Noname manuscript No.
(will be inserted by the editor)

Truck routing and scheduling

Csongor Gy. Csehi · Márk Farkas

Received: date / Accepted: date

Abstract The problem is part of a complex software solution for truck itinerary construction for one of the largest public road transportation companies in the EU. In practice a minor improvement on the operational cost per tour can decide whether a freight services company is profitable or not. Thus the optimization of routes has key importance in the operation of such companies. If parking places and gas stations are included in the planning then it is NP hard to find an optimal solution. This means that for long range tours an approximately optimal solution for refueling has to be given within an acceptable running time. Also the corridoring of the trucks is an important problem so that we try to optimize the performance, hence tours cannot be recalculated at each data arrival. The vehicle assignment part of this work is already finished and applied with very good results. The remaining part is subject of an ongoing research which started at January 2014. The company started to apply and test our product in the beginning of 2015 under increased human supervision. As a consequence of the project a large cost saving is anticipated by the company.

Keywords routing · logistics · scheduling · truck

This work was partially supported by grant No. OTKA 108947 of the Hungarian Scientific Research Fund.

Csongor Gy. Csehi
Budapest University of Technology and Economics
E-mail: cscsgy@cs.bme.hu

Márk Farkas
Nexogen
E-mail: farkas.mark@nexogen.hu

1 Introduction

A schematic model of the problem is the following: Suppose that there are trucks with actual latitude and longitude coordinates in Europe. All the trucks have one or two drivers. Demands can appear anytime. A demand consists of uploading points and downloading points. Our software gives a quasi optimal solution which determines which truck should undertake which demand, and how this truck should select the roads and make the schedule for the drivers. The matching of the trucks and demands is a covering minimal matching in a weighted bipartite graph with on-line emerging nodes. It is a whole separated project in our company which has been finished before, and is now running with great results. Now we will mainly focus on the other aspect of the problem, so we want to construct a route for one truck with given tasks.
There are many existing softwares in the market which can give the best route –in some sense– between two points. For the subproblem, to give the best route between two points we also use existing software. However our problem is different, we want the best route with scheduling, which means that parkings and refuelings included in the route in optimal locations. In most of the times the best plan with scheduling is not even near the best route between the starting and ending location. There are many problems with the best route finding solutions, because there is no such best route. Of course the motivation of the optimization is profit maximization so we need the cheapest route but there are many type of costs. There is the road cost which we have to pay if we use the road. There is the fuel cost which depends mostly on the length of the road, on the weight of the truck (with the load) and on the height fluctuation of the road (and also the speed limits, the quality of the road, etc). There is an amortization cost which mostly depends on the length, the quality and the sinuosity of the road. Time costs money in many senses, for example the extra salary of the driver and the delay compensation of the customer. We try to optimize the routes by fine-tuning the settings of the applied routing software. The schedule of the drivers must meet the criteria specified by the (EC) 561/2006 directive. There are upper bounds inter alia in the continuous driving time, in the daily driving time and in the weekly driving time. There are lower bounds inter alia in the break time, in the daily rest time and in the weekly rest time. We determine where should the driver take the rests according to safety limitations with overall cost optimization. Also we give the quasi optimal refueling locations in the sense of overall cost. Moreover we observed that we get better results if we take over the decision of determining the border, the tunnel and the bridge usage from the applied routing software. The frame of our solution is a branch and bound algorithm which we modified in many aspects.
We build a route graph with the help of PTV Group softwares. In this graph the nodes represent often used places in Europe. There are parking places, borders, gas stations, ferries, tunnels and bridges. This graph will be the main database, so during the algorithm whenever it is possible we try to use it instead of PTV, and accelerate the computing that way. Using the Floyd-

Warshall algorithm (Floyd [5] 1962; Warshall [13] 1962) on the route graph we construct two new graphs on the same vertex set, one for the cheapest options and one for the shortest ones between each pair of nodes. These two graph will give the heuristics for the branch and bound algorithm.
Now that we summarized the most relevant problems and tools we give a more detailed model of the main problem. We have a truck with one or two drivers and we have to construct a cost saving route which guides the truck to the location of the tasks in good order and in proper time-windows, without law-breaking. To keep the state of the driver legal we have to keep proper rests on some of the parking places of the previously defined route graph. We construct a route which first reaches a node of the route graph. Then in each step the plan goes in a neighbouring node –which means a driving activity– or do something in the node where it stays –which can mean a refueling, a rest, a border crossing, ferrying or waiting. If we are close enough to reach the place of the next task, the plan can also drive there. Obviously if the task is near enough to the starting location, the plan can go there without even using the route graph.
The model does not have a pretty Linear Programming form. Instead we will speak about nodes of the planning which represents a subroute of a solution. Such a node stores a valid route from the starting position and state, to an other position and state, with all the activities during the elapsed time.
It is easy to see that the refueling problem can not be solved separately, because the place of the filling station modifies the route itself, such as the parking places.
We will study the corridoring problem also. This is where we try to decide time after time if the original plan can be hold without law-breaking, and is still optimal. The question is that if the plan is still legal then what is the possibility for a much better solution to exist. Here a solution means the same as in the planning problem.

2 History

The area of operations research is widely studied (Carter and Price 2000 [2]). However to build such a complex solving system for truck routing and scheduling is a novel concept. Some approaches and solutions which are at least a bit similar to this problem have been collected in this section.
The theory of logistic management is about to plan the flow of goods (Schönsleben 2011 [10]). Here the problem is more about recurring demands. Also the storage plays important role in logistic management and unfortunately the driver-state and the resting places of the drivers is irrelevant in this theory.
The container terminal optimization has a huge amount of publications (Kozan and Preston 1999 [7]) but it is also very specific and has many properties which differs from the truck routing and scheduling problem. For example a subproblem is to optimize the truck trips inside the container terminal area (Chen, Langevin and Lu 2013 [3]; Miao, Lim and Ma 2009 [9]).

Hub-and-spoke transportation is a type of third-party logistics (Zäpfel and Wasner 2002 [14]). Therefore it is closer to our problem than the logistic management since the goods are not direct part of the optimization but the tours get leading role. However there the hub-and-spoke structure gives many specialities and it is more likely to be applicable in short-range tours. That way the driver-state and rest planning part is missing from that theory.
Scheduling of dispatching ready mixed concrete (RMC) trucks is an other short-range logistic problem (Feng, Cheng and Wu 2003 [4]). Here the material needs more attention and therefore it has an important role in the optimization.
There are also some attempts to give mathematical formulation to international logistic type problems (Takeyasu and Kainosho 2014 [11]). This formules are too general in the transportation type and can not optimize to the driver states. They work with air and sea transportation too. On the other hand they suppose a single supply site and a single demand site.
The problem in the paper of Adamski (2007 [1]) is more about public transportation, but he sums up well the difficulties of making such complex integrated system for logistics.
The vehicle routing problem (Toth and Vigo 2001 [12]) is well studied, but it deals with just a subproblem of truck routing and scheduling. In the paper of Giaglis, Minis, Tatarakis and Zeimpekis (2004 [6]) a subproblem of the vehicle routing has been studied. However this approach is also a subproblem of the corridoring part of truck routing and scheduling. They deal with the importance of the strong connection between the vehicle and the control center.

3 Problems

3.1 Refueling

Fuel cost takes up the largest part of the expenses of a transportation company. It can be a difficult problem to determine how many times and how much we should refuel because the trucks have huge fuel tanks and it is unnecessary to fill completely in each refueling. This problem is not novel, however we can not use the earlier results since the detours for refueling can modify the route and the schedule. That way we involve the refueling in the planning.
Performance criteria does not allow us to examine all possibilities. Hence in the first step we try to outline the possible countries near the region between the specific destinations with the best fuel prices, so that we can minimize the amount of work left to the branch and bound part of the algorithm. Here many different heuristics can be applied: For example suppose that we have to drive through countries with high fuel prices. Since we do not know yet the next destination of the vehicle, we guess whether a refueling should be performed or left for the next country with cheaper fuel prices. The on-line nature of the problem requires that decisions have to be made before knowing every

condition, but in some cases changing the remaining part of a tour according to the new information may be worthy. With these assumptions we always try to make the tours not just optimal, but robust in the sense that it can adjust to the most likely new information with only local changes. In fact, we try to optimize the matching of the vehicles to the tours with the additional assumption that each truck will pass through countries with low fuel prices within each 2000km. This would result in never having to buy fuel in countries with expensive fuel price. However it wouldn't be so much cheaper because of the additional distances that comes from the suboptimal covering matching of trucks and demands. Moreover this attempt would make the matching algorithm more difficult and it would have worse performance.

3.2 Corridoring

The corridoring consists of two parts, the processing of historical data and the analysis of enforcability of the earlier plan.
The historical data is mainly a set of positions. We get information from the drivers in regular time periods. From this information we try to guess which jobs and activities are already completed by the driver and estimate how much time is left from the actual activity. It would be much easier if we knew this information from the drivers, but there are some obstacles to achieve that. It would be much easier if we would know this information from the drivers, but there are some obstacles to achieve that. First of all the drivers do not have yet such a tool where they can fill in this data. The number of operators is less than how many would be needed to handle all that information on telephone. The tasks have locations and estimated time of arrival (ETA). One of the difficulties is that the real date of the tasks can differ from the ETA and most of the tasks can be skipped (all the non customer tasks, namely the borders, parkings etc). Moreover sometimes the positions are missing for some time periods. In this way the problem is not like a string searching type problem but more like a longest path in a graph. We implemented this search with additional studies according to certain assumptions. For example we have to stay in a position for at least a fixed duration to consider it as finishing that task. Another assumption is that we cannot skip important tasks in the timeline.
There are many location specific analysis too. If the task is refueling then we recognize it only if the fuel quantity grows when the truck is near the location. If there is a ferry task then we can recognize it by island indices (these correspond to areas which are separated from each other and we can cross them only with ferries). The borders can be recognized by country codes. It is important that if we recognize a task but cannot find positions near the location then we don't fill the fact data for this task, because it is possible that we make it somewhere else. For example we can cross a border in another place or take another ferry and this way the costs are different from the previously calculated ones.

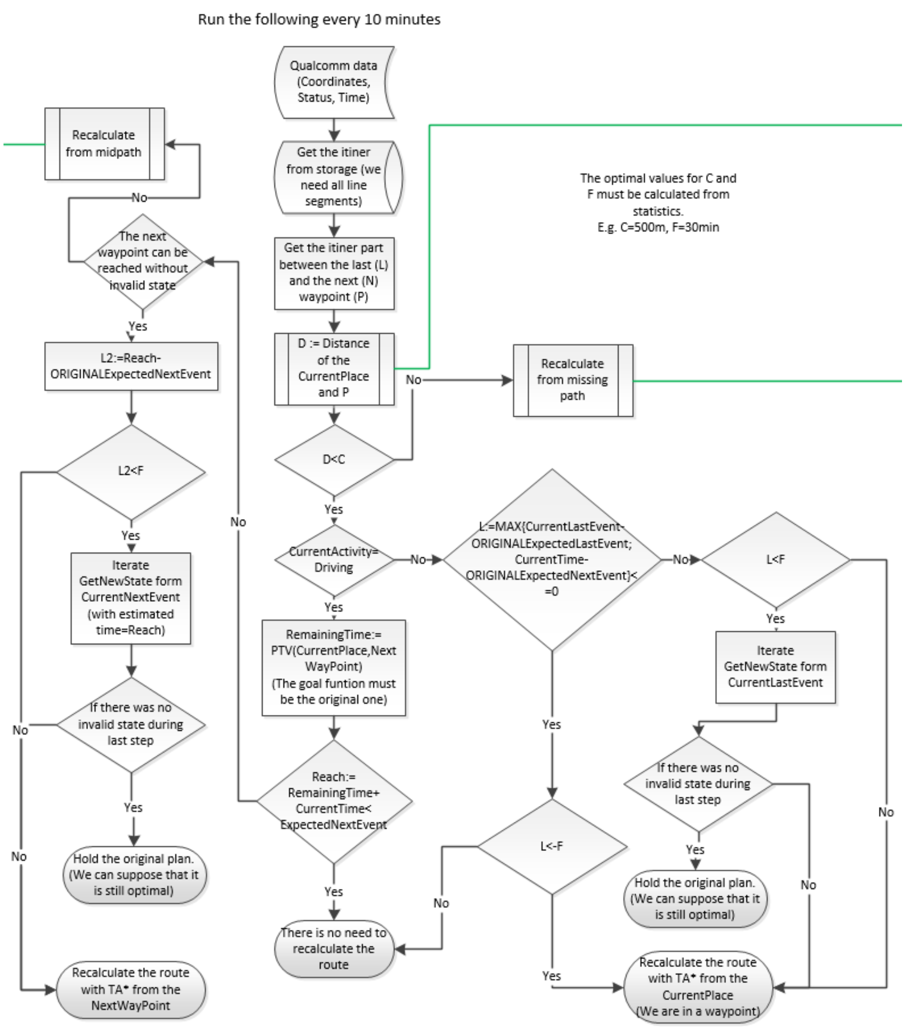

Run the following every 10 minutes
Qualcomm data (Coordinates, Status, Time)
Get the itiner from storage (we need all line segments)
Get the itiner part between the last (L) and the next (N) waypoint (P)
D := Distance of the CurrentPlace and P
No
Recalculate from missing path
The optimal values for C and F must be calculated from statistics. E.g. C=500m, F=30min
D<C
Yes
CurrentActivity= Driving
No
L:=MAX(CurrentLastEvent-ORIGINALExpectedLastEvent; CurrentTime-ORIGINALExpectedNextEvent]< =0
No
L<F
Yes
RemainingTime:= PTV(CurrentPlace,Next WayPoint) (The goal funtion must be the original one)
Reach:= RemainingTime+ CurrentTime< ExpectedNextEvent
Yes
There is no need to recalculate the route
Yes
L<-F
No
Yes
Iterate GetNewState form CurrentLastEvent
If there was no invalid state during last step
Yes
Hold the original plan. (We can suppose that it is still optimal)
No
No
Recalculate the route with TA* from the CurrentPlace (We are in a waypoint)
Recalculate from midpath
No
The next waypoint can be reached without invalid state
Yes
L2:=Reach-ORIGINALExpectedNextEvent
L2<F
Yes
Iterate GetNewState form CurrentNextEvent (with estimated time=Reach)
If there was no invalid state during last step
Yes
Hold the original plan. (We can suppose that it is still optimal)
No
No
Recalculate the route with TA* from the NextWayPoint
No

Fig. 1 Schematic representation of the second part of the corridoring

After that we processed the historical data, we calculate whether the destinations can be reached in the expected time window or not. There can be cases where we could find better solution despite the destinations can be reached in time. This is because many types of problems can occur, for example the truck misses the planned road (maybe goes in a faster but more expensive way), some work took less time than we expected, the driver finished a rest later or there was a traffic jam in the road. We make a decision using many changeable settings to allow fine-tuning later on. When replanning a tour caution has to be taken on the timing. By the time the driver receives a new itinerary the driver-state and position have already changed compared to the data used for planning. To do so we have to estimate the future position and

state of the driver.
A schematic representation of the analysis of enforcability part of the corridoring can be seen in Figure 1. The diamonds represent case separations, the rectangles are parts of the algorithm and the rounded boxes are the resulting decision. If the position is still in the road there will be two different cases. The left hand side represents the case where the actual activity is driving, and there is some delay. The right hand side represents the case where the actual activity is not driving, so we are in a node of the route graph or in a task location.

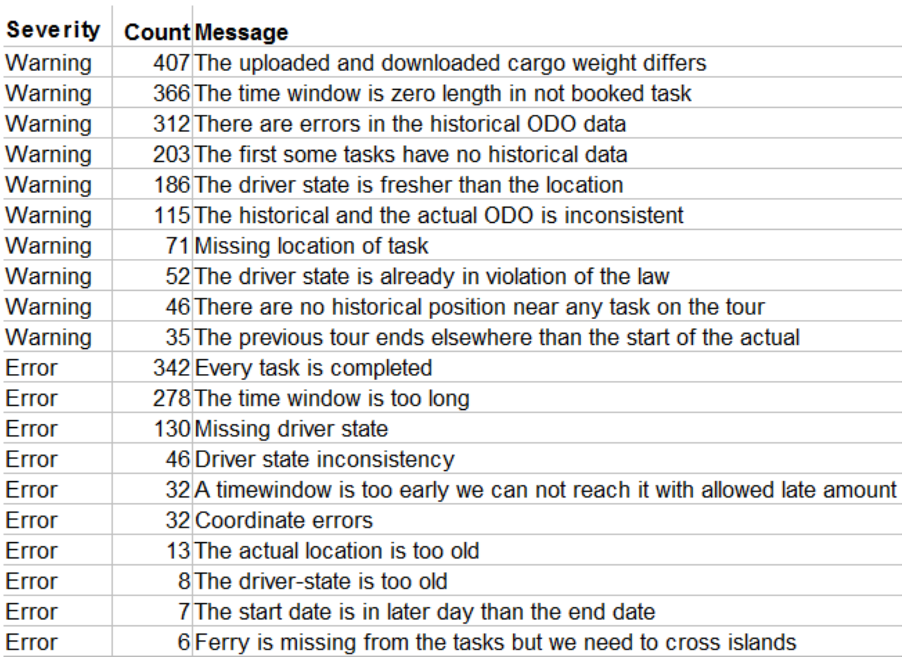
3.3 Errors in the data

There can be data errors in the incoming information and they can result in various system mistakes what we have to deal with. We try to filter the errors by assumptions from real experiences. Smaller mistakes are also corrected automatically. Some functionalities in the planning algorithm also suppose that if wrong incoming data have incorrectly been replaced then the corridoring will find it out leading to a better solution.
The most frequent data errors can be seen in Figure 2. These statistics are made for each run so we can see the changes time by time. There are errors where it is unnecessary to calculate the tour because the mistakes are incorrigible. There are errors where we calculate the tour with some corrections but send a warning message to the operator about the inconsistencies.
In Figure 3 there are some driver-states where some of the variables give contradictioning system. In this case there were about 80 driver-states with error among the approximately 3000 incoming states. The most frequent error (in 53 cases) was when the weekly driving time is higher than the driving time of the last two weeks. The other two frequent errors are when the daily driving time is higher than the weekly driving time and when the duration of the actual working is higher than the continuous working time.
The whole software is designed to work under human supervision. In most of the cases the operator can correct our missed data estimations. Also the main reaction to any problem is an exact failure message to the operator, who can investigate the reasons and can give information to help us find the best solution for the problem.

4 Algorithm

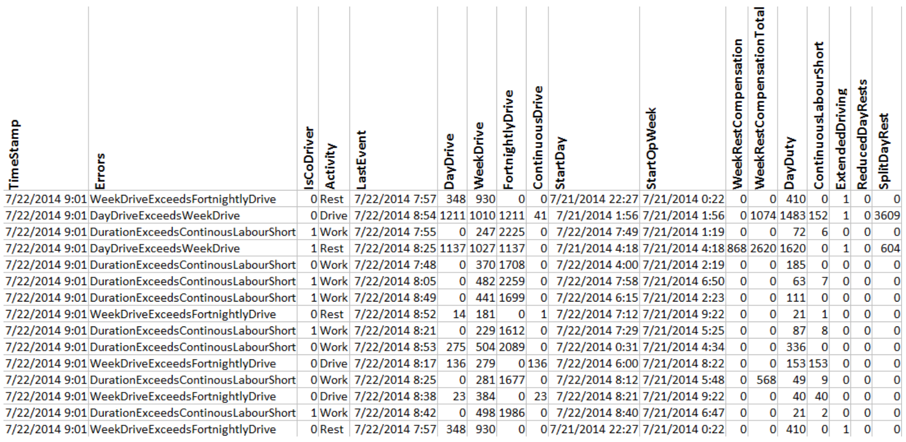
4.1 Branch and bound

The core of our algorithm is a branch and bound (Lawler and Wood 1966 [8]) structure. Every node of the tree has position, time, driver-state and truck-state. In each node there is a cumulated objective function value for the road done and a heuristic value which is a lower bound for the objective function



Severity	Count	Message
Warning	407	The uploaded and downloaded cargo weight differs
Warning	366	The time window is zero length in not booked task
Warning	312	There are errors in the historical ODO data
Warning	203	The first some tasks have no historical data
Warning	186	The driver state is fresher than the location
Warning	115	The historical and the actual ODO is inconsistent
Warning	71	Missing location of task
Warning	52	The driver state is already in violation of the law
Warning	46	There are no historical position near any task on the tour
Warning	35	The previous tour ends elsewhere than the start of the actual
Error	342	Every task is completed
Error	278	The time window is too long
Error	130	Missing driver state
Error	46	Driver state inconsistency
Error	32	A timewindow is too early we can not reach it with allowed late amount
Error	32	Coordinate errors
Error	13	The actual location is too old
Error	8	The driver-state is too old
Error	7	The start date is in later day than the end date
Error	6	Ferry is missing from the tasks but we need to cross islands

Fig. 2 The most frequent data errors



TimeStamp	Errors	IsCoDriver	Activity	LastEvent	DayDrive	WeekDrive	FortnightlyDrive	ContinuousDrive	StartDay	StartOpWeek	WeekRestCompensation	WeekRestCompensationTotal	DayDuty	ContinuousLabourShort	ExtendedDriving	ReducedDayRests	SplitDayRest
7/22/2014 9:01	WeekDriveExceedsFortnightlyDrive	0	Rest	7/22/2014 7:57	348	930	0	0	7/21/2014 22:27	7/21/2014 0:22	0	0	410	0	1	0	0
7/22/2014 9:01	DayDriveExceedsWeekDrive	0	Drive	7/22/2014 8:54	1211	1010	1211	41	7/21/2014 1:56	7/21/2014 1:56	0	1074	1483	152	1	0	3609
7/22/2014 9:01	DurationExceedsContinousLabourShort	1	Work	7/22/2014 7:55	0	247	2225	0	7/22/2014 7:49	7/21/2014 1:19	0	0	72	6	0	0	0
7/22/2014 9:01	DayDriveExceedsWeekDrive	1	Rest	7/22/2014 8:25	1137	1027	1137	0	7/21/2014 4:18	7/21/2014 4:18	868	2620	1620	0	1	0	604
7/22/2014 9:01	DurationExceedsContinousLabourShort	0	Work	7/22/2014 7:48	0	370	1708	0	7/22/2014 4:00	7/21/2014 2:19	0	0	185	0	0	0	0
7/22/2014 9:01	DurationExceedsContinousLabourShort	1	Work	7/22/2014 8:05	0	482	2259	0	7/22/2014 7:58	7/21/2014 6:50	0	0	63	7	0	0	0
7/22/2014 9:01	DurationExceedsContinousLabourShort	1	Work	7/22/2014 8:49	0	441	1699	0	7/22/2014 6:15	7/21/2014 2:23	0	0	111	0	0	0	0
7/22/2014 9:01	WeekDriveExceedsFortnightlyDrive	0	Rest	7/22/2014 8:52	14	181	0	1	7/22/2014 7:12	7/21/2014 9:22	0	0	21	1	0	0	0
7/22/2014 9:01	DurationExceedsContinousLabourShort	1	Work	7/22/2014 8:21	0	229	1612	0	7/22/2014 7:29	7/21/2014 5:25	0	0	87	8	0	0	0
7/22/2014 9:01	DurationExceedsContinousLabourShort	0	Work	7/22/2014 8:53	275	504	2089	0	7/22/2014 0:31	7/21/2014 4:34	0	0	336	0	0	0	0
7/22/2014 9:01	WeekDriveExceedsFortnightlyDrive	0	Drive	7/22/2014 8:17	136	279	0	136	7/22/2014 6:00	7/21/2014 8:22	0	0	153	153	0	0	0
7/22/2014 9:01	DurationExceedsContinousLabourShort	0	Work	7/22/2014 8:25	0	281	1677	0	7/22/2014 8:12	7/21/2014 5:48	0	568	49	9	0	0	0
7/22/2014 9:01	WeekDriveExceedsFortnightlyDrive	0	Drive	7/22/2014 8:38	23	384	0	23	7/22/2014 8:21	7/21/2014 9:22	0	0	40	40	0	0	0
7/22/2014 9:01	DurationExceedsContinousLabourShort	1	Work	7/22/2014 8:42	0	498	1986	0	7/22/2014 8:40	7/21/2014 6:47	0	0	21	2	0	0	0
7/22/2014 9:01	WeekDriveExceedsFortnightlyDrive	0	Rest	7/22/2014 7:57	348	930	0	0	7/21/2014 22:27	7/21/2014 0:22	0	0	410	0	1	0	0

Fig. 3 Some driver-state which consists inconsistency

value of the remaining road. In each step the algorithm chooses the node which has the best sum for cumulated objective function value and estimated objective function value, but has not already expanded in earlier steps. What makes it hard to bound some branches is that in this problem it is possible for two nodes in the same intermediate location that the node with a worse objective function value will give the better solution. This way we must not throw away all but the best in each location. We will specify the bounding method after

presenting the building of the tree.
The driver-state consists of many variables. There are variables for the driving time measuring (continuous, daily, weekly, fortnightly), the working time measuring (continuous, daily), the elapsed time measuring (time, daytime, weektime, number of weeks from the last not reduced weekly rest), storing compensations (weekly, fortnightly), and many counting variables (number of: extended driving days, reduced daily rests, small breaks). The transition between two driver-states depends not just on the activity but the time, too, since there are many regulations corresponding to calendar weeks.
From each node we branch limited number of new nodes by adding some new

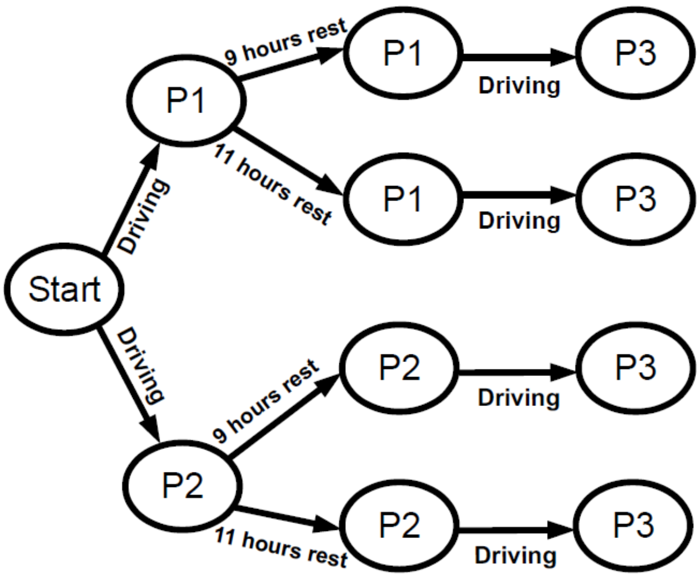


Fig. 4 An example of a subtree of the algorithm

activities. A very simple example of the branching can be seen on Figure 4 (without the data sets in each node). In this example suppose that there are three different parking places $P1$, $P2$ and $P3$. We can reach $P1$ and $P2$ from the starting state and location so we make those two new nodes which are after the driving activity of reaching the new locations. In each parking place we can rest any amount of time, for example 9 or 11 hours. That way we make new nodes after the restings, which are in the same location but with different driver state. Suppose now that $P3$ can be reached from both $P1$ and $P2$ that way we can make four new nodes corresponding to the four different reaching of $P3$.

4.2 Branching rules

To branch just the most important possibilities from each node we apply many rules which come from the real life application. In this subsection we summarize the most frequently used rules.
Mainly after each driving activity there comes a non-driving activity. This way if we want to find new nodes from a node where the last activity was a driving activity then we try to collect the possibilities in the position. This can be refueling at a gas station, resting at a parking place, working at the target, or traveling with a ferry. If we are at a parking place the length of the upcoming rest has to be determined. It is a hard problem to determine the length of the rest since at the node it is unknown what will be the next place and when will it be open. This problem is solved by the *backthrow* method which can increase the duration of the rest at some previous position. If a node arrives to a position which is closed then it finds out how much time is needed and it makes a backthrow with this amount. The backthrow finds the last rest activity in the past and try to copy the branch of the node with a different state chain (comes from the longer rest).
The refueling quantity is hard to find out, too. If the price of the fuel is really cheap at the gas station then a full refueling is performed. Otherwise estimations are made on how much fuel will be needed to reach a cheaper station (or the end of the tour).
At the target the risk of being late is reduced by calculating the proper amount of waiting. The last day of the driver is analized and he is sent to the target earlier if there is a long road from the parking place to the target.
If we want to find new nodes from a node where the last activity is non-driving then we try to find the best possible opportunities where to go next. It is important that we cannot make a node from all possibilities because this will result an enormous number of nodes in the tree (this limit gives the base of the exponential function which is an upper bound for the number of steps in the algorithm). We try to follow some rules to give the set of best possibilities. First we try to give positions with smallest sum for cumulated cost and estimated future cost. In this sum if we go to a parking place we weight the two components so that we focus on the remaining values because this way we can drive more in each day. We filter the same type of places by the distance. If two places of the same type are close to each other then we hold only the better as a solution. This filtering does not affect the number of resulting places. The reason of this is that in most of the cases the best values for the mentioned sum are close to each other. However sometimes a slower or more expensive route can result in better solution because for example it goes in the direction of cheaper gas stations, avoids an area with many restrictions or less parking places. On the other hand if two parking places are on the same road, in the same price but one is a bit further (in the good direction) then there is no way that the earlier is better.

4.3 Bounding

The bounding part of the branch and bound structure throw away a node if it is majorated by another node in the same location, namely the values of all of its variables are at most as good as those of the majorating one. This results a higher dimensional branch and bound structure. In this approach there can be infinitely many non-majorated nodes in the same location (if the dimension is at least two). For example in Figure 4 among the four nodes in $P3$ those who came from the same previous parking place can not majorate each other because one consumes a reduced daily rest (9 hours) but reaches $P3$ earlier, and the other is not consuming the reduced daily rest but reaches $P3$ later. However if the way through $P2$ is cheaper and shorter than the way through $P1$ then the two nodes in the top right in $P3$ can be discarded by the bounding.

4.4 Heuristics

To calculate a lower bound for the objective function value of the remaining road in a node we have to estimate the minimal road cost, distance and duration needed to reach the goal. For this we use the graph - previously constructed with the Floyd-Warshall algorithm - which stores the minimal distances and durations between each pair of nodes. The cost and the distance are counted in separately without modification to the objective function. For the duration we also have to give a lower bound to the sum of the needed rest and wait durations during the remaining route. We calculate these lower bounds by supposing that there will be proper parking places at the exact locations on the shortest (in duration) route where the driver runs out of driving time. After that we can give an earliest arriving time to the goal, and can calculate the objective function value corresponding to this fictional route (the three values, cost, duration and distance, can come from different routes).

5 Objective functions

We have two objective functions: a cheap and a fast.
These functions first estimate the business cost of every measurable thing in the tour. We calculate not just the real costs of the roads but the cost of the driver's work, the amortization of the car and many other things. The hardest part is the evaluation of the driver's state in the end of the tour. Of course it is better if the driver can drive more in the day after he finishes the last working task because that way he can approach the location of the next task. In fact every variable of the driver-state can be important in the end of the tour. We mainly apply a highly tested linear combination of these quantities. The cheap objective function tries to focus on the business cost while the fastest gives more to the duration of the tour.

One of the hardest parts of the whole problem was that in the first months of customer testing it turned out that they optimized for much more things than it was previously anticipated. The finishing driver-state is the newest component what we built in, but we have not finished it yet. The next problem is to measure other properties of the roads than the length, duration and cost. The height alternation can influence the fuel consumption and can make worse amortization value. Also the real cost of the sinuous or potholed roads can be much higher than it seems in the maps.

6 Validations and exceptions

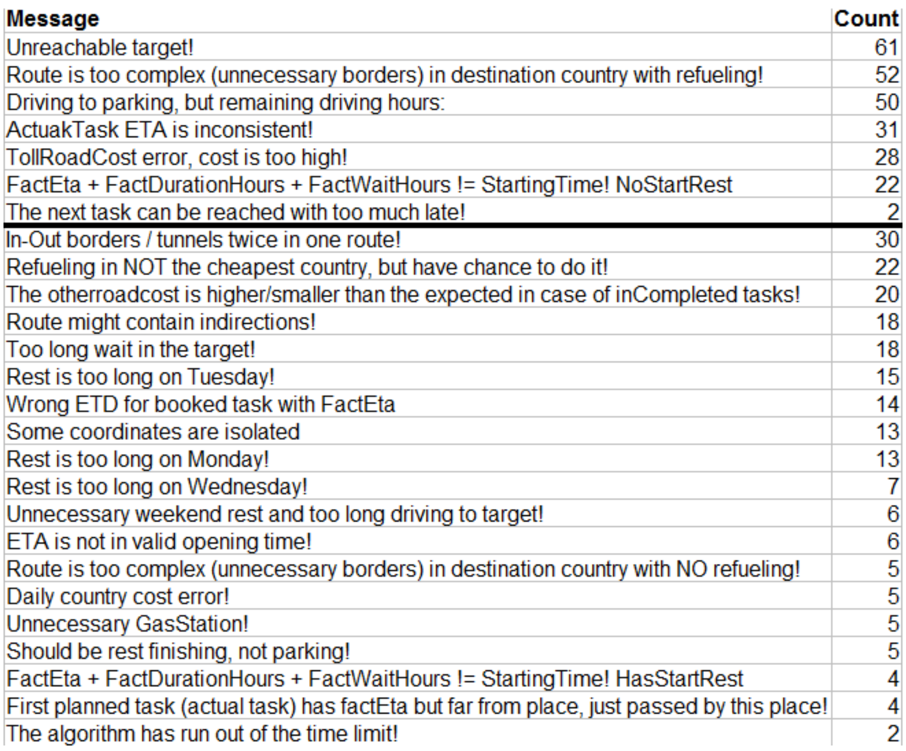
We made a huge number of validations according to previous remarks of the users. The validations warn us for possible planning errors, but they are not obligatory rules. We will show how can we suspect mistakes, and we will give examples where the suspicion is not correct and the plan is optimal.
The validations are important because each time a new feature is incorporated or some limiting values or coefficients were changed some old problems resurfaced. With the help of validations these anomalies can be identified and better solutions can be found.
These validations are mostly about one duty or two which are close to each other. Here comes some frequent examples.
Suppose that two parking places are in only some hours driving distance from each other. In most of the cases this means that we could go further on that day, but there are many exceptions. For example if we are near to the location of the task but not really close then in most of the cases it is better to go closer and this way reduce the risk of being late.
If we hold a seemingly unnecessarily long rest it can be the result of some conflicting heuristics, for example if the finishing driver-state has a too large coefficient. On the other hand it can be the part of the best solution if there are some country or road restrictions which stops us for longer time.
If we make a full refueling in an expensive station it can be a problem if we go through cheaper areas in the tour. However in some cases it can be optimal because it would be more expensive to refuel in the cheap places because of the detours.
If we cross the borders of two countries two or more times one after another it can mean that we make detours. There are some places where the best route must cross the border many times for geographic reasons.
If two refuelings are close to each other it can be the fault of the fueling quantity calculator. However if we can not reach the cheap place without refueling then we should make a small refueling to reach the cheap station and then make a full refueling there.
If we reach a task's location outside the opening time it can be because some mistakes in the availability converters or cutters. In some cases we have to go to a place earlier than it opens for example if we are closer to the position than to any parking place.

7 Results

The results of the software are better and better. Every month the customer company applies more tours planned by us and work less on verifying it. Nowadays the emerging problems are mostly because of user errors, user inattention or mapping problem.
Since previously all the planning was made manually by operators (and no such solution exists at other companies), we can not compare the running time of our solution to the previous one. Also the setting of the many constants coordinating the behaviour of the algorithm makes it impossible to measure an exact speed. By changing the number of allowed branches from each node in the branch and bound we can achieve faster running in the cost of a bit less optimal solutions. We limit the time of planning one single tour in 3 minutes, and set the branching limit to keep at least 99% of the plannings under the limit. We also limit the number of the expanded nodes by 300000 (less than 0.1% of the plans reach this limit). That way the planning of about 3000 tours runs in about 30 minutes (in 160 parallel machines).
The optimality of our results can not exactly been compared to the previous because the operators do not plan the same tasks again what we have planned. There are some measures for the efficency of the company but it is hard to tell anything about our solution from these measures (because they depends on the economy and growth). What we can tell is that about 90% of our plans (which was verified by the operators) do not need any changes so they are at least as good as the earlier plans.
An other concept to measure our solution's efficiency is to check the reason of the fails during planning.
Figure 5 shows the routing errors of one package. This package is one of the packages on September 7, 2015. The run finished with 1046 successful and 31 failed plannings. Almost all of the failed plannings are because of some delay in the tour (we could not reach the target in the given time window). This problem is mostly because some historical locations are missing so we don't have information for some completed tasks and therefore we try to make them again and that is time-consuming. The rows above the line are problems where we do not even give a result, and the rows under the line are just for statistical analysis. These numbers correspond to the two times 1046 tours (cheap and fast versions for each).

8 Interface

The interface is a very important part of the software, because it makes the results editable and more understandable for the operators. We have learned that for the customer an optimal solution is worthless without a good looking view.
Figure 6 shows a planned tour on the interface. There are many types of



Message	Count
Unreachable target!	61
Route is too complex (unnecessary borders) in destination country with refueling!	52
Driving to parking, but remaining driving hours:	50
ActuakTask ETA is inconsistent!	31
TollRoadCost error, cost is too high!	28
FactEta + FactDurationHours + FactWaitHours != StartingTime! NoStartRest	22
The next task can be reached with too much late!	2
In-Out borders / tunnels twice in one route!	30
Refueling in NOT the cheapest country, but have chance to do it!	22
The otherroadcost is higher/smaller than the expected in case of inCompleted tasks!	20
Route might contain indirections!	18
Too long wait in the target!	18
Rest is too long on Tuesday!	15
Wrong ETD for booked task with FactEta	14
Some coordinates are isolated	13
Rest is too long on Monday!	13
Rest is too long on Wednesday!	7
Unnecessary weekend rest and too long driving to target!	6
ETA is not in valid opening time!	6
Route is too complex (unnecessary borders) in destination country with NO refueling!	5
Daily country cost error!	5
Unnecessary GasStation!	5
Should be rest finishing, not parking!	5
FactEta + FactDurationHours + FactWaitHours != StartingTime! HasStartRest	4
First planned task (actual task) has factEta but far from place, just passed by this place!	4
The algorithm has run out of the time limit!	2

Fig. 5 The resulting routing errors of a run

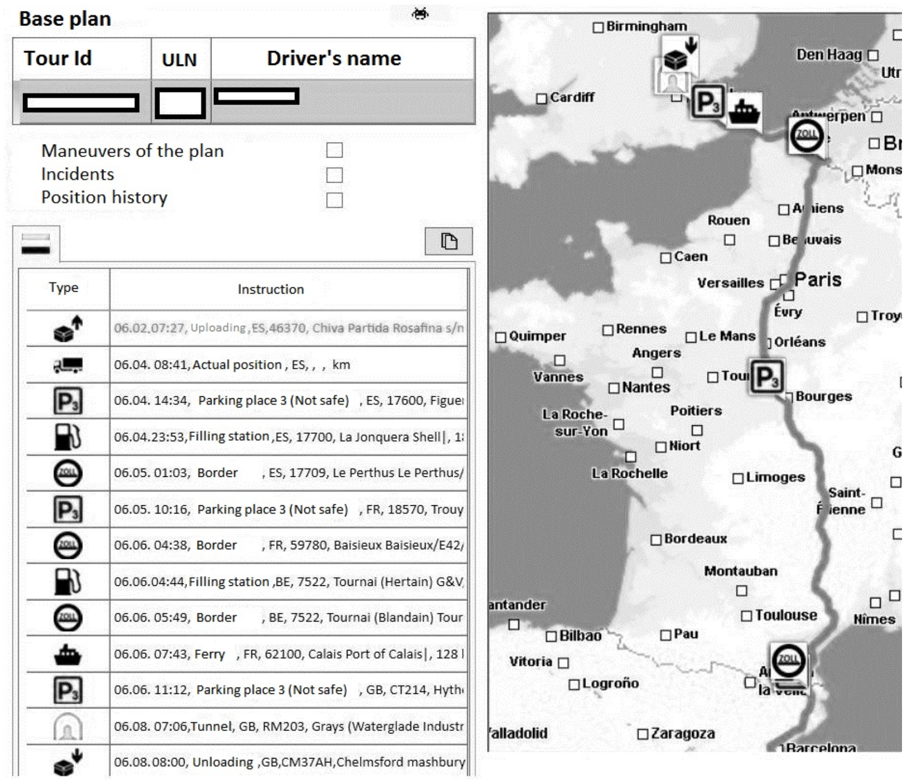
presentations of the tours. This way the operators can analyze the results in the map or in timetables. The operators of the company are mostly Hungarian so the main language of the program is set to Hungarian (but it can be changed). In the map the tasks are denoted by different logos according to the type of the tasks. In the PTV map the operators can zoom in to analyze the route in details.

In the task list in the left side of Figures 6 and 7 the types the places and the estimated arrival times are shown. The timetables have a more detailed view too where the operators can check the driving times, the breaks, the working durations and the waitings as well.

In Figure 7 the wider black path consists of the dots of the historical locations. It can be seen that the tasks in the beginning are identified to be completed and the plan goes for the first incompleted task. In the timetable the color of the completed tasks are different from the incompleted ones.

Figure 7 also demonstrates that if we don't have fixed task in Switzerland then we should avoid it due to the different regulations.

Figure 8 shows a tour with many tasks. The problem with those tours is that it is much harder for the algorithm to give the solution since there will be much more nodes in the branch and bound like part. This problem can occur if the distance (and duration) of the tour is large. However in this example the



Base plan

Tour Id	ULN	Driver's name
[redacted]	[redacted]	[redacted]

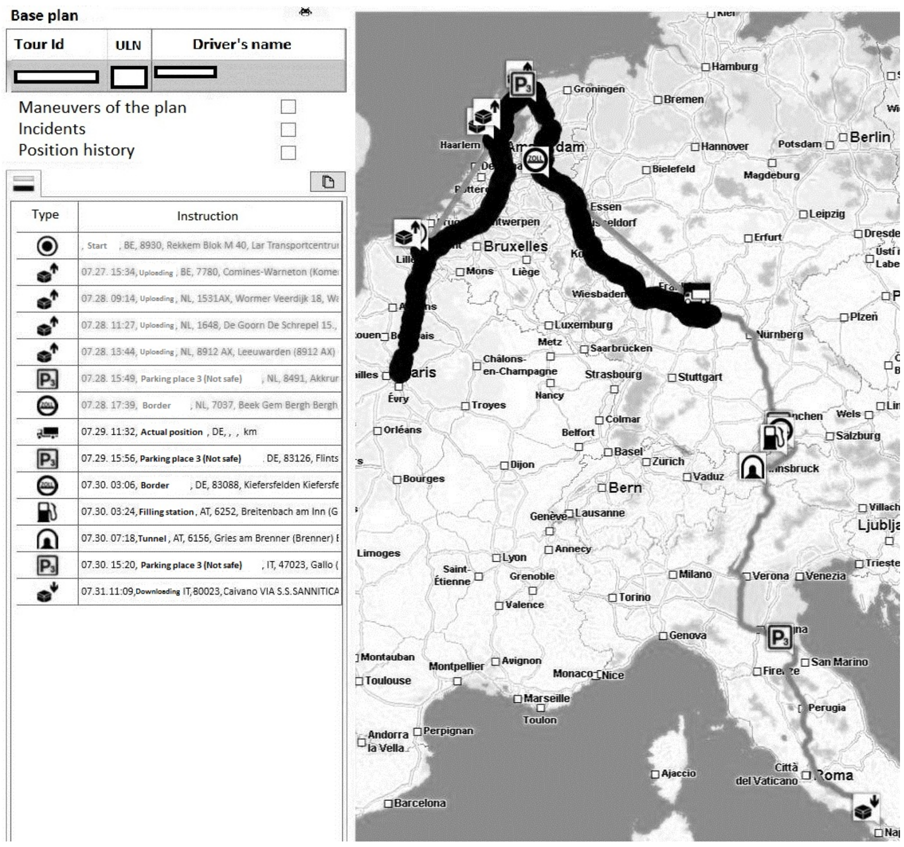
Maneuvers of the plan ☐
Incidents ☐
Position history ☐

Type	Instruction
	06.02.07:27, Uploading ,ES,46370, Chiva Partida Rosafina s/n
	06.04. 08:41, Actual position , ES, , , km
P3	06.04. 14:34, Parking place 3 (Not safe) , ES, 17600, Figue
	06.04.23:53, Filling station ,ES, 17700, La Jonquera Shell|, 1
ZOLL	06.05. 01:03, Border , ES, 17709, Le Perthus Le Perthus/
P3	06.05. 10:16, Parking place 3 (Not safe) , FR, 18570, Trouy
ZOLL	06.06. 04:38, Border , FR, 59780, Baisieux Baisieux/E42/
	06.06.04:44, Filling station ,BE, 7522, Tournai (Hertain) G&V,
ZOLL	06.06. 05:49, Border , BE, 7522, Tournai (Blandain) Tour
	06.06. 07:43, Ferry , FR, 62100, Calais Port of Calais|, 128
P3	06.06. 11:12, Parking place 3 (Not safe) , GB, CT214, Hyth
	06.08. 07:06, Tunnel, GB, RM203, Grays (Waterglade Industr
	06.08.08:00, Unloading ,GB,CM37AH,Chelmsford mashbury

Fig. 6 Parts of the interface with a tour

reason of the fragmentation is the large number of border crossings.
Figure 8 is a good example for a refueling behaviour, too (an example can be seen in Figure 9 as well). Most of the time it is more expensive to refuel in Great Britain so we should plan a full refueling before taking a ferry to cross the English Channel.
We made it possible to view the results in Google Earth, too. Figure 9 shows a picture with three tours at a time. This tool is mainly for the developers to analyze the results with more internal informations. Here each node contains many information of the variables in the algorithm. Also some of the most important driver-state variables are shown to better understand the reasons of the planned tour's details.
The project is almost complete but there are many future expectations and needs for new functionalities as well and we are working on fulfilling them.

Acknowledgements The project would not have been possible without the help of the people of Nexogen.
We wish to thank András Recski for stimulating discussions and suggestions.



Base plan

Tour Id	ULN	Driver's name
[redacted]	[redacted]	[redacted]

Maneuvers of the plan ☐
Incidents ☐
Position history ☐

Type	Instruction
◉	, Start , BE, 8930, Rekkem Blok M 40, Lar Transportcentru
⬆	07.27. 15:34, Uploading , BE, 7780, Comines-Warneton (Kome
⬆	07.28. 09:14, Uploading , NL, 1531AX, Wormer Veerdijk 18, Wa
⬆	07.28. 11:27, Uploading , NL, 1648, De Goorn De Schrepel 15.,
⬆	07.28. 13:44, Uploading , NL, 8912 AX, Leeuwarden (8912 AX)
P₃	07.28. 15:49, Parking place 3 (Not safe) , NL, 8491, Akkrur
⊖	07.28. 17:39, Border , NL, 7037, Beek Gem Bergh Bergh
🚚	07.29. 11:32, Actual position , DE, , , km
P₃	07.29. 15:56, Parking place 3 (Not safe) . DE, 83126, Flints
⊖	07.30. 03:06, Border , DE, 83088, Kiefersfelden Kiefersfe
⛽	07.30. 03:24, Filling station , AT, 6252, Breitenbach am Inn (G
∩	07.30. 07:18,Tunnel , AT, 6156, Gries am Brenner (Brenner) E
P₃	07.30. 15:20, Parking place 3 (Not safe) , IT, 47023, Gallo (
⬇	07.31.11:09,Downloading IT,80023,Caivano VIA S.S.SANNITICA

Fig. 7 A tour with visible historical locations

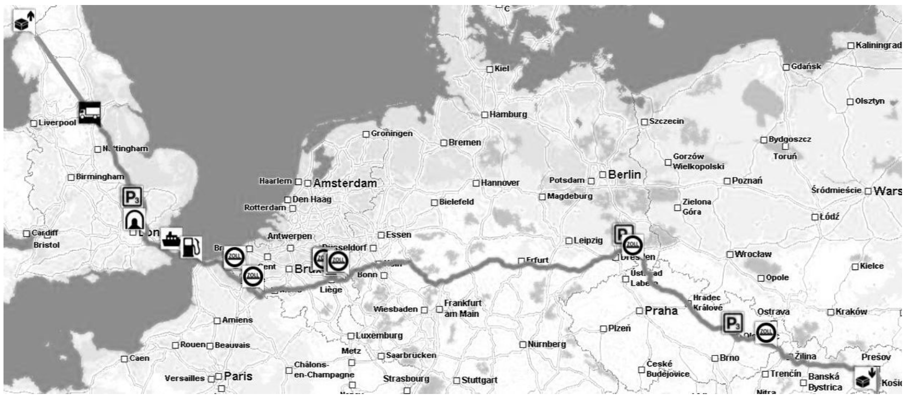


Fig. 8 A tour with many border crossings

References

1. Adamski, A. (2007). Integrated transportation and logistics systems, ITS ILS, 7:46-53.

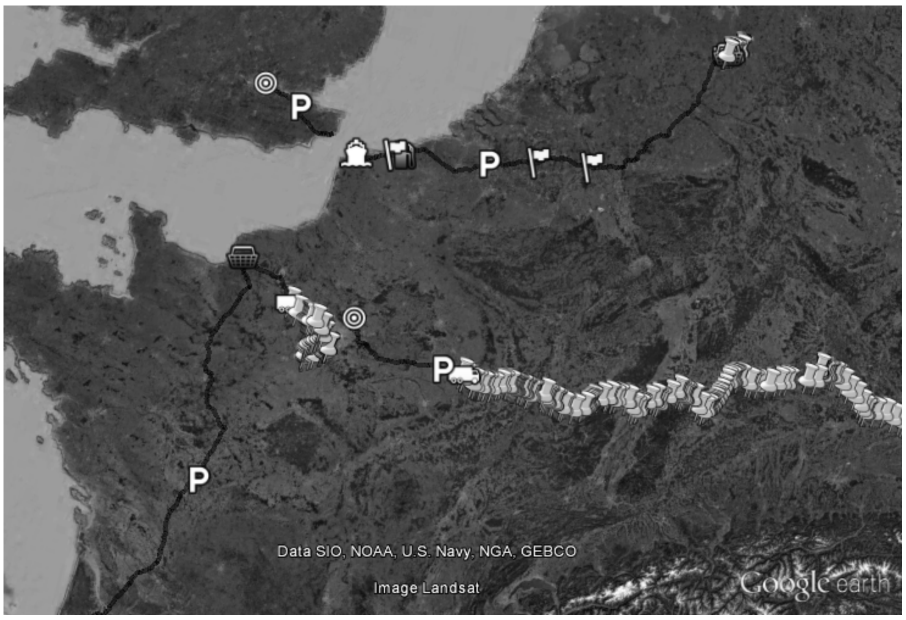


Fig. 9 Three tours represented in Google Earth

2. Carter MW, Price CC (2000) Operations research: a practical introduction, Crc Press.
3. Chen L, Langevin A, Lu Z (2013) Integrated scheduling of crane handling and truck transportation in a maritime container terminal, European Journal of Operational Research, 225(1):142-152
4. Feng CW, Cheng TM, Wu HT (2004) Optimizing the schedule of dispatching RMC trucks through genetic algorithms, Automation in Construction, 13(3):327-340
5. Floyd RW (1962) Algorithm 97: Shortest path, Communications of the ACM 5(6):345
6. Giaglis GM, Minis I, Tatarakis A, Zeimpekis V (2004) Minimizing logistics risk through real-time vehicle routing and mobile technologies: Research to date and future trends, International Journal of Physical Distribution and Logistics Management, 34(9):749-764
7. Kozan E, Preston P (1999) Genetic algorithms to schedule container transfers at multimodal terminals, International Transactions in Operational Research, 6: 311-329
8. Lawler EL, Wood DE (1966) Branch-and-bound methods: A survey, Operations Research, 14(4):699-719
9. Miao Z, Lim A, Ma H (2009) Truck dock assignment problem with operational time constraint within crossdocks, European Journal of Operational Research 192:105115
10. Schönsleben P (2011) Integral Logistics Management: Operations and Supply Chain Management Within and Across Companies, Fourth Edition, Series on Resource Management
11. Takeyasu K, Kainosho M (2014) Optimization technique by genetic algorithms for international logistics, Journal of Intelligent Manufacturing, 25(5):1043-1049
12. Toth P, Vigo D (2001) The vehicle routing problem, Society for Industrial and Applied Mathematics
13. Warshall S (1962) A theorem on Boolean matrices, Journal of the ACM 9(1):1112
14. Zäpfel G, Wasner M (2002) Planning and optimization of hub-and-spoke transportation networks of cooperative third-party logistics providers, International Journal of Production Economics, 78(2):207-220