	*0*	1	1	**14**	10	#N/D
36	**15**	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	0	0	**15**	0	#N/D
37	**16**	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	0	0	**16**	0	#N/D
38	**17**	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	0	0	**17**	0	#N/D
39	sums by i	1	1	1	0	0	0	0	0	0	1	1	1	1	1	0	0	0	trip cost		total penalties		total cost
40	$1-z_i$	1	1	1	0	0	0	0	0	0	1	1	1	1	1	0	0	0	24		47		71

Figure A4. Data entered in the spreadsheet—variables x_{ij}, formulas for: sums of x_{ij} by i and j, differences $1 - z_i$ and $1 - z_j$, optimal sequence of nodes, trip cost, penalty cost and total cost.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W
42		n-1	16																				
43		fictional commodity flows																					balance
44		**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	outflows		inflows	differences	values
45	**1**	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	0		7	16	16
46	**2**	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	**5**	*0*	*0*	*0*	5		4	-1	-1
47	**3**	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	**2**	*0*	*0*	*0*	*0*	*0*	*0*	2		1	-1	-1
48	**4**	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	0		0	0	0
49	**5**	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	0		0	0	0
50	**6**	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	0		0	0	0
51	**7**	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	0		0	0	0
52	**8**	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	0		0	0	0
53	**9**	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	0		0	0	0
54	**10**	**7**	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	7		6	-1	-1
55	**11**	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	**3**	*0*	*0*	*0*	*0*	*0*	3		2	-1	-1
56	**12**	*0*	**4**	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	4		3	-1	-1
57	**13**	*0*	*0*	**1**	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	1		0	-1	-1
58	**14**	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	**6**	*0*	*0*	*0*	*0*	*0*	*0*	*0*	6		5	-1	-1
59	**15**	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	0		0	0	0
60	**16**	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	0		0	0	0
61	**17**	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	0		0	0	0
62	inflows	7	4	1	0	0	0	0	0	0	6	2	3	0	5	0	0	0					

Figure A5. Data entered in the spreadsheet—variables y_{ij}, formulas for: sums of y_{ij} by i and j, transposed sums of y_{ij} by index i, differences in the above sums, parameters $n - 1$ and $z_i - 1$.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
64		LHS's of constraints for fictional flows																	
65		**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	
66	**1**	0	0	0	0	0	0	0	0	0	0	0	0	15	0	0	0	0	
67	**2**	0	0	0	0	0	0	0	0	0	0	0	0	0	15	0	0	0	
68	**3**	0	0	0	0	0	0	0	0	0	0	15	0	0	0	0	0	0	
69	**4**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
70	**5**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
71	**6**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
72	**7**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
73	**8**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
74	**9**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
75	**10**	15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
76	**11**	0	0	0	0	0	0	0	0	0	0	0	15	0	0	0	0	0	
77	**12**	0	15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
78	**13**	0	0	15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
79	**14**	0	0	0	0	0	0	0	0	0	15	0	0	0	0	0	0	0	
80	**15**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
81	**16**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
82	**17**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

Figure A6. Data entered into the spreadsheet—formulas for products $(n-1)x_{ij}$.

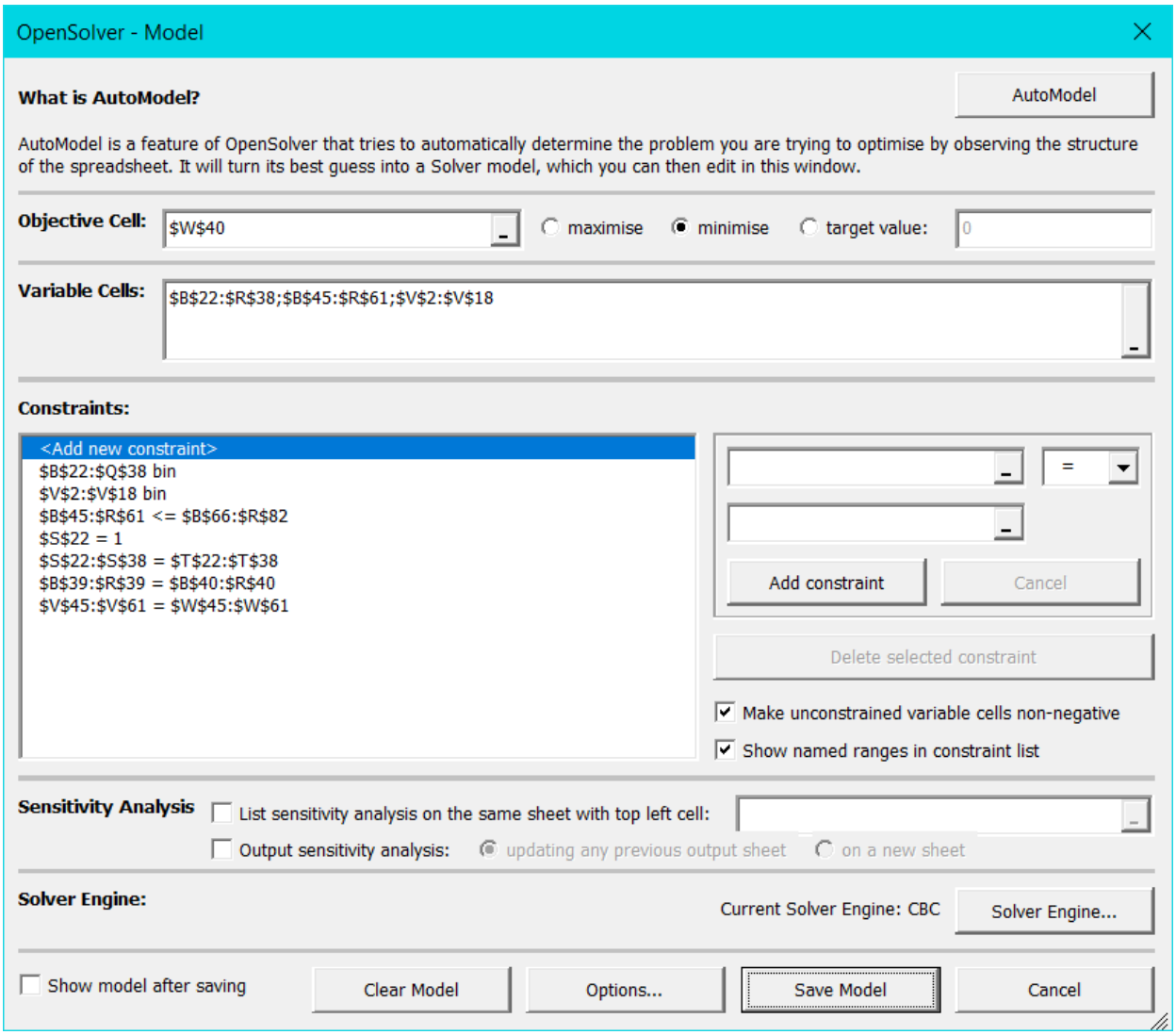

Figure A7. Settings entered into OpenSolver—model window.

References

1. Borucka, A. Logistic regression in modeling and assessment of transport services. *Open Eng.* **2020**, *10*, 26–34. [CrossRef]
2. Lund, S.; Manyika, J.; Woetzel, J.; Bughin, J.; Krishnan, M. Globalization in Transition: The Future of Trade and Value Chains. 2019. Available online: http://dln.jaipuria.ac.in:8080/jspui/bitstream/123456789/10848/1/MGI-Globalization%20in%20transition-The-future-of-trade-and-value-chains-Full-report.pdf (accessed on 3 November 2022).
3. Kot, S.; Haque, A.U.; Baloch, A. Supply Chain Management in Smes: Global Perspective. *Montenegrin J. Econ.* **2020**, *16*, 87–104. [CrossRef]
4. Chen, C.J. Developing a model for supply chain agility and innovativeness to enhance firms' competitive advantage. *Manag. Decis.* **2018**, *57*, 1511–1534. [CrossRef]
5. Gligor, D.; Feizabadi, J.; Russo, I.; Maloni, M.J.; Goldsby, T.J. The triple—A supply chain and strategic resources: Developing competitive advantage. *Int. J. Phys. Distrib. Logist. Manag.* **2020**, *50*, 159–190. [CrossRef]
6. Sirilertsuwan, P.; Thomassey, S.; Zeng, X. A Strategic Location Decision-Making Approach for Multi-Tier Supply Chain Sustainability. *Sustainability* **2020**, *12*, 8340. [CrossRef]
7. Zhou, J.; Wu, Y.; Tao, Y.; Gao, J.; Zhong, Z.; Xu, C. Geographic information big data-driven two-stage optimization model for location decision of hydrogen refueling stations: An empirical study in China. *Energy* **2021**, *225*, 120330. [CrossRef]
8. Saif-Eddine, A.S.; El-Beheiry, M.M.; El-Kharbotly, A.K. An improved genetic algorithm for optimizing total supply chain cost in inventory location routing problem. *Ain Shams Eng. J.* **2019**, *10*, 63–76. [CrossRef]
9. Camisón-Haba, S.; Clemente, J.A. A global model for the estimation of transport costs. *Econ. Res.-Ekon. Istraživanja* **2020**, *33*, 2075–2100.
10. Zheng, X.-B.; Kim, Y.-S.; Shin, Y.-R. Cost Effectiveness Analysis in Short Sea Shipping: Evidence from Northeast Asian Routes. *J. Mar. Sci. Eng.* **2021**, *9*, 1340. [CrossRef]
11. Sergii, R.; Bogdan, K. Methodical Tools for Evaluating the Effectiveness of Transport and Logistics Services Management of an Industrial Enterprise. 2021. Available online: https://elartu.tntu.edu.ua/handle/lib/35643 (accessed on 4 November 2022).
12. La Cagnina, L.; Mertoli, F.; Nicotra, A.; Scirè'Chianetta, S.; Ingrao, C.; Di Martino, M. Prevention and Control of Emissions in Intermodal Transport: The Importance of Environmental Protection. *Procedia Environ. Sci. Eng. Manag.* **2019**, *6*, 159–167.
13. Tucki, K.; Mruk, R.; Orynycz, O.; Botwińska, K.; Gola, A.; Bączyk, A. Toxicity of Exhaust Fumes (CO, NOx) of the Compression-Ignition (Diesel) Engine with the Use of Simulation. *Sustainability* **2019**, *11*, 2188. [CrossRef]
14. Kozłowski, E.; Borucka, A.; Świderski, A.; Skoczyński, P. Classification Trees in the Assessment of the Road–Railway Accidents Mortality. *Energies* **2021**, *14*, 3462. [CrossRef]
15. Borucka, A.; Kozłowski, E.; Oleszczuk, P.; Świderski, A. Predictive analysis of the impact of the time of day on road accidents in Poland. *Open Eng.* **2021**, *11*, 142–150. [CrossRef]
16. Świderski, A.; Borucka, A.; Skoczyński, P. Characteristics and assessment of the road safety level in Poland with multiple regression model. In Proceedings of the 22nd Interna tional Scientific Conference, Part I, Trakai, Lithuania, 3–5 October 2018; pp. 92–97.
17. Song, M.; Fisher, R.; Kwoh, Y. Technological challenges of green innovation and sustainable resource management with large scale data. *Technol. Forecast. Soc. Chang.* **2019**, *144*, 361–368. [CrossRef]
18. Małachowski, J.; Ziółkowski, J.; Oszczypała, M.; Szkutnik-Rogoż, J.; Lęgas, A. Assessment of Options to Meet Transport Needs Using the MAJA Multi-Criteria Method. *Arch. Transp.* **2021**, *57*, 25–41. [CrossRef]
19. Ziółkowski, J.; Żurek, J.; Małachowski, J.; Oszczypała, M.; Szkutnik-Rogoż, J. Method for Calculating the Required Number of Transport Vehicles Supplying Aviation Fuel to Aircraft during Combat Tasks. *Sustainability* **2022**, *14*, 1619. [CrossRef]
20. Bellizzi, M.G.; Eboli, L.; Mazzulla, G. Air transport service quality factors: A systematic literature review. *Transp. Res. Procedia* **2020**, *45*, 218–225.
21. Kedzior-Laskowska, M. Technical and Technological Innovations and Quality in Road transport of Goods—Selected Aspects. *Transp. Econ. Logist.* **2019**, *83*, 51–62. [CrossRef]
22. Izadi, A.; Nabipour, M.; Titidezh, O. Cost models and cost factors of road freight transportation: A literature review and model structure. *Fuzzy Inf. Eng.* **2020**, *11*, 257–278. [CrossRef]
23. Litman, T. Transportation cost and benefit analysis. *Vic. Transp. Policy Inst.* **2009**, *31*, 1–19.
24. Hoffmann, N.; Stahlbock, R.; Voß, S. A decision model on the repair and maintenance of shipping containers. *J. Shipp. Trade* **2020**, *5*, 22. [CrossRef]
25. Droździel, P.; Komsta, H.; Krzywonos, H. An analysis of unit repair costs as a function of mileage of vehicles in a selected transport company. *Transp. Probl.* **2014**, *9*, 73–81.
26. Terentyev, A.; Karelina, M.; Egorov, V.; Andreev, A.; Kashyzadeh, K.R. Model for determining optimal routes in complex transport systems. *Transp. Res. Procedia* **2021**, *57*, 679–687. [CrossRef]
27. Abduljabbar, R.; Dia, H.; Liyanage, S.; Bagloee, S.A. Applications of artificial intelligence in transport: An overview. *Sustainability* **2019**, *11*, 189. [CrossRef]
28. Ziółkowski, J.; Lęgas, A.; Szymczyk, E.; Małachowski, J.; Oszczypała, M.; Szkutnik-Rogoż, J. Optimization of the Delivery Time within the Distribution Network, Taking into Account Fuel Consumption and the Level of Carbon Dioxide Emissions into the Atmosphere. *Energies* **2022**, *15*, 5198. [CrossRef]

29. Kesavan, V.; Kamalakannan, R.; Sudhakarapandian, R.; Sivakumar, P. Heuristic and meta-heuristic algorithms for solving medium and large scale sized cellular manufacturing system NP-hard problems: A comprehensive review. *Mater. Today Proc.* **2020**, *21*, 66–72. [CrossRef]
30. Salhi, S.; Thompson, J. An overview of heuristics and metaheuristics. In *The Palgrave Handbook of Operations Research*; Springer Nature: Cham, Switzerland, 2022; pp. 353–403.
31. Shao, Z.; Shao, W.; Pi, D. Effective heuristics and metaheuristics for the distributed fuzzy blocking flow-shop scheduling problem. *Swarm Evol. Comput.* **2020**, *59*, 100747. [CrossRef]
32. Zaied, A.N.H.; Ismail, M.; El-Sayed, S. A survey on meta-heuristic algorithms for global optimization problems. *J. Intell. Syst. Internet Things* **2020**, *1*, 40–47. [CrossRef]
33. Menger, K. *Das Botenproblem, Ergebnisse Eines Mathematischen Kolloquiums 2*; Menger, K., Ed.; Teubner: Leipzig, Germany, 1932; pp. 11–12.
34. Robinson, J.B. On the Hamiltonian Game (A Traveling-Salesman Problem), RAND Research Memorandum RM-303. 1949. Available online: https://www.semanticscholar.org/paper/On-the-Hamiltonian-Game-(A-Traveling-Salesman-Robinson/528d8f79497006766cf65b026205637479e23c7a (accessed on 2 February 2023).
35. Dantzig, G.B.; Fulkerson, D.R.; Johnson, S.M. Solutions of a large scale travelling salesman problem. *Oper. Res.* **1954**, *2*, 393–410.
36. Heller, I. On the travelling salesman's problem. In Proceedings of the Second Symposium in Linear Programming, Washington, DC, USA, 27–29 January 1955; Volume 1.
37. Morton, G.; Land, A.H. A contribution to the 'travelling-salesman' problem. *J. R. Stat. Soc. Ser. B* **1955**, *17*, 185–194. [CrossRef]
38. Flood, M.M. The traveling-salesman problem. *Oper. Res.* **1956**, *4*, 61–75. [CrossRef]
39. Dacey, M.F. Selection of an initial solution for the traveling-salesman problem. *Oper. Res.* **1960**, *8*, 133–134. [CrossRef]
40. Bellman, R. Combinatorial processes and dynamic programming. In *Combinatorial Analysis*; Bellman, R., Hall, M., Jr., Eds.; American Mathematical Society: Providence, RI, USA, 1960; pp. 217–249.
41. Bellman, R. Dynamic programming treatment of the travelling salesman problem. *J. Assoc. Comput. Mach.* **1962**, *9*, 61–63. [CrossRef]
42. Gonzales, R.H. *Solution to the Traveling Salesman Problem by Dynamic Programming on the Hypercube*; Operations Research Center Technical Report Number 18; Massachusetts Institute of Technology: Cambridge, MA, USA, 1962.
43. Held, M.; Karp, R. A dynamic programming approach to sequencing problems. *J. Soc. Ind. Appl. Math.* **1962**, *10*, 196–210. [CrossRef]
44. Fox, K. Production Scheduling on Parallel Lines with Dependencies. Ph.D. Thesis, John Hopkins University, Baltimore, MD, USA, 1973.
45. Picard, J.; Queyranne, M. The time-dependent traveling salesman problem and its application to the tardiness problem in one-machine scheduling. *Oper. Res.* **1978**, *26*, 86–110. [CrossRef]
46. Fox, K.R.; Gavish, B.; Graves, S.C. An n-constraint formulation of the (timedependent) traveling salesman problem. *Oper. Res.* **1980**, *28*, 1018–1021. [CrossRef]
47. Vander Wiel, R.J.; Sahinidis, N.V. An exact solution approach for the timedependent traveling-salesman problem. *Nav. Res. Logist.* **1996**, *43*, 797–820. [CrossRef]
48. Savelsbergh, M.W.P. Local search in routing problems with time windows. *Ann. Oper. Res.* **1985**, *4*, 285–305. [CrossRef]
49. Savelsbergh, M.W.P. The vehicle routing problem with time windows: Minimizing route duration. *INFORMS J. Comput.* **1992**, *4*, 146–154. [CrossRef]
50. Dumas, Y.; Desrosiers, J.; Gélinas, É.; Solomon, M.M. An optimal algorithm for the traveling salesman problem with time windows. *Oper. Res.* **1995**, *43*, 367–371. [CrossRef]
51. Dash, S.; Günlük, O.; Lodi, A.; Tramontani, A. A time bucket formulation for the traveling salesman problem with time windows. *INFORMS J. Comput.* **2012**, *24*, 132–147. [CrossRef]
52. Gutin, G.; Punnen, A.P. (Eds.) *The Traveling Salesman Problem and Its Variations*; Springer: New York, NY, USA, 2007.
53. Laporte, G.; Mercure, H.; Nobert, Y. Generalized Traveling Salesman Problem through n sets of nodes: The asymmetrical cases. *Discret. Appl. Math.* **1987**, *18*, 185–197. [CrossRef]
54. Chisman, J.A. The clustered traveling salesman problem. *Comput. Oper. Res.* **1975**, *2*, 115–119. [CrossRef]
55. Baniasadi, P.; Foumani, M.; Smith Miles, K.; Ejov, V. A transformation technique for the clustered generalized traveling salesman problem with applications to logistics. *Eur. J. Oper. Res.* **2020**, *285*, 444–457. [CrossRef]
56. Bektas, T. The multiple traveling salesman problem: An overview of formulations and solution procedures. *Omega* **2006**, *34*, 209–219. [CrossRef]
57. Kara, I.; Bektas, T. Integer linear programming formulations of multiple salesman problems and its variations. *Eur. J. Oper. Res.* **2006**, *174*, 1449–1458. [CrossRef]
58. Fanjul Peyro, L.; Ruiz, R.; Perea, F. Reformulations and an exact algorithm for unrelated parallel machine scheduling problems with setup times. *Comput. Oper. Res.* **2019**, *101*, 173–182. [CrossRef]
59. Podstawka, M. Applicaton of the Traveling Salesman Problem to Transportation Cost Optimization in the Sylpo s.c. Company. Master's Thesis, Lublin University of Technology, Lublin, Poland, 2009. (In Polish).
60. Volgenant, T.; Jonker, R. On Some Generalizations of the Travelling-Salesman Problem. *J. Oper. Res. Soc.* **1987**, *38*, 1073. [CrossRef]

61. Bienstock, D.; Goemans, M.X.; Simchi-Levi, D.; Williamson, D. A note on the prize collecting traveling salesman problem. *Math. Program.* **1993**, *59*, 413–420. [CrossRef]
62. Balas, E. The prize collecting traveling salesman problem. *Networks* **1989**, *19*, 621–636. [CrossRef]
63. Gavish, B.; Graves, S.C. *The Travelling Salesman Problem and Related Problems*; Working Paper OR-078-78; Operations Research Center, MIT: Cambridge, MA, USA, 1978.
64. Orman, A.; Williams, H. A Survey of Different Integer Programming Formulations of the Travelling Salesman Problem. In *Optimisation, Econometric and Financial Analysis*; Kontoghiorghes, E.J., Gatu, C., Eds.; Springer: Berlin/Heidelberg, Germany, 2007; pp. 91–104. [CrossRef]
65. Miller, C.E.; Tucker, A.W.; Zemlin, R.A. Integer programming formulation of travelling salesman problems. *J. ACM* **1960**, *3*, 326–329. [CrossRef]
66. Finke, G.; Claus, A.; Gunn, E. A two-commodity network flow approach to the travelling salesman problem. In *Combinatorics, Graph Theory and Computing, Proceedings of the 14th South Eastern Conference*; Atlantic University: Boca Raton, FL, USA, 1983.
67. Wong, R.T. Integer programming formulations of the travelling salesman problem. In Proceedings of the IEEE International Conference of Circuits and Computers, New York, NY, USA, 1–3 October 1980; pp. 149–152.
68. Claus, A. A new formulation for the travelling salesman problem. *SIAM J. Alg. Disc. Math.* **1984**, *5*, 21–25. [CrossRef]
69. Vajda, S. *Mathematical Programming*; Addison-Wesley: London, UK, 1961.
70. Desrochers, M.; Laporte, G. Improvements and extensions to the Miller–Tucker–Zemlin subtour elimination constraints. *Oper. Res. Lett.* **1991**, *10*, 27–36. [CrossRef]
71. Mason, A.J. OpenSolver—An Open Source Add-in to Solve Linear and Integer Progammes in Excel. In *Operations Research Proceedings 2011*; Klatte, D., Lüthi, H.-J., Schmedders, K., Eds.; Springer: Berlin/Heidelberg, Germany, 2012; pp. 401–406. [CrossRef]
72. Available online: http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/atsp/br17.atsp.gz (accessed on 9 November 2022).
73. Available online: http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/tsp/bayg29.tsp.gz (accessed on 9 November 2022).
74. Available online: http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/atsp/ftv44.atsp.gz (accessed on 9 November 2022).