Computers & Operations Research 39 (2012) 2161–2167

Contents lists available at SciVerse ScienceDirect

Computers & Operations Research

journal homepage: www.elsevier.com/locate/caor

The Traveling Salesman Problem with Draft Limits

Jørgen Glomvik Rakke [a,*], Marielle Christiansen [a], Kjetil Fagerholt [a], Gilbert Laporte [b]

[a] Department of Industrial Economics and Technology Management, The Norwegian University of Science and Technology, Trondheim, Norway
[b] Canada Research Chair in Distribution Management, HEC Montréal, Montréal, Canada H3T 2A7

ARTICLE INFO

Available online 7 November 2011

Keywords:
Maritime transportation
Traveling salesman problem
Draft limits

ABSTRACT

In maritime transportation, routing decisions are sometimes affected by draft limits in ports. The draft of a ship is the distance between the waterline and the bottom of the ship and is a function of the load onboard. Draft limits in ports can thus prevent ships to enter these ports fully loaded and may impose a constraint on the sequence of visits made by a ship. This paper introduces the *Traveling Salesman Problem with Draft Limits* (TSPDL), which is to determine an optimal sequence of port visits under draft limit constraints. We present two mathematical formulations for the TSPDL, and suggest valid inequalities and strengthened bounds. We also introduce a set of instances based on TSPLIB. A branch-and-cut algorithm is applied on both formulations for all these instances. Computational results show that introducing draft limits make the problem much harder to solve. They also indicate that the proposed valid inequalities and strengthened bounds significantly reduce both the number of branch-and-bound nodes and the solution times.

1. Introduction

In maritime transportation, routing decisions are affected by a number of constraints, one of which is *draft limit*. The draft of a ship is the distance between the waterline and the bottom of a ship. It is equal to the draft of the empty ship, plus a function of the load onboard the ship. Each port has a draft limit, which imposes a constraint on the sequence of visits made by a ship (see Fig. 1). Some ports have a very low draft limit, which can sometimes be problematic. For example, the break bulk terminal in the port of Kota Kinabalu in Malaysia has a draft limit of 9.2 m, while typical fully loaded Panamax vessels have drafts of about 12 m, thus preventing such ships from entering this port when fully loaded. Determining a feasible and optimal sequence of visits under draft limit constraints gives rise to a *Traveling Salesman Problem with Draft Limits* (TSPDL).

Formally, the TSPDL is defined on a directed graph $G=(V,A)$, where $V=\{1,\ldots,n\}$ is the vertex set and $A=\{(i,j): i,j\in V, i\neq j\}$ is the arc set. Vertex 1 denotes a depot and the other ports constitute the set $V\backslash\{1\}$. The depot has a demand $d_1=0$ and every port $i\in V\backslash\{1\}$ has a non-negative integer demand d_i. For convenience, we define the draft of a ship as its load in terms of demand on board. The draft limit of port $i\in V$ is equal to l_i and we assume without loss of generality that the ports are labeled so that $l_i\geq l_{i+1}$ $(i=1,\ldots,n-1)$.

The TSPDL consists of determining a least cost Hamiltonian circuit $(i_1=1,i_2,\ldots,i_n)$ on G starting at vertex 1 with a load $\sum_{j=1}^{n} d_j$, and delivering a load d_i at each vertex $i\in V\backslash\{1\}$ without ever exceeding the draft limit. In other words, the circuit must satisfy $\sum_{j=1}^{n} d_j-\sum_{r=1}^{t-1} d_{i_r}\leq l_{i_t}$ for $t=2,\ldots,n$. The TSPDL is clearly NP-hard since it reduces to the *Asymmetric Traveling Salesman Problem* (ATSP) whenever $l_i\geq\sum_{j=1}^{n} d_j$ for all $i\in V$.

Not every TSPDL instance is feasible. Here we provide a necessary and sufficient condition for feasibility. To this end, denote by $\pi=(i_1,\ldots,i_n)$ a TSPDL solution and let L_i^{π} be the load onboard the ship upon entering i in solution π. If $\pi=(1,\ldots,n)$ we simply write L_i instead of L_i^{π}. Then $L_1=\sum_{j=1}^{n} d_j$ and $L_i=L_{i-1}-d_{i-1}$ for all $i\in V\backslash\{1\}$.

Proposition 1.1. *The TSPDL is feasible if and only if $L_i\leq l_i$ for all $i\in V$. If the solution $(1,\ldots,n)$ is infeasible, then no other solution is feasible.*

Proof. The condition is clearly sufficient since it means that $\pi=(1,\ldots,n)$ is feasible. Now suppose the solution is violated for π and let $(1,\ldots,i^*-1)$ be the longest feasible sequence of π starting at vertex 1, i.e., i^* is the first vertex for which $L_{i^*}>l_{i^*}$. Let $S=\{1,\ldots,i^*\}$. No feasible solution λ in which i^* is preceded by only some vertices of S exists because this would yield $L_{i^*}^{\lambda}\geq L_{i^*}>l_{i^*}$, which is again infeasible. Now suppose there exists a feasible solution ϕ in which i^* is preceded by some vertices now following it in π, and let k^* be the first such vertex in ϕ. Since k^* is only preceded by some vertices of S, we have $L_{k^*}^{\phi}\geq L_{i^*}>l_{i^*}\geq l_{k^*}$, which is also infeasible. Hence, the condition is necessary since

* Corresponding author.
E-mail address: jorgen.rakke@iot.ntnu.no (J. Glomvik Rakke).

0305-0548/$ - see front matter © 2011 Elsevier Ltd. All rights reserved.
doi:10.1016/j.cor.2011.10.025

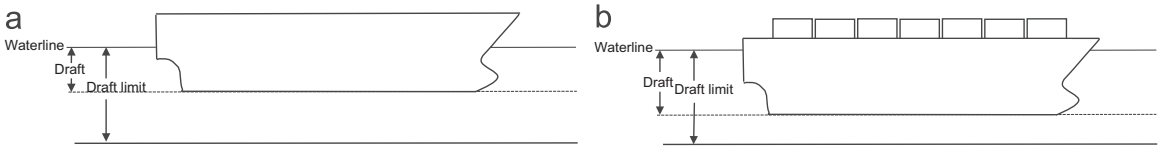

Fig. 1. Illustration of draft and draft limit. (a) Unladen ship. (b) Laden ship.

there exists no TSPDL solution when it is violated, i.e., when π is infeasible. □

To our knowledge, the TSPDL has never been previously studied. In addition, the scientific literature on related problems is rather limited. The TSPDL is indirectly related to the *TSP with Precedence Constraints* (TSPPC) [2]. In the TSPPC, the precedence relations are expressed in matrix form, whereas in the TSPDL they are more complex and can take the form of single or multiple disjunctions. For example, consider three ports i, j and k, where $l_i = l_j = 5$, $l_k = 3$, $d_i = d_j = 2$, and $d_k = 1$. Then i or j must be visited before k for a feasible solution to exist. In the area of maritime routing and scheduling, draft limits have been considered in a number of papers, e.g., Hennig [11], Song and Furman [21], Christiansen et al. [4], but the draft limit is not the main focus of these articles. In the area of vehicle routing, several problems describing combined routing and loading problems have been studied in recent years, e.g., Petersen and Madsen [18], Felipe et al. [9], Erdoğan et al. [7], Cordeau et al. [5], Battarra et al. [3], Iori and Martello [12], but none considers any constraints close to a draft limit. Finally, there exist some maritime variants of the TSP, e.g., Fagerholt and Christiansen [8], but again these do not incorporate draft limits.

Our main contribution is to introduce, model and solve the TSPDL, a new problem relevant to several ship routing applications. The remainder of this paper is organized as follows. In Section 2 we present two mathematical models and valid inequalities for the TSPDL. Section 3 describes our algorithm. Computational results are presented in Section 4, followed by some concluding remarks in Section 5.

2. Mathematical models

We have developed two mathematical models for the TSPDL. The first, called *GG*, is based on the Gavish and Graves [10] formulation for the ATSP. The second, called *SD*, extends the Sherali and Driscoll [20] ATSP formulation. The relative interest of *GG* lies in its compactness and simplicity, whereas *SD* is one of the strongest available formulations for the ATSP [15] and can easily handle flow variables. One of our goals is to provide an empirical comparison of these two formulations for the TSPDL. We first present the two models, followed by valid inequalities for the TSPDL.

2.1. GG formulation

Let x_{ij} be a binary variable equal to 1 if and only if arc (i, j) is used in the optimal solution, and let y_{ij} denote the load onboard the ship on arc (i, j). The model is then

$$(GG): \text{ minimize } \sum_{(i,j)\in A} c_{ij}x_{ij} \tag{2.1}$$

$$\text{subject to} \sum_{i\in V} x_{ij} = 1, \quad j \in V, \tag{2.2}$$

$$\sum_{j\in V} x_{ij} = 1, \quad i \in V, \tag{2.3}$$

$$\sum_{i\in V} y_{ij} - \sum_{i\in V} y_{ji} = d_j, \quad j \in V\backslash\{1\}, \tag{2.4}$$

$$\sum_{j\in V} y_{1j} = \sum_{i\in V} d_i, \tag{2.5}$$

$$\sum_{i\in V} y_{i1} = 0, \tag{2.6}$$

$$0 \le y_{ij} \le l_j x_{ij}, \quad (i,j) \in A, \tag{2.7}$$

$$x_{ij} \in \{0,1\}, \quad (i,j) \in A. \tag{2.8}$$

In this formulation, constraints (2.2) and (2.3) define the in-degree and the out-degree of each vertex. Constraints (2.4) ensure that the demand of each vertex is satisfied and also prevent the creation of subtours. Constraints (2.5) and (2.6) state that the ship leaves the depot fully loaded and returns to it empty. Constraints (2.7) impose the draft limit at each vertex.

2.2. SD formulation

The *SD* formulation extends the Gavish and Graves model through the incorporation of lifted Miller, Tucker, and Zemlin [14] (MTZ) subtour elimination constraints. These constraints use additional variables u_i denoting the position of vertex i in the circuit, and variables t_{ij} denoting the position of arc (i, j) in the circuit. The *SD* formulation is defined by (2.1)–(2.8) and

$$\sum_{j\in V\backslash\{i,1\}} t_{ij} + (n-1)x_{i1} = u_i, \quad i \in V\backslash\{1\}, \tag{2.9}$$

$$\sum_{j\in V\backslash\{i,1\}} t_{ji} + 1 = u_i, \quad i \in V\backslash\{1\}, \tag{2.10}$$

$$x_{ij} \le t_{ij} \le (n-2)x_{ij}, \quad i,j \in V\backslash\{1\}, \tag{2.11}$$

$$u_j + (n-2)x_{ij} + (n-1)(1-x_{ji}) \le t_{ij} + t_{ji} \le u_j - (1-x_{ji}), \quad i,j \in V\backslash\{1\}, \tag{2.12}$$

$$1 + (1-x_{1i}) + (n-3)x_{i1} \le u_i \le (n-1) - (n-3)x_{1i} - (1-x_{i1}), \quad i \in V\backslash\{1\}, \tag{2.13}$$

$$t_{ij} \ge 0, \quad (i,j) \in A, \tag{2.14}$$

$$u_i \ge 0, \quad i \in V. \tag{2.15}$$

In this formulation, constraints (2.9)–(2.13) prevent the creation of subtours. Their derivation is detailed in Sherali and Driscoll [20]. Since the *GG* model already prevents subtours, constraints (2.9)–(2.13) can be viewed as valid inequalities for the *GG* model.

2.3. Valid inequalities

Formulations *GG* and *SD* can both be reinforced through the introduction of valid inequalities.

2.3.1. Subtour elimination constraints and T-lift cuts

We impose the classical subtour elimination constraints (SECs)

$$\sum_{i,j\in S} x_{ij} \le |S| - 1, \quad S \subset V,\ 2 \le |S| \le n-1, \tag{2.16}$$

initially proposed by Dantzig et al. [6]. Since their number is exponential, we initially imposed the two-vertex subtour elimination constraints

$$x_{ij}+x_{ji}\le 1 \quad (i,j)\in A, \tag{2.17}$$

and we generate the remaining ones in a branch-and-cut algorithm. As proposed by Bales and Fischetti [1], we have lifted the SECs (2.16) as

$$x_{rt}+x_{ts}+x_{rs}+\sum_{i,j\in S}x_{ij}\le |S|, \quad S\subset V,\ 2\le |S|\le n-2, \tag{2.18}$$

where r, s and t are three distinct vertices satisfying $r,s\in V\backslash\{S\}$ and $t\in S$. These T-lift cuts are dynamically generated only if $x_{rt}+x_{ts}+x_{rs}>1$.

2.3.2. Lower bounds on the u_i variables

Lower bounds f_i on the u_i variables in the *SD* formulation can be computed by using the draft limits at the vertices.

Proposition 2.1. *The constraints*

$$u_i\ge f_i=\left\lceil\frac{\sum_{j\in V}d_j-l_i}{\max_{j\in V}\{d_j\}}\right\rceil+1, \quad i\in V\backslash\{1\} \tag{2.19}$$

are valid inequalities for SD.

Proof. The ceiling expression in constraints (2.19) simply computes the least number of vertices that must be visited before entering vertex i to ensure that the load on the ship upon entering vertex i does not exceed l_i. □

The bounds defined by constraints (2.19) can be rather weak if the demand variance is large. A stronger bound can be obtained through the solution of the following *Knapsack Problem*:

$$\text{minimize } f_i=\sum_{j\in V}v_j+1, \tag{2.20}$$

$$\text{subject to } \sum_{j\in V}d_jv_j\ge\sum_{j\in V}d_j-l_i, \tag{2.21}$$

$$v_j\in\{0,1\}, \quad j\in V. \tag{2.22}$$

Lower bounds can also be imposed on some sums of u_i variables.

Proposition 2.2. *The constraints*

$$\sum_{j\in S}u_j\ge |S|\min_{j\in S}\{f_j\}+\sum_{j=1}^{|S|-1}j, \quad S\subseteq V, \quad \max_{j\in S}\{f_j\}-\min_{j\in S}\{f_j\}\le |S|-2 \tag{2.23}$$

are valid inequalities for SD.

Proof. The computation of these bounds is based on the observation that the u_i variables associated with any subset S of V must take $|S|$ different values at least equal to $\min_{j\in S}\{f_j\}$. □

To illustrate, if $S=\{6,7,8,9\}$ and $f_6=f_7=5$, $f_8=f_9=6$, then the position variables u_6 to u_9 must take at least four different values at least equal to 5, i.e., $\sum_{j=6}^{9}u_j\ge 26$.

2.3.3. Upper bounds on sums of y_{ij} variables

Upper bounds on sums of y_{ij} variables can be generated in a similar fashion.

Proposition 2.3. *The constraints*

$$\sum_{i\in V,j\in S}y_{ij}\le |S|\min\left\{\sum_{j\in V}d_j,\max_{j\in S}\{l_j\}\right\}-\sum_{j=1}^{|S|-1}(|S|-j)d'_j,$$

$$S\subseteq V, \quad \max_{j\in S}\{l_j\}-\min_{j\in S}\{l_j\}\le |S|-2 \tag{2.24}$$

are valid inequalities for the GG and SD formulations.

Proof. The proof is similar to that of Proposition 2.2. □

Here d'_j refers to the j-th vertex of a non-decreasing ordering of S in terms of demand.

2.3.4. Draft limit and demand induced cuts on x_{ij} variables

Valid inequalities can be derived from the draft limits and vertex demands.

Proposition 2.4. *The constraints*

$$\sum_{i\in S}x_{i1}=1, \quad S\subseteq V\backslash\{1\}, \quad \max_{j\in S}f_j-\min_{j\in S}f_j\le |S|-2,$$
$$\max_{j\in S}f_j+|S|-1=n-1 \tag{2.25}$$

are valid inequalities for SD.

Proof. Let S be a set of vertices where $\max_{j\in S}\{f_j\}-\min_{j\in S}\{f_j\}\le |S|-2$ and $\max_{j\in S}\{f_j\}+|S|-1=n-1$, we then know that at least one of the vertices in this set has to be the last vertex visited before returning to the depot. □

Proposition 2.5. *The constraints*

$$\sum_{i\in\overline{S},j\in S\backslash\overline{S}}x_{ij}\ge 1, \quad S\subseteq V\backslash\{1\},\overline{S}\subset S, \quad \max_{j\in S}\{f_j\}-\min_{j\in S}\{f_j\}\le |S|-2,$$
$$\max_{j\in S}\{f_j\}+|S|-1=n-1, \quad \min_{j\in\overline{S}}\{f_j\}=\max_{j\in S}\{f_j\} \tag{2.26}$$

are valid inequalities for SD.

Proof. At least one vertex j with $f_j=\max_{i\in S}\{f_i\}$ has to connect to a vertex k with $f_k<\max_{i\in S}\{f_i\}$. Defining $\overline{S}$ as the set of vertices with a lower bound on u of $\max_{j\in S}\{f_j\}$, the thesis follows. □

3. Branch-and-cut algorithm

We have applied a branch-and-cut algorithm to models *GG* and *SD* in which the integrality conditions are initially relaxed and to which two-vertex subtour elimination constraints (2.17) are incorporated. The constraints (2.23) or (2.24) and constraints (2.25) and (2.26) are also separated a priori and added to the initial relaxation for all sets S_i of V, i.e., $S_1\cap S_2=\emptyset$. The remaining violated inequalities are generated dynamically whenever a violation exceeding a preset threshold $\epsilon=0.001$ is identified. To separate the SECs (2.16), we used the Padberg and Rinaldi [16] MINCUT algorithm. Whenever such a violated constraint is identified, we also search for a T-lift cut (2.18). For both the SECs and the T-lifts only the most violated cuts are added in each iteration.

4. Computational study

To evaluate the two proposed models and to test the relative difficulty of the TSPDL compared to the TSP, we introduce a test set of 240 instances based on well-known instances from TSPLIB. These are described in Section 4.1. First, the proposed formulations were applied to the original TSP in order to get an indication of the solution time and the relative optimality gap of the TSP compared to the TSPDL in Sections 4.2 and 4.3. To test these two formulations on the generated test sets we first do a preliminary testing of the proposed cuts and bounds on the two smallest instance sets in Section 4.4 before we present the computational results for all instances in Section 4.5.

The branch-and-cut algorithm was implemented using Xpress-MP, 7.2 and custom callbacks were written in Mosel 3.2.0. The MINCUT algorithm was written in C++, and called as a module from Mosel. All tests were executed on HP dl160 G3 machines with 2x3.0 GHz Intel E5472 Xeon CPU and 16 GB of RAM. The setup has eight cores and Xpress was allowed to utilize all. All tests had a CPU time limit of 10 000 s.

4.1. Test sets

We have used eight TSPLIB instances with size ranging from 14 to 48 vertices. The instances can be seen in the leftmost column of Table 1. From the TSPLIB instances we generated 10 new instances for 10%, 25% and 50% of vertices with restricting draft limit, respectively. This means that for the p % instances, $\lceil n \times p\% \rceil$ vertices have a draft limit less than the total demand. In these test sets all vertices, except for the depot, have a demand of one. Thus, the draft limits have been randomly generated between 1 and $n-1$. To ensure that all the generated instances have a feasible solution we have applied Proposition 1.1. To easier describe the results we define a *test set* as all instances generated from one TSPLIB instance, while a *test group* is defined as all test instances from one test set with the same percentage of vertices with restricting draft limits. All test instances can be found at http://jgr.no/tspdl.

4.2. Testing the models for TSPLIB instances

To our knowledge, the use of SECs (2.16) has never been discussed in the literature in combination with the *GG* or *SD* formulation. To validate the results of the combination of these two formulations we have run tests on the original TSP instances. In Table 1 the gap columns show the average relative optimality gap at the root node for the instances solved to optimality. Further, time and nodes are the computational time and number of nodes used to find and prove optimality, respectively.

Table 1 shows that both formulations are strengthened by the introduction of SECs (2.16). For both formulations the introduction of SECs also drastically decreases the computational time and the number of nodes explored.

4.3. Linear relaxation results

A commonly used criterion to measure the strength of a model is the relative optimality gap. Since the TSP and the TSPDL are strongly related problems we use the relative optimality gap as an indicator of the difficulty of the TSPDL compared to the TSP. The gap in Table 2 is the average optimality gap at the root node for the instances that have been solved to optimality. We can clearly see that the gap increases as the number of vertices with restricting draft limits increases.

4.4. Preliminary testing of strengthened bounds, SECs and T-lift cuts

To determine whether we should include the strengthened bounds from (2.20)–(2.22) and the cuts (2.16) and (2.18) we have performed tests on the two smallest instances from TSPLIB. In all tables presented, $p\%$ refers to the group of test instances with restrictive draft limits on p% of the vertices as described in Section 4.1. The time and number of nodes provided are averages over the 10 instances. If an instance is not solved within the given maximum time of 10 000 s we calculate the average using 10 000 s and display the result as using more than the average time ($>\#s$). The first and second columns in the result tables indicate the test sets and test groups, respectively.

First, we tested the proposed strengthening of bounds in (2.20)–(2.22) against the regular bounds on the u_i variables in the *SD* formulation. As we can clearly see from Table 3, the strengthened bounds outperform the regular *SD* bound. On average, equal or less time is used for all test groups and fewer nodes are explored in five out of six cases. With strengthened bounds we are also able to solve one more instance than with regular bounds. We will therefore include the strengthened bounds in the remaining tests.

Next, we wanted to test the effect of the SECs (2.16) on the TSPDL for both the *GG* and *SD* formulations. From Table 4 we see that on average the tests with SECs explore significantly fewer nodes than the tests without SECs. For the 10% instances the SECs reduce the number of nodes by more than 80%. In two-thirds of the test groups less time is spent to prove optimality when using

Table 2
The table shows how the number of vertices with draft limit affects the gap.

TSPLIB instance	GG				SD			
	Gap				Gap			
	TSP	10%	25%	50%	TSP	10%	25%	50%
burma14	4.4	5.7	11.5	15.2	4.4	5.7	11.5	15.2
ulysses16	3.7	3.8	7.8	10.6	3.7	3.8	7.8	10.6
ulysses22	6.8	7.2	10.8	18.4	6.8	7.1	10.8	18.3
fri26	5.0	7.5	15.3	20.3	5.0	7.5	16.1	20.3
bayg29	3.7	8.6	9.4	11.6	3.7	8.6	9.4	11.6
gr17	12.9	14.6	17.9	29.3	12.9	14.6	17.7	29.2
gr21	0.0	4.1	7.9	25.9	0.0	4.1	7.9	25.9
gr48	5.2	8.4	–	–	5.2	8.4	–	–
Avg.	5.2	7.5	11.5	18.8	5.2	7.5	11.6	18.7

Table 1
Test of formulations on the original TSPLIB instances. The table shows the percent gap and the CPU time in seconds for all test groups.

TSPLIB instance	GG						SD					
	No SECs			SECs			No SECs			SECs		
	Gap	Time	Nodes	Gap	Time	Nodes	Gap	Time	Nodes	Gap	Time	Nodes
burma14	4.4	0.1	61	0.0	0.1	3	4.4	0.4	119	0.0	0.1	1
ulysses16	3.7	0.2	561	0.0	0.1	3	3.7	1.3	649	0.0	0.5	3
ulysses22	6.8	0.8	1657	0.0	0.2	3	6.8	6.5	3937	0.0	1.0	3
fri26	5.0	3.4	5755	0.0	0.5	5	5.0	25.5	5629	0.0	1.9	5
bayg29	3.7	1.2	1315	0.1	0.4	11	3.7	10.1	1941	0.1	2.4	11
gr17	12.9	0.6	2429	0.0	0.1	3	12.9	3.1	3001	0.0	0.5	3
gr21	0.0	0.0	1	0.0	0.0	1	0.0	0.1	1	0.0	0.1	1
gr48	5.2	428.4	220 217	1.7	6.3	471	5.2	4697.3	322 403	1.7	40.8	483
Avg.	5.2	54.4	29 000	0.2	1.0	63	5.2	593.0	42 210	0.2	5.9	64

SECs. Based on these results, the SECs will be included in the remaining tests.

Finally, the effect of adding T-lift cuts in addition to the SECs was tested. From Table 5 it is not as easy as in the two previous tests to see whether the cuts make a positive impact on the solution process. The time to optimality is on average equal or slightly worse with the cuts, while the number of nodes is smaller for the *SD* formulation (2.3%) and worse for the *GG* formulation (0.4%). When including T-lift cuts at most the same number of nodes was explored in all *SD* test groups, and in four out of six *GG* test groups, and the time to optimality at most equal, in nine out of 12 test groups. We chose to use the T-lift cuts in the final tests since the remaining instances are larger and the time to solve each node will therefore increase. This means that the time spent generating the cuts will be less important than the time saved by solving fewer nodes.

These results from the preliminary tests were used to determine the settings for further testing. Based on the results from Section 4.4 we will use strengthened bounds, SECs, and T-lift cuts.

4.5. Computational results with problem specific valid inequalities

Finally, we have tested the effect of the problem specific valid inequalities (2.23)–(2.26) on all test sets. Here, the test setup from Section 4.4 including strengthened bounds, SECs, and T-lift cuts is called *basic*, and the test setup also including the problem specific valid inequalities is called *extended*. In these final tests we call the test setup found in the preliminary testing the base setup, while the test setup including the problem specific valid inequalities is called the extended setup.

Table 6 shows that the *GG* formulation can solve five more instances including the valid inequalities, while the *SD* formulation can solve four more. We can also see the *SD* formulation outperforms the *GG* formulation in terms of number of instances solved. It solves 198 out of 240 instances, while the *GG* formulation only solves 186. Table 7 also shows that the *SD* formulation generates on average fewer nodes and requires less time than the *GG* formulation. In terms of average solution time the tests with the valid inequalities are at least as good those as without the valid inequalities in 18 out of 24 test groups for the *GG* formulation, and in 14 out of 24 groups for the *SD* formulation. The

Table 3
Test of strengthened bounds on the SD formulation. The table shows the number of nodes used and time spent in seconds.

	SD			
	Regular bounds		Strengthened bounds	
	Time	Nodes	Time	Nodes
burma14				
10%	0.4	95.0	0.4	90.2
25%	6.3	14 244.2	1.6	1996.4
50%	32.5	92 242.0	2.1	2813.4
ulysses16				
10%	0.9	452.4	0.9	459.2
25%	3.4	4925.2	3.2	4804.2
50%	> 1028.4	> 165 046.6	15.5	16 850.6
Avg.	> 178.7	> 46 168	3.9	4502

Table 4
Test of subtour elimination constraints. The table shows number of nodes used and time spent in seconds.

	GG				SD			
	No SECs		SECs		No SECs		SECs	
	Time	Nodes	Time	Nodes	Time	Nodes	Time	Nodes
burma14								
10%	0.1	59.6	0.1	8.2	0.4	90.2	0.2	19.2
25%	1.7	12 770.1	2.1	6582.5	1.6	1996.4	1.3	1199.2
50%	6.9	48 945.2	9.1	28 566.3	2.1	2813.4	1.7	1751.8
ulysses16								
10%	0.2	327	0.1	3.4	0.9	459.2	0.3	3.2
25%	0.5	2312.4	0.7	1244.1	3.2	4804.2	2.5	2701.5
50%	> 1008.5	> 552 155.9	> 1018.1	> 544 627.9	15.5	16 850.6	12.4	9847.3
Avg.	> 169.7	> 102 761.7	> 171.7	> 96 838.7	3.9	4502.3	3.1	2587.0

Table 5
Test of T-lift cuts.

	GG				SD			
	No T-lift		T-lift		No T-lift		T-lift	
	Time	Nodes	Time	Nodes	Time	Nodes	Time	Nodes
burma14								
10%	0.1	8.2	0.1	8.2	0.2	19.2	0.2	13.2
25%	2.1	6582.5	2.4	6659.0	1.3	1199.2	1.3	1132.2
50%	9.1	28 566.3	11.8	34 801.8	1.7	1751.8	1.8	1679.6
ulysses16								
10%	0.1	3.4	0.1	3.3	0.3	3.2	0.3	2.1
25%	0.7	1244.1	0.7	1228.7	2.5	2701.5	2.5	2541.7
50%	> 1018.1	> 544 627.9	> 1019.6	> 540 921.5	12.4	9847.3	12.3	9801.1
Avg.	> 171.7	> 96 838.7	> 172.5	> 97 270.4	3.1	2587.0	3.1	2528.3

Table 6
Number of instances solved to optimality.

	GG		SD	
	Basic	**Extended**	**Basic**	**Extended**
burma14				
10%	10	10	10	10
25%	10	10	10	10
50%	10	10	10	10
ulysses16				
10%	10	10	10	10
25%	10	10	10	10
50%	9	10	10	10
ulysses22				
10%	10	10	10	10
25%	10	10	10	10
50%	4	5	6	7
fri26				
10%	10	10	10	10
25%	6	6	8	8
50%	5	5	5	7
bayg29				
10%	9	9	10	10
25%	6	6	7	8
50%	1	1	1	1
gr17				
10%	10	10	10	10
25%	10	10	10	10
50%	8	10	10	10
gr21				
10%	10	10	10	10
25%	10	10	10	10
50%	6	7	10	10
gr48				
10%	7	7	7	7
25%	0	0	0	0
50%	0	0	0	0
Total	181	186	194	198

Table 7
The average time in seconds and number of nodes for the basic and extended test setup.

	GG				SD			
	Basic		**Extended**		**Basic**		**Extended**	
	Time	**Nodes**	**Time**	**Nodes**	**Time**	**Nodes**	**Time**	**Nodes**
burma14								
10%	0.1	8.2	0.1	8.2	0.2	13.2	0.3	17.2
25%	2.4	6659.0	2.1	5806.8	1.3	1132.2	1.2	1058.8
50%	11.8	34 801.8	8.7	24 873.9	1.8	1679.6	1.9	1902.8
ulysses16								
10%	0.1	3.3	0.1	3.3	0.3	2.1	0.3	2.1
25%	0.8	1228.7	0.8	1228.7	2.6	2541.7	2.5	2385.1
50%	> 1020.0	> 540 921.5	16.5	31 603.4	12.6	9801.1	2.1	1327.6
ulysses22								
10%	0.6	468.5	0.6	459.7	2.1	372.1	2.7	467.9
25%	237.4	266 099.8	189.7	210 022.1	196.4	113 094.4	167.5	99 915.2
50%	> 6151.1	> 3 205 865.2	> 5377.6	> 2 901 328.7	> 4402.1	> 2 226 464.7	> 3502.2	> 1 823 024.6
fri26								
10%	55.3	49 349.9	55.0	49 349.7	88.6	25 669.9	121.4	28 706.6
25%	> 5192.1	> 2 995 477.6	> 5264.1	> 3 049 302.3	> 4139.4	> 1 814 477.6	> 4168.6	> 1 856 987.9
50%	> 5561.7	> 2 973 369.3	> 5640.6	> 3 058 586.8	> 5507.3	> 2 680 548.3	> 4455.1	> 1 811 333.9
bayg29								
10%	1014.8	508 423.6	1014.6	508 384.4	337.5	72 688.7	260.7	41 716.9
25%	> 4031.4	> 2 019 633.8	> 4030.3	> 2 019 393.1	> 3880.4	> 1 650 065.9	> 3391.2	> 1 294 143.1
50%	> 9038.1	> 4 513 777.0	> 9043.0	> 4 514 439.6	> 9059.0	> 4 506 191.3	> 9077.3	> 4 508 332.5
gr17								
10%	0.2	73.8	0.1	23.2	0.6	72.1	0.5	25.0
25%	36.0	63 369.8	36.0	63 433.8	10.2	7135.7	7.5	5592.3
50%	> 2644.2	> 1 336 910.7	655.4	1 054 622.3	31.1	22 520.2	13.1	8545.5

Table 7 (*continued*)

	GG				SD			
	Basic		Extended		Basic		Extended	
	Time	Nodes	Time	Nodes	Time	Nodes	Time	Nodes
gr21								
10%	1.6	1909.4	1.5	1874.1	2.4	608.5	3.4	892.3
25%	163.9	182 553.3	160.7	177 533.8	36.8	20 343.6	41.5	22 500.8
50%	> 4471.0	> 2 489 343.6	> 3717.9	> 2 333 099.4	539.8	203 451.8	437.3	176 175.0
gr48								
10%	> 3079.0	> 1 509 894.4	> 3079.3	> 1 509 930.2	> 3627.2	> 1 514 322.8	> 3632.3	> 1 515 089.4
25%	> 10 000.0	> 5 000 000.0	> 10 000.0	> 5 000 000.0	> 10 000.0	> 5 000 000.0	> 10 000.0	> 5 000 000.0
50%	> 10 000.0	> 5 000 000.0	> 10 000.0	> 5 000 000.0	> 10 000.0	> 5 000 000.0	> 10 000.0	> 5 000 000.0
Avg.	> 2613.1	> 1 362 505.9	> 2429.0	> 1 313 137.8	> 2161.7	> 1 036 383.2	> 2053.8	> 969 422.6

average number of nodes is also not larger than without valid inequalities in 19 out of 24 test groups, and in 15 out of 24 test groups for the *GG* and *SD* formulation, respectively.

5. Concluding remarks

We have introduced the TSPDL, a new variant of the TSP arising in maritime transportation. We have developed two different mathematical models and improved the bounds on some variables by solving knapsack problems. Valid inequalities for these models were also introduced and tested. Our computational experiments show that there is a significant advantage in adding both the strengthened bounds and the subtour elimination constraints. Results were more ambiguous for the T-lift cuts, but leaned towards including them. The problem specific valid inequalities enabled us to solve more instances, and reduce the computational time, as well as the number of nodes explored in the branch-and-cut tree. Even though the *GG* formulation has much shorter solution times for the LPs our experiments showed that the *SD* formulation performed better overall in the global search. Solving larger instances would require the development of a strong heuristic, such as the Adaptive Large Neighborhood Search (ALNS) scheme proposed by Ropke and Pisinger [19]. This heuristic has proved very effective on a variety of routing problems (see, e.g. [13,17]).

Acknowledgments

This work was partly supported by The Research Council of Norway through the OPTIMAR and Desimal projects, and by the Canadian Natural Sciences and Engineering Research Council under Grant 39682-10. This support is gratefully acknowledged. Thanks are due to an anonymous referee for several valuable comments.

References

[1] Balas E, Fischetti M. Polyhedral theory for the asymmetric traveling salesman problem. In: Du D-Z, Pardalos PM, Gutin G, Punnen A, editors. The traveling salesman problem and its variations. Dordrecht: Springer; 2004. p. 117–68.
[2] Balas E, Fischetti M, Pulleyblank WR. The precedence-constrained asymmetric traveling salesman polytope. Mathematical Programming 1995;68(3): 241–65.
[3] Battarra M, Erdoğan G, Laporte G, Vigo D. The traveling salesman problem with pickups, deliveries, and handling costs. Transportation Science 2010; 44(3):383–99.
[4] Christiansen M, Fagerholt K, Flatberg T, Haugen Ø, Kloster O, Lund EH. Maritime inventory routing with multiple products: a case study from the cement industry. European Journal of Operational Research 2011;208(1): 86–94.
[5] Cordeau J-F, Iori M, Laporte G, Salazar González JJ. A branch-and-cut algorithm for the pickup and delivery traveling salesman problem with LIFO loading. Networks 2010;55(1):46–59.
[6] Dantzig GB, Johnson DR, Fulkerson SM. Solutions of a large-scale traveling-salesman problem. Operations Research 1954;2(4):363–410.
[7] Erdoğan G, Cordeau J-F, Laporte G. The pickup and delivery traveling salesman problem with first-in-first-out loading. Computers & Operations Research 2009;36(6):1800–8.
[8] Fagerholt K, Christiansen M. A travelling salesman problem with allocation, time window and precedence constraints—an application to ship scheduling. International Transactions in Operational Research 2000;7(3):231–44.
[9] Felipe A, Ortuño MT, Tirado G. The double traveling salesman problem with multiple stacks: a variable neighborhood search approach. Computers & Operations Research 2009;36(11):2983–93.
[10] Gavish B, Graves S. The travelling salesman problem and related problems. Working Paper; OR 078-78. Operations Research Center, Massachusetts Institute of Technology, 1978.
[11] Hennig F, Optimization in maritime transportation: crude oil tanker routing and scheduling. PhD thesis. Norwegian University of Science and Technology; 2010.
[12] Iori M, Martello S. Routing problems with loading constraints. TOP 2010;18(1):4–27.
[13] Korsvik JE, Fagerholt K, Laporte G. A large neighbourhood search heuristic for ship routing and scheduling with split loads. Computers & Operations Research 2011;38(2):474–83.
[14] Miller CE, Tucker AW, Zemlin RA. Integer programming formulation of traveling salesman problems. Journal of the ACM 1960;7(4):326–9.
[15] Öncan T, Altinel İK, Laporte G. A comparative analysis of several asymmetric traveling salesman formulations. Computers & Operations Research 2009; 36(3):637–54.
[16] Padberg MW, Rinaldi G. An efficient algorithm for the minimum capacity cut problem. Mathematical Programming 1990;47(1–3):19–36.
[17] Pepin A-S, Desaulniers G, Hertz A, Huisman H. A comparison of five heuristics for the multiple depot vehicle scheduling problem. Journal of Scheduling 2009;12(1):17–30.
[18] Petersen HL, Madsen OBG. The double travelling salesman problem with multiple stacks—formulation and heuristic solution approaches. European Journal of Operational Research 2009;198(1):139–47.
[19] Ropke S, Pisinger D. An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows. Transportation Science 2006;40(4):455–72.
[20] Sherali HD, Driscoll PJ. On tightening the relaxations of Miller–Tucker–Zemlin formulations for asymmetric traveling salesman problems. Operations Research 2002;50(4):656–69.
[21] Song J-H, Furman KC. A maritime inventory routing problem: practical approach. Computers & Operations Research, 2010, doi:10.1016/j.cor.2010.10.031.