Exact approaches for the Traveling Salesman Problem with Draft Limits

Maria Battarra
School of Mathematics, University of Southampton
Southampton, SO17 1BJ, UK
Anand Subramanian
Departamento de Engenharia de Produção, Centro de Tecnologia, Universidade Federal da Paraíba
Campus I, Bloco G, Cidade Universitária, 58051-970, João Pessoa, PB
anand@ct.ufpb.br
Artur Pessoa, Eduardo Uchoa
Departamento de Engenharia de Produção, Universidade Federal Fluminense
Rua Passo da Pátria 156, Bloco E - 4^{o}andar, São Domingos, Niterói-RJ, 24210-240
{artur,uchoa}@producao.uff.br

ABSTRACT

This paper proposes exact algorithms for the Traveling Salesman Problem (TSP) with Draft Limits (TSPDL), which is a variant of the well-known TSP arising in the context of maritime transportation where draft limits are imposed due to restrictions on the port infrastructures. The exact algorithms are based on three mathematical formulations and their performances are compared through extensive computational experiments. The column generation based formulation and resulting branch-cut-and-price algorithm outperform the other two exact algorithms and previously proposed methods. Moreover, the open instances from the literature have been solved to optimality.
KEY WORDS. Draft limits. Traveling Salesman. Cutting planes. Column generation.

Main areas: Combinational Optimization. Logistics and Transport. Mathematical Programming.

RESUMO

Este artigo propõe algoritmos exatos para o *Traveling Salesman Problem* (TSP) *with Draft Limits* (TSPDL) que consiste em uma variante do TSP identificada no contexto do transporte marítimo onde limites de capacidade são impostos em virtude de restrições de infraestrutura dos portos. Os algoritmos exatos são baseados em três formulações matemáticas e seus desempenhos são comparados por meio de experimentos computacionais. A formulação baseada em geração de colunas e o algoritmo *branch-cut-and-price* resultante apresentou melhores resultados tanto em relação aos outros dois algoritmos exatos como em relação aos métodos propostos anteriormente. Além disso, as instâncias abertas da literatura foram resolvidas até a otimalidade.
PALAVRAS CHAVE. *Draft limits*. Caixeiro Viajante. Planos de corte. Geração de colunas.

Áreas Principais: Otimização Combinatória. Logística e Transportes. Programação Matemática.

1 Introduction

The Traveling Salesman Problem (TSP) with Draft Limits (TSPDL) is a variant of the well-known TSP, recently introduced by Rakke et al. (2012), that arises in the context of maritime transportation. The sequence of ports that a cargo boat visits in a tour is dependent on the port infrastructures: the sea-level in a port is sometimes not sufficiently deep to accommodate loaded cargo boats. The port is thus associated to a draft limit, such as the maximum vertical distance allowed between the waterline and the bottom of the hull. Note that the draft of a cargo boat depends on the load: the heavier the load, the higher the boat's draft. Therefore, draft limits can be easily translated into restrictions on the maximum load of the boat.

The problem can be formalized as follows: A directed graph $G(V, A)$ is given, where $V = \{0, \ldots, n\}$ is the set of ports to be visited and $A = \{(i, j), i, j \in V, i \neq j\}$ is the arc set, or set of connections between ports. Each arc $(i, j) \in A$ is associated to a routing cost $c_{ij} > 0, i \neq j$. The vertex 0 is the port from which the boat starts and ends its tour, whereas vertices $V' = \{1, \ldots, n\}$ are ports to be visited exactly once. Each port requires the delivery of $d_i, i \in V'$, units of load and is associated to a draft limit $l_i, i \in V'$. The initial load is $Q = \sum_{i \in V'} d_i$ and we denote $\underline{d} = \min_{i \in V'}\{d_i\}$. The boat cannot enter port i if its load is heavier than l_i, or the hull of the boat could be grounded. Therefore the TSPDL asks for the minimum cost Hamiltonian tour, visiting each port exactly once and not violating draft limit constraints.

Despite its simple definition, the TSPDL proved to be hard to solve to optimality. The problem, as a generalization of the TSP, is $\mathcal{NP}$-Hard and combines the complexity of a TSP with that of a scheduling problem: draft limits implicitly define precedence constraints along the tour.

Rakke et al. (2012) proposed two mathematical formulations: the first formulation makes use of the binary variables x_{ij} assuming value 1 if arc (i, j) is in the solution, and continuous variables y_{ij}, representing the load on the boat while travelling arc (i, j). The resulting formulation is compact, but provides poor quality bounds. The second formulation includes two additional sets of variables: u_j and t_{ij} specifying the position of port j and of arc (i, j) in the circuit, respectively. These two sets of variables allow for the introduction in the model of the Miller, Tucker, and Zemlin constraints (MTZ)(Miller et al., 1960). The MTZ constraints, usually employed to avoid subtours, are used to strengthen the formulation and they are included at the root node of the branch-and-bound tree.

Both formulations have been further strengthened by dynamically separating subtour elimination constraints (Dantzig et al., 1954), as well as their lifted counterpart (Balas and Fischetti, 2004). Moreover, lower bounds on the u_j variables and lower bounds on the sum of y_{ij} variables are imposed.

The branch-and-cut algorithms originating from both formulations are capable of solving quite effectively instances with a limited amount of ports with draft limits, but, when the percentage of ports with a draft limit increases, the algorithm struggles even for medium-sized instances. The problem seems therefore challenging and, as far as we are aware, no other attempts have been made to solve it exactly. This motivated our interest in the problem and we decided to investigate alternative formulations and solution techniques.

Three alternative mathematical formulations are introduced (see Section 2). The first formulation is based on two-index variables, the second formulation is based on three-index variables, whereas the third can be viewed as an improvement over a Dantzig Wolfe decomposition of the second, including the concept of ng-routes (Baldacci et al., 2011), and it is solved through a branch-cut-and-price algorithm.

In Section 3, a description of the branch-and-cut and branch-cut-and-price implementations are presented. The results of our computational experience are summarized in Section 4, and Section 5 presents conclusions and future possible research directions.

2 Mathematical Formulations

The first formulation we propose, denoted F1, is composed of two sets of two-indexed binary variables. The x_{ij} variable assumes value 1 if $(i,j) \in A$ is in the solution, 0 otherwise. The variable y_{ik} assumes value 1 when the boat enters port $i \in V'$ carrying $k \in \{d_i, \ldots, l_i\}$ units of load, 0 otherwise. Note that arcs $(i,j)|d_j > l_i - d_i$ can be removed from the network: in order to simplify the notation we do not explicitly remove these arcs, but it is sufficient to disregard the corresponding variables in our models to take this aspect into account. The formulation F1 can be stated as follows:

$$\text{(F1)} \ \min \sum_{(i,j)\in A} c_{ij}x_{ij} \tag{1}$$

$$\text{s.t.} \sum_{j\in V, j\neq i} x_{ji} = 1, \ \forall i \in V \tag{2}$$

$$\sum_{j\in V, j\neq i} x_{ij} = 1, \ \forall i \in V \tag{3}$$

$$\sum_{i\in V'|l_i\geq k} y_{ik} \leq 1, \ \forall k \in \{\underline{d}, \ldots, Q - \underline{d}\} \tag{4}$$

$$\sum_{k=d_i}^{l_i} y_{ik} = 1, \ \forall i \in V' \tag{5}$$

$$y_{iQ} = x_{0i}, \ \forall i \in V'|l_i = Q \tag{6}$$

$$y_{id_i} = x_{i0}, \ \forall i \in V' \tag{7}$$

$$x_{ij} + y_{ik} \leq 1 + y_{j,k-d_i}, \ \forall (i,j) \in A, k \in \{d_j + d_i, \ldots, \min\{l_i, l_j + d_i\}\} \tag{8}$$

$$x_{ij} \in \{0,1\}, \ \forall (i,j) \in A \tag{9}$$

$$y_{ik} \in \{0,1\}. \ \forall i \in V', k \in \{d_i, \ldots, l_i\} \tag{10}$$

The Objective Function (1) aims at minimizing routing costs, Constraints (2) and (3) are the degree constraints. Constraints (4) impose that the boat visits at most a port for each intermediate load value. Constraints (4) are not necessary to define the optimal integer solution, but they proved to strengthen the linear relaxation of F1. Therefore we included them in the formulation. Constraints (5) state that each port has to be assigned a load. The first and last position of the tour are imposed to be connected to the initial port 0 (Constraints (6) and (7), respectively). Constraints (8) link variables x_{ij} and y_{ik}: if arc (i,j) is traversed, $x_{ij} = 1$ and i and j are located in consecutive positions of the tour. Therefore summing variables x_{ij}, y_{ik} can result in a value equal to 2 only if $y_{j,k-d_i} = 1$. These constraints generalize similar constrains encountered in single machine scheduling problems (an interested reader can refer to the models based on assignment and positional date variables in Keha et al., 2009). Finally Constraints (9) and (10) define the binary nature of the variables.

Formulation F1 presents similarities with the MTZ-based formulation for the *Asymmetric TSP* (ATSP) (see Roberti and Toth, 2012 for a recent overview and comparison of ATSP models), but consists only of binary variables.

Finally, F1 is strengthened by incorporating the trivial constraints

$$x_{ij} + x_{ji} \leq 1, \forall (i,j) \in A \tag{11}$$

and by separating in a cutting plane fashion the subtour elimination constraints:

$$\sum_{i\in S}\sum_{j\in\bar{S}} x_{ij} \geq 1. \quad \forall S \subseteq V' \tag{12}$$

For this latter set of inequalities, the exact separation can be done in polynomial time.

Formulation F2 considers three-indexed binary variables z_{ij}^k, assuming value 1 if arc $(i,j) \in A$ is visited by the boat carrying k units of load (including the demand of port j). By denoting $K_{ij} = \min\{l_j, l_i - d_i\}$, formulation F2 is:

$$\text{(F2)} \quad \min \sum_{(i,j)\in A}\sum_{k=d_j}^{K_{ij}} c_{ij} z_{ij}^k \tag{13}$$

$$\text{s.t.} \sum_{\substack{i\in V\\ i\neq j}}\sum_{k=d_j}^{K_{ij}} z_{ij}^k = 1,\ \forall j \in V \tag{14}$$

$$\sum_{\substack{j\in V|j\neq i\\ l_j\geq k+d_j}} z_{ji}^k - \sum_{\substack{j\in V|j\neq i\\ l_j\geq k-d_i, d_j\leq k-d_i}} z_{ij}^{k-d_i} = 0,\ \forall i \in V, k \in \{d_i,\ldots,l_i\} \tag{15}$$

$$z_{i0}^k = 0,\ \forall i \in V', k \in \{1,\ldots,Q\} \tag{16}$$

$$z_{0j}^k = 0,\ \forall j \in V' | l_j = Q, k < Q, k \in \{d_j,\ldots,l_j\} \tag{17}$$

$$z_{ij}^k \in \{0,1\}.\ \forall (i,j) \in A, k \in \{d_j,\ldots,K_{ij}\} \tag{18}$$

The Objective Function (13) minimizes routing costs. Constraints (14) are the degree constraints. Constraints (15) preserve the load conservation. Constraints (16) and (17) force the boat to return to the depot empty and leave the depot carrying Q units, respectively. Constraints (18) define the nature of the variables. Formulation F2 is similar to the three-index formulations proposed in Fox et al. (1980), for the *Time Dependent TSP*.

Constraints

$$\sum_{\substack{j\in V\\ j\neq i}}\sum_{k=d_j}^{K_{ij}} z_{ij}^k = 1, \forall i \in V \tag{19}$$

are implied by Constraints (14) and Constraints (15), as previously stated in Pessoa et al. (2008).

F2 can be strengthened by the trivial constraints:

$$\sum_{k=d_j}^{K_{ij}} z_{ij}^k + \sum_{k=d_i}^{K_{ji}} z_{ji}^k \leq 1, \quad \forall (i,j) \in A \tag{20}$$

that have been included in the formulation *a priori*.

Flow conservation constraints ensure that subtours are avoided for F2 integer solutions, however we strengthened the formulation by including the subtour elimination constraints as cutting planes (as for F1):

$$\sum_{i\in S}\sum_{j\in\bar{S}}\sum_{k=d_j}^{K_{ij}} z_{ij}^k \geq 1, \quad \forall S \subseteq V' \tag{21}$$

Finally, F2 can be strengthened by adding the *2-cycle* inequalities:

$$z_{ij}^k \leq \sum_{t\in V, t\neq\{i,j\}, t|l_t\geq k-d_j} z_{jt}^{k-d_j}, \quad \forall (i,j) \in A, i,j \neq 0, \forall k = \{d_j,\ldots,K_{ij}\} \tag{22}$$

as proposed in Abeledo et al. (2013). These constraints have been added to F2 in a cutting-plane fashion, using an exact enumerative separation procedure. Note that the constraints can be avoided for the arcs incident in the depot, because Constraints (16) and (17) ensure that no arcs return to the depot after visiting the first port and no arcs would visit a port after the depot.

Proposition 1. *The linear relaxation of F2 dominates the linear relaxation of F1.*

Proof. A $\overline{z_{ij}^k}$ solution of F2 can be converted into an F1 solution $(\overline{x_{ij}}, \overline{y_{jk}})$ with the same cost, considering the following transformation:

$$\begin{cases} \overline{x_{ij}} = \sum_{k=d_j}^{K_{ij}} \overline{z_{ij}^k} & \forall (i,j) \in A \\ \overline{y_{ik}} = \sum_{\substack{j \in V | j \neq i \\ l_j \geq k + d_j}} \overline{z_{ij}^k}, & \forall i \in V', k = \{d_i, \ldots, l_i\} \end{cases}$$

It is straightforward that, given $(\overline{x_{ij}}, \overline{y_{jp}})$, Constraints (2), (3) and (5) are implied by constraints in F2, in fact, once converted into $\overline{z_{ij}^k}$ variables, they correspond to Constraints (14), (19), and (14), respectively.

According to Constraints (17), Q units of load are shipped from the origin port and, from Constraints (16), zero units return to the same port. Moreover, Constraints (15) guarantee that the load monotonically decreases in the tour cycle defined by Constraints (14) and (15). Hence, Constraints (4),(6) and (7) are satisfied by F2.

In the following we proof that Constraints (8) are satisfied by F2: we first express the constraints in term of z_{ij}^k variables:

$$x_{ij} + y_{ik} = \sum_{p=d_j}^{K_{ij}} z_{ij}^p + \sum_{\substack{s \in V | s \neq i \\ l_s \geq k + d_s}} z_{si}^k.$$

Given Constraints (15), we can write:

$$\sum_{p=d_j}^{K_{ij}} z_{ij}^p + \sum_{\substack{s \in V | s \neq i \\ l_s \geq k + d_s}} z_{si}^k = \sum_{p=d_j}^{K_{ij}} z_{ij}^p + \sum_{\substack{s \in V | s \neq i \\ l_s \geq k - d_i, d_s \leq k - d_i}} z_{is}^{k - d_i}$$

The latter term can be overestimated by

$$\sum_{p=d_j}^{K_{ij}} z_{ij}^p + \sum_{\substack{s \in V | s \neq i \\ l_s \geq k - d_i, d_s \leq k - d_i}} z_{is}^{k - d_i} \leq \sum_{s \in V, s \neq i} \sum_{p=d_s}^{K_{is}} z_{is}^p + z_{ij}^{k - d_i}.$$

Given Constraints (19)

$$\sum_{s \in V, s \neq i} \sum_{p=d_s}^{K_{is}} z_{is}^p + z_{ij}^{k - d_i} = 1 + z_{ij}^{k - d_i}.$$

This can be overestimated by

$$1 + z_{ij}^{k - d_i} \leq 1 + \sum_{\substack{s \in V | s \neq j \\ l_s \geq k - d_i + d_s}} z_{sj}^{k - d_i} = 1 + y_{j, k - d_i}.$$

Therefore, Constraints 8 are satisfied.

On the other hand, the linear relaxation of F1 allows for disconnected subtours with at least three vertices of lower cost than the corresponding integer connected solution, while F2 ensures connectivity because of Constraints (15). Therefore the feasible region of the linear relaxation of F2 is a subset of the feasible region of the linear relaxation of F1.

□

A useful relaxation of the problem is obtained by removing (14) from F2. In this case, a feasible solution to (15)-(18) is also a tour starting at the port 0 with load Q and ending at the same port with load 0, respecting the draft limits. Nevertheless, such tour does not have to visit all ports, while other ports may be visited more than once. Optimizing this relaxation can be done in $O(n^2Q)$ time by a dynamic programming procedure, which suggests the following reformulation of F2. Let P be a set of all paths defined above. For each $p \in P$, we introduce the binary variable λ_p indicating if p is used or not in the solution. Define q_{ij}^{kp} as a binary coefficient indicating whether variable z_{ij}^k is associated to path p.

$$\text{Minimize} \quad \sum_{p \in P} \Big(\sum_{i \in V} \sum_{\substack{j \in V \\ j \neq i}} \sum_{k=d_j}^{K_{ij}} q_{ij}^{kp} c_{ij} \Big) \lambda_p \tag{23}$$

S.t.

$$\sum_{p \in P} \Big(\sum_{\substack{j \in V \\ j \neq i}} \sum_{k=d_j}^{K_{ij}} q_{ij}^{kp} \Big) \lambda_p = 1, \quad \forall i \in V \tag{24}$$

$$\lambda_p \in \{0,1\} \qquad \forall p \in P. \tag{25}$$

Cuts over the z_{ij}^k variables can be translated to the λ_p variables by applying the equality

$$\sum_{p \in P} q_{ij}^{kp} \lambda_p = z_{ij}^k,$$

as already shown when deriving (24) from (14).

The linear relaxation of this reformulation can be efficiently solved by column generation. Significantly stronger linear relaxations can be obtained by forbidding some paths in P that visit some ports more than once. One alternative is to avoid returning to a port i before visiting at least s ports other than i (Irnich and Villeneuve, 2006), thus eliminating tours with s-cycles.

Instead, we use the following strategy that has already been proved to be more efficient in practice (Baldacci et al., 2011). For each customer $i \in V'$, let $N_i \subseteq V'$ be the ng-set of i, defining its neighbourhood. This may stand for the $|N_i|$ closest ports and includes i itself. The cardinalities of all ng-sets are the same, denoted by T. An ng-path is a path as defined in P where every cycle that starts and ends at a vertex i must contain at least one vertex j such that $i \notin N_j$. This can be interpreted as if the boat "forgot" the visit to i when passing by the port j. In our implementation, we define a different ng-set N_i^k for each load k of the boat after visiting each port i. The neighbors of port i with load k are then selected as the T closest ports that can be visited immediately before i without violating the draft limits. This modification strengthens the relaxation since original ng-sets spend part of their memory to store visits to ports that would violate the draft limits. The same pricing algorithm can be used to handle these ng-sets. To the best of our knowledge, this is the first time that ng-sets depending on the load are considered.

Formulation F3 is then defined as (23)-(25) plus all inequalities described in Abeledo et al. (2013) translated to the λ_p variables.

3 Algorithms Description

The branch-and-cut algorithms for formulations F1 and F2 have been implemented using the Callable Libraries of CPLEX, and, the subtour elimination constraints (12) and (21) have

been separated using the CVRPSEP Library (Lysgaard, 2003). The subtour constraints have been separated only up to the seven-th level of the branching tree to speed up computation. Moreover, at most 50 subtour cuts have been added per iteration. The 2-cycle inequalities (22) were separated only at the root node. CPLEX strong branching was chosen for branching and all CPLEX cuts have been used for strengthening the formulation.

The branch-cut-and-price algorithm for formulation F3 is an adaptation of the one proposed by Abeledo et al. (2013). We enhance it by replacing the 5-cycle elimination in the pricing problem with the ng-sets described above. When increasing the value of T, the lower bounds improve but the pricing becomes slower. In our case, $T = 12$ offers a reasonable compromise between speed and lower bound quality. The method starts by only considering constraints (24) and iteratively adds the remaining inequalities on demand using the separation procedures described by Abeledo et al. (2013). Furthermore, we branch over the x_{ij} variables translated from the z_{ij}^k variables using

$$x_{ij} = \sum_{k=d_j}^{K_{ij}} z_{ij}^k.$$

Given a relaxed solution $\bar{x}$, we choose a pair of ports (i, j) such that $\bar{x}_{ij} + \bar{x}_{ji}$ is strictly between 0 and 1 and add the constraint $x_{ij} + x_{ji} \leq 0$ in one branch and $x_{ij} + x_{ji} \geq 1$ in the other. These cuts are translated back to the z_{ij}^k variables and then to the λ_p variables. In the selection of the best pair (i, j), we try to maximize the absolute difference between $\bar{x}_{ij} + \bar{x}_{ji}$ and a target value 0.6 by choosing the pair that maximizes

$$\min\{(\bar{x}_{ij} + \bar{x}_{ji})/0.6, (1.0 - \bar{x}_{ij} - \bar{x}_{ji})/0.4\}.$$

4 Computational Results

The instances considered in our extensive computational testing belong to the benchmark set proposed by Rakke et al. (2012). The problems are adaptations of instances from the TSPLIB (namely *burma14*, *ulysees16*, *ulysses22*, *fri26*, *bayg29*, *gr17*, *gr21*, and *gr48*), with number of vertices ranging between 14 and 48. In the following, the acronym *a_b_c* refers to the c^{th} instance, adapted from the TSPLIB problem *a* with *b*% of ports with draft limit smaller than Q.

For each TSP instance, problems with 10%, 25% and 50% of the ports having a draft limit smaller than Q have been generated. More precisely, given each TSP instance and each percentage of ports with drafts smaller than Q, ten instances have been proposed. Drafts smaller than Q have been randomly generated between 1 and $n - 1$. The instances are available at `http://jgr.no/tspdl`.

Our tests were performed on an Intel Core i5 with 3.2 GHz and 4 GB of RAM, running Ubuntu Linux 10.04. A two-hours time limit was imposed and a single thread used throughout the computational experience.

In the following, we compare the root node lower bounds of F1, F2, and F3 (including or not additional constraints). For each $Instance$ and each formulation, the percentage deviation of lower bound z with respect to the optimal solution z_{opt} is computed as:

$$Dev(\%) = 100 \times (1 - \frac{z}{z_{opt}}).$$

The computing time in seconds is reported in the columns $Time(sec)$.

Aggregated results of the root node for each instance set are provided in Table 1, where for each set of instances the average percentage deviation of the lower bounds from the optimal solutions ($Dev(\%)$) and average computing time in seconds ($Time(sec)$) are provided. Note that the lower bounds are obtained by including subtour elimination constraints (in a cutting plane

Table 1: Root node lower bounds: summary of the average results

	F1+ (12)		F2+ (21)+(22)		F3	
Instance	Dev. (%)	Time (sec)	Dev. (%)	Time (sec)	Dev. (%)	Time (sec)
burma14	2.57	0.44	0.16	0.19	0.00	0.38
ulysses16	1.52	0.72	0.20	0.34	0.00	24.45
ulysses22	5.50	4.26	2.56	2.49	0.00	27.73
fri26	7.90	8.59	2.34	6.79	0.00	15.02
bayg29	7.13	18.99	2.65	13.38	0.00	12.14
gr17	2.90	0.98	0.22	0.41	0.00	13.92
gr21	7.37	2.29	2.05	1.22	0.00	11.13
gr48	11.24	606.14	5.05	274.72	0.84	125.22
Avg.	5.77	80.30	1.90	37.44	0.11	28.75

fashion) in F1 and F2, a well as Constraints (22) for F2, because these constraints proved to be necessary to achieve good quality lower bounds for both algorithms.

F2 with Constraints (21) and (22) provides lower bounds that are on average between 0.16% and 5.05% from the optimal solutions, whereas F1 with (12) provides solutions with average gaps from 1.52% to 11.24%. Moreover F2 is roughly five times faster than F1. F3 provides even more competitive lower bounds: all set of instances but some of the $gr48$ have been solved to optimality at the root node, in less than roughly 30 seconds on average. The maximum percentage deviation for the lower bounds is achieved in the instances with 48 vertices and it is 0.48%.

In terms of optimal solutions achieved, we compare the best bounds provided by Rakke et al. (2012) with those obtained by F1 with (12), F2 with (21) and (22) and F3. The experiments conducted by Rakke et al. (2012) had been performed on a HP dl160 G3 computer with 2x3.0GHz Intel E5472 Xeon CPU and 16GB of RAM. The solver used was Xpress-MP 7.2 and the separation routine for the subtour elimination constraints was coded in C using the Mosel 3.2.0 callbacks. Finally, the algorithm has been executed using parallel processing on the 8 cores available on the machine and for at most 10000 seconds. Our testing has been performed on a more modest computer, using a single core and a 7200 seconds time limit, therefore our results are not directly comparable to those of Rakke et al. (2012).

A summary of the results can be found in Table 2, where the number of optimal solutions achieved for each benchmark set if given. Even if the number of optimal solutions obtained by Rakke et al. (2012) is higher than that of F1, it is interesting to notice that F1 is not dominated. The $ulysses22\_50\_4$ instance is an example of problem in which F1 outperforms Rakke et al. (2012) algorithms, even if executed on a slower computer and for a shorter amount of time. F2 is capable of solving 214 out of the 240 instances in the set, but only 4 of the $gr48$ instances. F3 solves to optimality all instances in the set. Detailed results for each instance are available at `http://www.optimization-online.org/DB_HTML/2013/02/3790.html`.

5 Conclusions

In this paper, we presented three exact algorithms for the TSPDL. The first algorithm is a branch-and-cut and it is based on a compact formulation in which two sets of two-index binary variables and a polynomial number of constraints are employed. This formulation, when strengthened with subtour elimination constraints and trivial constraints is not empirically dominated by the best formulation proposed by Rakke et al. (2012). The second algorithm is also a branch-and-cut, but the underlying formulation considers three-index variables. This formulation is proven to dominate the previous one, both theoretically and empirically. The third algorithm is a branch-cut-and-price and it is based on a path interpretation of the second formulation. Columns are generated by dynamically introducing ng-paths to the formulations. The latter algorithm was

Table 2: Optimal solutions: summary of the number of optimal solutions achieved

	Rakke et al. (2012)	F1+ (12)	F2+ (21) + (22)	F3
burma14	30	30	30	30
ulysses16	30	30	30	30
ulysses22	27	17	30	30
fri26	23	13	30	30
bayg29	19	14	30	30
gr17	30	30	30	30
gr22	30	26	30	30
gr48	7	1	4	30
Total	196	161	214	240

capable of solving to optimality all the benchmark instances from the literature.

6 Acknowledgements

The authors would like to thank Prof. Marielle Christiansen and Dr. Jørgen Glomvik Rakke for providing their results. This work was partially supported by Santander Universities UK, UoS Internationalisation Fund, by CORMSIS, Centre of Operational Research, Management Sciences and Information Systems, and Brazilian research agencies CNPq, CAPES and FAPERJ.

References

Abeledo, H., Fukasawa, R., Pessoa, A., Uchoa, E., 2013. The time dependent traveling salesman problem: polyhedra and algorithm. Mathematical Programming Computation. Accepted for publication.

Balas, E., Fischetti, M., 2004. Polyhedral theory for the asymmetric traveling salesman problem. In: Gutin, G., Punnen, A., Du, D.-Z., Pardalos, P. (Eds.), The Traveling Salesman Problem and Its Variations. Combinatorial Optimization. Springer US, pp. 117–168.

Baldacci, R., Mingozzi, A., Roberti, R., 2011. New route relaxation and pricing strategies for the vehicle routing problem. Operations Research 59, 1269–1283.

Dantzig, G., Fulkerson, R., Johnson, S., 1954. Solution of a large-scale traveling-salesman problem. Operations Research 2, 393–410.

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

Irnich, S., Villeneuve, D., 2006. The shortest-path problem with resource constraints and k-cycle elimination for $k \geq 3$. INFORMS Journal on Computing 18, 391–406.

Keha, A., Khowala, K., Fowler, J., 2009. Mixed integer programming formulations for single machine scheduling problems. Computers & Industrial Engineering 56, 357–367.

Lysgaard, J., 2003. CVRPSEP: A package of separation routines for the capacitated vehicle routing problem. Working paper.

Miller, C., Tucker, A., Zemlin, R., 1960. Integer programming formulation of traveling salesman problems. Journal of the ACM 7, 326–329.

Pessoa, A., Poggi de Aragão, M., Uchoa, E., 2008. The Vehicle Routing Problem: Latest Advances and New Challenges. Springer US, Ch. Robust BCP algorithms for Vehicle Routing Problems, pp. 297–325.

Rakke, J., Christiansen, M., Fagerholt, K., Laporte, G., 2012. The traveling salesman problem with draft limits. Computers & Operations Research 39, 2161–2167.

Roberti, R., Toth, P., 2012. Models and algorithms for the asymmetric traveling salesman problem: an experimental comparison. EURO Journal of Transportation and Logistics 1, 113–133.