A branch-and-cut algorithm for a vehicle routing problem with capacity and time constraints[1]

Carlos E. Ferreira	Marko Loparic
Dept. Computer Science	CORE
University of São Paulo	Université Catholique de Louvain
Brazil	Belgium
`cef@ime.usp.br`	`loparic@core.ucl.ac.be`

Abstract In this work we investigate a real world application of a vehicle routing problem, involving the transportation of workers to oil extraction platforms located in the sea. We developed a branch-and-cut algorithm for the problem, which provided a guarantee for the heuristic solutions that we found and for those provided by other works on the same application. We were able to improve the existing solutions, as well to provide good lower bounds for the optimal solutions.

Keywords integer programming, branch-and-cut, vehicle routing.

1 Introduction

In this paper we investigate a vehicle routing problem that arises in the transportation of workers to sea platforms. The Brazilian State Oil Company (Petrobrás) has a set of sea platforms for the oil extraction in the coast of Rio de Janeiro. Ships are to be used to transport workers between the continent and these platforms. A related transportation problem is the application investigated.

This application has been studied by Brinati et al. (1988b), who developed a logistic study for the company. The authors presented solutions for the problem obtained by heuristic procedures. Our aim in this paper is to revisit the application, using a branch-and-cut approach. This approach can be used to provide not only good solutions but also good lower bounds on the optimum, which provide us a quality guarantee for the solutions found.

In this application, as described later, beside the capacity constraint, frequent in vehicle routing problems, a time bound restriction is also present. This restriction (which is also very common in real world applications) turns the problem particularly difficult to deal with when using a branch-and-cut approach. For this reason we were not able to use the nice polyhedral studies and strong cutting plane approaches proposed in Campos, Corberán, and Mota (1991), Cornuéjols and Harche (1993), and Araque, Kudva, Morin, and Pekny (1994).

[1]This work has been partially supported by project ProComb (CNPq No. 680065/94). The first author has been partially supported by CNPq (Proc. No. 300752/94-6

The development of our algorithm was initially based on the experience of Laporte et al. (1985), who also developed a branch-and-cut algorithm for the vehicle routing problem with capacity and time constraints (the authors actually used a distance constraint, but we will refer to it as time constraint since they are very similar). Next we implemented several original features, focussing particularly on the time constraints. Our main goal in this paper is to investigate a real world application and how the branch-and-cut approach can help us to solve real problems.

The paper is organized as follows. In Section 2 we describe the application and provide some results obtained by Brinati et al. In Section 3 the problem is modeled by means of an integer linear program. The approach used to solve the problem is described in Section 4, where the main separation routines and other implementations details are closely discussed. The computational results and comparisons with previous work from Brinati et al. and Laporte et al. are provided in Section 5.

2 The application

Petrobrás has a set of platforms constructed off the sea near Campos, a city located in the southeast of the Brazil. The work in these platforms is done by four crews for each platform. Each crew works in a twelve hours shift, so that the work can be done continually by two crews. Meanwhile the two other crews are in the continent. Each crew stays in the platform 14 days, and every week a crew is replaced. This exchange is done in a fixed day of the week for each platform.

The transportation of the crews between the platforms and continent is done by a fleet of several ships. Each ship can make only one trip per day. In a single trip the crew of several platforms can be exchanged, provided that the *capacity* and *time* constraints are respected. Namely the total number of workers in the ship is limited by its fixed capacity and the total time of the trip must not exceed a certain time bound. The time of a trip is not only given by the travelling time (i.e. the time spent by travelling between platforms), but it also includes a waiting time, which is proportional to the crew size of each platform visited. The rate of passagers transferred per unit of time (*transfer rate*) depends on the transfer system used.

The transportation problem is then to define a set of trips so that the total number of ships needed to form the fleet is minimized. Among solutions minimizing this number, a solution minimizing the total distance (or equivalently the total time) is to be found. The capacity of the ship, its speed (needed for the calculation of the travelling time), the time bound and the transfer rate are fixed for each instance of the problem.

In Brinati et al. (1988) the authors present results of their algorithm applied to a set of instances similar to those we present, but the instances and solutions found are not given. We are very grateful to Prof. Marco Brinati who made us available the description of the instances we have studied, as well as the solutions that he together with his coauthors obtained.

The instances share the same set of points, presented in Table 1, and the ship capacity of 250. The differences refer to the velocity of the ship, the transfer rate and the time bound, as shown in Table 2. The solutions obtained by Brinati et al. appear in Tables 3 to 5. It is not indicated in the table the number of ships needed for realizing the transportation. But, since each ship can make only one trip per day, this number is easily given by $\lceil \frac{\text{number of trips}}{7} \rceil$. Thus, in case of instance A, three ships are needed, and in instances B and C, two ships ares suficient.

3 The vehicle routing problem

We now formulate the vehicle routing problem. We are given:

- n: number of points, including the base and the platforms.
- d_{ij}: vector of distances for each pair of points (i, j) with $i < j$.
- C: capacity of the ship.
- c_i: crew size of each platform.
- T: time bound.
- μ: velocity of the ship.
- ρ: transfer rate (number of passagers exchanged by unit of time).

In the vehicle routing literature the base is usually referred to as *depot*, the platforms as *clients*, and the ships as *vehicles*. We use the fact that we are dealing with an Euclidian problem, i.e. the distances obey the triangular inequality.

Now consider a complete graph $G = (V, E)$, where $V = \{1, ..., n\}$ is the set of points. Given a set $W \subseteq V$, we define $E(W) := \{e = uv \in E \,|\, u \in W \text{ and } v \in W\}$. The node 1 is assumed to be the depot. For $v \in V$ and $W \subseteq V$ we define $\delta(W) := \{wu | w \in W \text{and } u \notin W\}$, $\delta(v) := \delta(\{v\})$ and $E(v : W) := \{vw | w \in W\}$. We partition the set of edges E in $E_1 := \delta(1)$ and $E_2 := E \setminus \delta(1)$.

A *trip* is either a circuit containing the base or a walk of the kind (e, e) where $e \in E_1$ (corresponding to a trip to a single platform). The *length* of a trip $(e_1, \ldots, e_t)$ is the sum $d := \sum_{i=1}^{n} d_{e_i}$. A *routing* for $W \subseteq V \setminus \{1\}$ is a set of trips with each $v \in W$ belonging to exactly one trip such that each trip satisfies the capacity and time constraints. The length of the routing is given by the sum of the lengths of its trips. Note that the routing for W must satisfy $c(W) \leq C$ (the capacity constraint) and $\frac{d}{\mu} + \frac{c(W)}{\rho} \leq T$ (the time constraint), where d is the length of the routing. Given a set $W \subseteq V$ we denote by $r(W)$ the minimum number of trips among all routings for W.

We consider now the problem of finding a routing minimizing the total length. Considerations regarding the optimization of the number of ships, which is the primary goal of our application are given in Section 5.

An integer formulation of the problem follows. The variable t corresponds to the number of trips. The variable $x \in \{0, 1\}^E$ represents a routing R in the following way:

	x	y	crew size		x	y	crew size
1	0	0		26	-5019	139734	145
2	-33720	96469	30	27	-5019	139734	40
3	-35704	102125	40	28	-3590	147120	100
4	-37510	100790	30	29	-3692	138424	145
5	-39736	98601	50	30	-3692	138424	40
6	-46233	103076	60	31	-3119	132670	165
7	-43843	104110	140	32	-3119	132670	40
8	-43843	104110	100	33	1318	125784	25
9	-43843	104110	40	34	3390	139109	120
10	-43843	104110	40	35	7976	153264	55
12	-39524	112157	55	36	2005	168645	45
12	-39514	112132	50	37	14451	173070	50
13	-38690	110844	50	38	17523	166838	70
14	-34770	108430	45	39	25785	184957	35
15	-33837	111111	175	40	34075	190644	35
16	-33836	111111	40	41	31848	183202	55
17	-36451	114689	35	42	27250	153167	35
18	-33955	115222	50	43	27250	153167	45
19	-30959	120053	40	44	25592	151719	45
20	-29054	123778	45	45	25592	151868	45
21	-29770	126170	50	46	23992	150669	45
22	-18761	127980	55	47	16675	147986	100
23	-8108	129288	40	48	19026	139160	45
24	-6749	133914	165	49	17735	136322	45
25	-6749	133914	40	50	63494	123096	30

Table 1: Coordinates of the points and crew size of the platforms.

instance	time bound (hours)	speed (knots)	transfer rate (workers per hour)
A	10	25	60
B	12	23	60
C	12	20	360

Table 2: Instances of the problem

$$x_e = \begin{cases} 1, & \text{when } e \text{ appears once in } R; \\ 2, & \text{when } e \text{ appears twice in } R \text{ (in a trip involving a single platform);} \\ 0, & \text{when } e \text{ does not appear in } R. \end{cases}$$

$$\begin{array}{lllr} \text{minimize} & \sum_{e \in E} d_e x_e & & \\ \text{such that} & x(\delta(1)) = 2t, & & (1) \\ & x(\delta(v)) = 2, & \text{for all } v \in V \setminus \{1\}, & (2) \\ & x(E(W)) \leq |W| - r(W), & \text{for all } W \subseteq V, \setminus \{1\}, & (3) \\ & x_e \in \{0,1,2\}, & \text{for } e \in E_1, & (4) \\ & x_e \in \{0,1\}, & \text{for } e \in E_2, & (5) \\ & t \in \mathbb{N}. & & (6) \end{array}$$

It is easy to show that x corresponds to a routing if and only if x satisfies the restrictions (1) to (6). Note that the capacity and time constraints appear only implicitly in the restrictions (they determine the value of $r(W)$ that appears in constraints (3)).

4 The branch-and-cut algorithm

We describe here the outline of our approach. Branch-and-cut algorithms are a combination of a cutting plane with a branch-and-bound algorithm. For a more detailed description of the method and its practical issues see Jünger, Reinelt and Thienel (1995).

The algorithm works as follows. Initially we construct a linear program, by means of inequalities (1) and (2) of the above formulation and the integrality constraints (4) and (5) substituted by $0 \leq x_e \leq 2$ for $e \in E_1$ and $0 \leq x_e \leq 1$ for $e \in E_2$. Then an lp solver provides a solution $\overline{x}$. If this point corresponds to a routing for V it is an optimal solution. Otherwise valid inequalities violated by $\overline{x}$ are added to the linear program and the lp solver is called again. This procedure is repeated until an optimal solution is given or no more violated inequalities can be found.

This cutting plane procedure is executed at each node of a branch-and-bound algorithm. At each node the lower bound of the corresponding problem is given by the objective function of the last linear program solved. An upper bound is also calculated using a heuristic procedure to find a routing based on the lp solutions obtained. This procedure is run after each call to the lp solver.

trip	route	crew size	length (km)	time
1	01 30	195	282,300	9h20’
2	06 07	240	225,930	8h52’
3	04 05 08 09 02	230	233,363	8h52’
4	15 16 10 11 12	230	243,520	9h05’
5	13 14 03	250	235,205	9h14’
6	17 18 19 20 21	240	272,990	9h53’
7	22 23	205	268,447	9h12’
8	24 25 26	225	279,979	9h47’
9	27 34	155	313,731	9h21’
10	31 28 29	225	276,962	9h43’
11	33 48	165	291,234	9h02’
12	35 37	115	352,035	9h31’
13	32 36 38	110	378,036	9h59’
14	39 40	90	387,382	9h52’
15	43 41 42 44	170	312,179	9h34’
16	47 45 46	190	309,705	9h51’
17	49	30	277,013	6h29’
total length: 4940,020 km				
total time: 157h47’				

Table 3: Solution by Brinati et al. for instance A.

trip	route	crew size	length (km)	time
1	06 07	240	225,930	9h18’
2	05 08 09 04	190	228,751	8h32’
3	02 12 10 11 13	240	238,870	9h36’
4	01 14 15	245	232,984	9h33’
5	17 19 20 18 16 03	250	267,869	10h27’
6	21 23	220	276,829	10h 9’
7	24 25 31	225	280,177	10h19’
8	26 27 34 47	240	318,815	11h29’
9	28 29 22	225	278,162	10h16’
10	30	165	265,413	8h58’
11	32 33 48 49	220	340,028	11h38’
12	35 36 37 42	210	361,164	11h58’
13	38 39 40 41	160	390,523	11h50’
14	43 44 45 46	235	312,726	11h15’
total length: 4018,248 km				
total time: 145h25’				

Table 4: Solution by Brinati et al. for instance B.

trip	route	crew size	length (km)	time
1	06 07	240	225,930	6h46’
2	01 04 03 08 09 05	250	234,421	7h 1’
3	10 11 12 13 02	240	239,637	7h 8’
4	14 15	215	232,298	6h52’
5	16 17 18 19 20	220	264,894	7h45’
6	22 23 24	245	268,447	7h55’
7	21 25 26	240	287,255	8h25’
8	27 33	220	296,939	8h37’
9	48 28 29	230	297,473	8h40’
10	32 30 31	230	266,689	7h50’
11	47 46 34 35	245	344,919	9h59’
12	37 36 38 39 40	245	394,898	11h20’
13	45 43 44 41 42 49	245	342,337	9h55’
total length: 3696,142 km				
total time: 108h18’				

Table 5: Solution by Brinati et al. for instance C.

The most important features of the approach described above are the separation routines, i.e. given a solution to the lp, how can we find a valid inequality violated by this point? Next in this section we describe with more details how work the procedures we use for separation and discuss some relevant aspects of the implementation of the algorithm.

4.1 Separation procedures

The valid inequalities used in our separation procedures are of the type

$$x(\delta(W)) \geq |W| - \overline{r}$$

where $W \subseteq V \setminus \{1\}$ and $\overline{r}$ is a lower bound on $r(W)$. It is clear that inequalities of this type are valid, since they derive from the subtour elimination inequalities, given in the formulation. We used two types of lower bounds, one related to the capacity constraint,

$$r_C(W) = \left\lceil \frac{c(W)}{C} \right\rceil,$$

and another related to the time constraint,

$$r_T(W,d) = \left\lceil \frac{\frac{d}{\mu} + \frac{c(W)}{\rho}}{T} \right\rceil,$$

where d is a lower bound for the length of a routing for W. Note that $\frac{d}{\mu}$ is a lower bound for the travelling time and $\frac{c(W)}{\rho}$ is the waiting time of a routing for W. Clearly $r_C(W) \leq r(W)$ and $r_T(W,d) \leq r(W)$.

Our first heuristic separation procedure takes as input subgraph $H_{\overline{x}} = (V, E^*)$ of G, where we have that e belongs to E^* if and only if e is in E_2 and $\overline{x}_e > 0$.

We define the slack

$$z(Y) = x(\delta(Y)) - |Y| + r_C(Y).$$

The idea is to find $Y \subseteq V \setminus \{1\}$ such that $z(Y) < 0$. The set Y_e denotes the set of vertices of the new component of the graph H' obtained with the insertion of edge e.

Procedure 4.1 Input: *graph* $H_{\overline{x}} = (V, E^*)$

Let $H' = (V, \emptyset)$ *be the empty subgraph of* H*;*
while there exists e *in* E^* *connecting different components of* H'*:*
  take such an edge e *minimizing* $z(Y_e)$*;*
  insert e *in* H'*;*
  if $z(Y_e) < 0$
    save Y_e*;*

The time spent by this procedure is given by $O(|V||E^*|)$, which does not exceed $O(n^3)$, but shows in practice the behaviour of a $O(n^2)$, since the number of edges in E^* is rather linear than quadratic on n.

This procedure was written to replace a similar one from Laporte et al. In their procedure the sets Y are taken at the beginning as the (connected) components of graph $H_{\overline{x}}$. Then iteratively a vertex is removed from Y in a way that $z(Y)$ is minimized. We have tested this procudure first, but we substituted it later by the new one presented here, since the algorithm got stuck when some fractional solutions could not be separated. More details can be found in Loparic (1995).

Both procedures deal only with the capacity constraint. To take the time constraint into consideration the authors observed that if $H^* = (V^*, E^*)$ is a component of $H_{\overline{x}}$ then $d^* := \sum_{e \in E^*} x_e$ is a lower bound of the length of a routing for V^*. The authors use the value $r(V^*, d^*)$ to generate cuts in the same way. We did not implement this procedure because these cuts would only take entire components of $H_{\overline{x}}$ into consideration. Since the time constraint play a very important role in our application we decided to write a second procedure which would also investigate subgraphs of these components.

The procedure calculates a lower bound for the length of a routing for a set Y, in order to find violated cuts of the kind $x(E(Y)) < |Y| - r_T(Y, d)$. The algorithm is based on the following proposition.

Definition 4.2 *Given* $Y \subseteq V \setminus \{1\}$*, we say that a subgraph* F *of* $E(Y)$ *is a* k-forest *of* Y *if* F *does not contain circuits and has* k *components. A* k-routing *for* Y *is a routing for* Y *composed of* k *trips.*

Proposition 4.3 *Let $Y \subseteq V \setminus \{1\}$ and k be an integer such that $k \geq r(Y)$. Consider a k-forest F of Y minimizing $\sum_{e \in F} d_e$ and a set F' containing k edges of minimum distance from $E(1:Y)$. Then*

$$\overline{d}(Y) := \sum_{e \in F} d_e + 2 \sum_{e \in F'} d_e$$

is a lower bound for the length of a k-routing for Y.

Proof: Since our vehicle routing problem is Euclidian, we can choose an optimum k-routing R for Y without platforms outside Y. So as R is made of k trips, the set of its edges corresponds to the union of a k-forest and a set of edges from $E(1:Y)$. The edges from $E(1:Y)$ appear $2k$ times in total, and each one of them is used at most twice (recall the an edge is used twice when it is in a trip to a single platform).

■

Based on this proposition, we developed the following procedure.

Procedure 4.4 Input: $Y \subseteq V$

let $k = r_C(Y) - 1$;
repeat
  let k=k+1;
  get F and F' for Y and k as in proposition 4.3
  calculate $\overline{d}(Y)$ for F and F';
until $k \geq r_T(Y, \overline{d}(Y))$

The value of k on termination of the procedure is taken as a lower bound of $r(Y)$. Note that if k reaches the value of $r(Y)$, then, as by the proposition we have that $r_T(Y, \overline{d}(Y)) \leq r(Y)$, the procedure stops. This guarantees the procedure always stops and that at this moment k is less or equal to $r(Y)$.

This procedure is particularly interesting since the sets F and F' can be found quickly. Before the beginning of the branch-and-cut algorithm a list of edges of E_2 is built, sorted by distance. We are able to find in this way the set F in time which slightly exceeds $O(|E|)$ using Kruskal algorithm (see e.g. Aho, Hopcroft and Ullman, 1974). To find F' a similar list is built for the edges of E_1.

This procedure is applied to every set Y generated in the loop of procedure 4.1.

As a next improvement made, an enumerative travelling salesman problem solver is applied to the set $Y \bigcup \{1\}$ everytime we have that $|Y| \leq 6$ and $k = 1$ at the end of the procedure 4.4. The length $\tilde{d}(Y)$ obtained is typically higher than $\overline{d}(Y)$. In many of those cases it happens that $r(Y, \tilde{d}(Y)) > 1$. This additional procedure provides several additional cuts, improving the separation, without increasing significatively its time. Clearly this kind of procedure is useful when the vehicle routing problem is rather of a tight nature, where trips with less than 10 points are frequent. This was the case in our application.

We have also implemented the feature of fixing variables to zero, for which Laporte et al. apply the name of *subtour prevention constraints*. On the other hand, we have not implemented the generation of Gomory cuts and the elimination of ineffective constraints as proposed by the authors.

4.2 Other implementation details of the branch-and-cut algorithm

Besides the separation procedure, several aspects of the design of branch-and-cut algorithm influence its performance. We present here the most relevant ones.

The heuristic procedure builds a solution choosing edges from $H_{\overline{x}}$ (see the definition in Section 4), where $\overline{x}$ is the solution given by the lp solver. The edges are greedily selected so that the ones with the highest $\overline{x}_e$ come first. Edges with the same value of $\overline{x}_e$ are randomly chosen to increase the number of different solutions provided by the several calls to the procedure.

The branching is based on fixing the value of a fractional variable to 0 or 1. Several criteria for the choice of variable were tested. The one that worked best was to take an edge whose fractional value is the closest to 0.5. In case of a tie an edge with the maximum distance is chosen.

The selection rule of a node of the branching tree to be processed is known as the "best-node strategy", i.e. a node with the lowest lower bound is to be processed first. This strategy keeps the size of the branching tree to a minimum.

5 Results

We have applied the algorithm to three different instances named A, B and C (the instances are described in Section 2). The algorithm run in a SPARC 1000 with a time limit of 10 hours. We were not able to find certified optimal solutions. The lower bounds obtained are shown in Table 9.

As we said for the optimization problem presented by the company the optimality of the total length was less important than the optimality of the number of ships. Instead of expressing this rather complicated objective function (tests in that direction showed a bad behaviour of the algorithm) we used simply the total length as objective function leaving the optimality of the solutions obtained in terms of number of ships for further investigation. For, although it is quite possible that a solution minimizing the total length does not minimize the number of ships, our practice showed that this is not likely to occur in practical instances, at least for the kind of instances that we were working with. And in fact the solutions we obtained for instances A, B and C — as well as those from Brinati et al. — were optimal in terms of number of ships. For instances B and C the number of ships equals the lower bound $\lceil r_C(V)/7 \rceil$. For instance A this trivial lowerbound is not reached. The certificate of optimality comes from the lower bound of the total length $\tilde{d} = 3955,126$, since the number of ships equal $\lceil r_T(V,\tilde{d})/7 \rceil$. Such a certificate to a solution for instance A was not known before.

We observe that when applying this algorithm for some other instances, if this kind of certificate for the optimality of the number of ships is not obtained, a new optimization would have to be done. The number of trips would be set to be bounded by $7(t^* - 1)$, where t^* is the number of trips obtained. If this problem then happens to be infeasible we would be sure to have a solution minimizing the number of ships.

After having applied the algorithm for the 3 instances we decided to apply the following splitting method to each of the instances in order to improve the quality of the best solution found. First the set of platforms is ordered as $(p_1, \ldots, p_{n-1})$ in a way that the trips in the best solution found R are composed of consecutive platforms. Two partitions are then considered, a first one $S_i^1 = (p_1, \ldots, p_{i-1})$ and $S_i^2 = (p_i, \ldots, p_{n-1})$ and a second $S_j^1 = (p_1, \ldots, p_{j-1})$ and $S_j^2 = (p_j, \ldots, p_{n-1})$. The value of i (j) is chosen as the maximum (minimum) number such that (a) the algorithm obtains a solution for S_i^1 (S_j^2) either optimal or with a very small optimality gap; and (b) the partition $\{S_i^1, S_i^2\}$ ($\{S_j^1, S_j^2\}$) does not split the set of platforms of a trip in R.

This splitting procedure provided the best solutions found for the three instances. They are shown in Tables 6, 7 8. A comparison to the solution obtained by Brinati et al. and the values of the lower bounds are shown in 9.

Even with the efforts made to improve the separation regarding the time constraints, we can note that for instance C, where the time constraint is less tight, we had a much better lower bound than for instance A, the tightest one.

Unfortunately it is not easy to compare our results with those of Laporte et al. Instead of real world they used randomly generated instances controlling the tightness with two parameters, α and L. They run the program for a big number of instances, but restricted the execution time to only 500 seconds and pointed only whether instances were solved to optimality or not, without showing lower bounds. With Euclidian distance, instances up to 50 points were solved. These instances had however the least tight values ($\alpha = 0.66$ and $L = 1000$). We believe that the type of our three instances would fall into the tightness of $\alpha = 0.1$ and $L = 150$. For $\alpha = 0.33$ and $L = 150$ they were able to solve instances only up to 15 points.

6 Conclusion

In this work we investigated a transportation problem that arises in a practical application. Our aim was to investigate the quality of the heuristic solutions provided to the problem by former works by Brinati et al. Using a branch-and-cut approach for the problem we could prove that the number of ships in Brinati's solution were optimal for all instances. Moreover, regarding to the value of the total distance of the trips, we could provide good lower bounds to the optimal solutions and slightly improve the existing solutions (the improvement is of 0.82%, 0.73% and 0.31% resp.). By means of our lower bounds we can prove that the best solutions found lay at most 23.87%, 7.93%, and 0.5% for instances A, B, and C, resp.

trip	route	workers	length (km)	time
1	02 17 18 19 21 20	250	264,723	9h53'
2	31 22	220	278,384	9h40'
3	05 07 10	230	226,143	8h43'
4	06 09 08 04	230	230,267	8h48'
5	03 13 12 11 14	240	238,867	9h09'
6	16 15	215	232,298	8h36'
7	23 26 27	225	280,259	9h48'
8	24 25	205	268,167	9h12'
9	28 35	155	313,731	9h21'
10	29 30 32	225	276,962	9h43'
11	33 34	145	278,426	8h25'
12	39 38	105	374,415	9h50'
13	41 40	90	387,382	9h52'
14	42 37 36	130	361,101	9h57'
15	46 44 45 43	180	312,308	9h44'
16	47 48 49	190	298,644	9h37'
17	50	30	277,013	6h29'
total length: 4899,099 km				
total time: 156h54'				

Table 6: Best solution found for instance A.

trip	route	crew size	length (km)	time
1	05 06 10 08	250	229,764	9h33'
2	03 04 09 07	250	230,547	9h34'
3	13 12 11 18 14	250	246,024	9h56'
4	02	30	204,384	5h17'
5	15 16 17	250	240,923	9h49'
6	19 20 21 22 23	230	282,093	10h27'
7	24 25 32	245	270,627	10h26'
8	26 28	245	294,510	10h59'
9	48 35 29	245	315,722	11h29'
10	27 30 31	245	280,177	10h39'
11	33 34 47	245	304,176	11h13'
12	36 37 38 49	210	356,801	11h52'
13	39 40 41 42	160	390,523	11h50'
14	46 44 45 43 50	210	342,337	11h32'
total length: 3988,617 km				
total time: 144h43'				

Table 7: Best solution found for instance B.

trip	route	crew size	length (km)	time
1	06 07 10	240	228,538	6h50'
2	08 09 05	190	226,143	6h38'
3	02 15 16	245	232,984	6h58'
4	04 17 18 19 21 20	250	265,590	7h51'
5	14 11 12 13 03	240	238,867	7h06'
6	24 25 23	245	268,447	7h55'
7	22 30 29	240	286,155	8h23'
8	26 28	245	294,510	8h37'
9	33 34 48 49	235	295,500	8h37'
10	31 32	205	265,413	7h44'
11	47 35 36 27	240	345,172	9h59'
12	38 37 39 40 41	245	394,898	11h20'
13	46 44 45 42 43 50	245	342,337	9h55'
total length: 3684,461 km				
total time: 108h00'				

Table 8: Best solution found for instance C.

instance	A	B	C
solution	4899,099	3988,617	3684,561
lower bound	3955,126	3695,533	3666,135
Brinati et al.	4940,020	4018,248	3696,142

Table 9: Best solutions compared and lower bounds (lengths in km)

References

A.V. Aho, J.E. Hopcroft, J.D. Ullman, “The Design and Analysis of Computer Algorithms”, Addison-Wesley, 1974.

J.R. Araque, G. Kudva, T.L. Morin, J.F. Pekny, “A branch-and-cut algorithm for vehicle routing problems”, *Annals of Operations Research* **50** (1994) 37–59.

L.D. Bodin, B.L. Golden, A. Assad, M. Ball, “Routing and scheduling of vehicles and crews, the state of art”, *Computers and Operations Research* **10** (1983) 69–211.

M.A. Brinati, R.C. Botter, J.R.R. Roque, M.T. de Gouvea, R.L. de Souza, P.R.B. Carneiro, “Aplicação de heurísticas de roteamento para o dimensionamento de uma frota de transporte marítimo”, *Anais do Congresso Latino-Ibero-Americano de Pesquisa Operacional e Engenharia de Sistemas* **21** (1988) 349–363.

V. Campos, A. Corberan, E. Mota, “Polyhedral results for a vehicle routing problem”, *European Journal of Operations Research* **52** (1991) 75–85.

G. Cornuéjols, F. Harche, “Polyhedral study of the capacited vehicle routing problem”, *Mathematical Programming* **60** (1993) 21–52.

M. Jünger, G. Reinelt, S. Thienel, “Practical problem solving with cutting plane algorithms in combinatorial optimization”, *DIMACS Series in Discrete Mathematics and Theoretical Computer Science* **20** (1995) 111–152.

G. Laporte, Y. Nobert, M. Desrochers, “Optimal routing under capacity and distance restrictions”, *Operations Research* **33** (1985) 1050–1073.

M. Loparic, “Uma aplicação do método de *branch-and-cut* a um problema de roteamento de veículos”, M.S. dissertation, Instituto de Matemática e Estatística, Universidade de São Paulo (1996).