SICS Technical Report 2017:07

DOIT WP4 Report on Planning and Optimization

Björn Bjurling
RISE SICS AB

September 2017
1 Introduction

The objective of WP4 in DOIT is to develop design methodologies for data-driven optimization models for the optimization of time and cost in heavy-duty road-based transportation. WP4 should as a part of its objective formalize the decision problems given by the project proposal. This document reports on the first step towards that part: the formalization and implementation of an optimization model for the so-called assignment planning use case.

For finding optimal alternatives among a set of candidates with respect to time and cost, it is in general needed assessments of time and cost for the candidates. An important idea in DOIT is to base the initial assessment of time and cost for the candidates on data-driven models. These models are developed in WP3 and are described in reports D3.1 and D3.2 in the project. A further important input to the optimization problems is information with the potential to constrain the solution. In the case of this report, i.e. for the assignment planning use case, this kind of input comes from transport assignments. For this report, as for the work in project to this date, transport assignment data has been extracted from operational data (using a model also described the report D3.1). A drawback with using extracted transport assignment data for optimization is that not only the quality of, but also the choice of model for, the optimization becomes dependent on the quality of the extraction. The model described in Section 3.1 is adequate for the given problem formulation and the (extracted) assignment data. We shall in the final report of WP4 show that the present model can be used as a stepping stone for building extended models for transport assignment data of quite different characteristics. These extensions will crucially also lead to the fulfillment of the objective of designing methodologies for data-driven optimization.

A third input category we have taken into account is that of driving regulations. Driving regulations constrain the solutions further and add an important aspect of realism to the model. In this report we work with a simplified version of driving regulations: i) drivers are not allowed to work more than 9 hours per day; and ii) drivers are not allowed to drive more than 4.5 hours without a break of 45 minutes.

The rest of the report is structured as follows. Section 2 describes the assignment planning use case and gives a formulation of the problem. Here we also introduce some notation. Section 3 first introduces a formal optimization model for the transport assignment planning use case, and then goes on to describe the demonstrator implementation of it. The section concludes with a discussion including some notes on the complexity of the model. In Section 4, we map out and discuss further steps that will be taken in WP4 and what to expect from the second and final WP4 report.
2 The Assignment Planning Use Case

The project has formulated a use case called Transport Assignment Planning. It is closely related to the well-studied Vehicle Routing Problem (VRP)\[3, 4\]. It differs from VRP in the sense that the optimal routes are assumed to be given as input to the problem. The focus is instead on finding the combinations of vehicles, drivers, routes, and assignments with the least overall cost.

In the (transport) assignment planning use case we are given a set of transport assignments and a carrier. We are asked to find an association of vehicles, drivers, and routes to the transport assignments minimizing the total cost for the carrier carrying out the transport assignments.

The present formulation can be extended in several ways for increasing the realism of the model. Such extensions include finding optimal departure times and optimal speed profiles. In this report we have modelled optimal departure times.

In the formulation of the problem, we shall make assumptions restricting the complexity of the model. The assumptions have been chosen for the purpose of preserving both realism and adequacy of the results from WP4 within the timeframe of the project. Most assumptions are made as a part of the modelling process and should be clear from the exposition below.

2.1 Problem Formulation

In this section we formulate an optimization problem based on the transport assignment use case.

A (transport) assignment specifies two locations: pickup and dropoff\(^1\); two timepoints: earliest pickup time and latest dropoff time; and cargo characteristics. Currently, cargo is characterized only by its weight.

A carrier is assumed to be made up by its vehicles and its drivers, denoted collectively below by \(V\) respectively \(D\). Vehicles are characterized by location, type, and capacity. Drivers are characterized by location and driving regulations. Each driver and vehicle is associated with a location called its base.

To each transport assignment \(i\), we assume that there is associated a set of routes connecting the pickup and dropoff locations of \(i\). In the initial formulation of the assignment planning use case, we have assumed that the given routes are optimal with respect to time or distance. This is for allowing a focus on the matching problem in the use case rather than on a vehicle routing problem. In particular, the project has as a working assumption that there is a unique route connecting the pickup and drop-off location for each assignment. In order to perform an optimization with respect to cost, we must assume that each route has an associated cost (in terms of time or fuel consumptions). In DOIT, it is the models developed in WP3 that gives the associated costs to each route.

Let \(A\) be a set of assignments and \(V\) and \(D\) be the sets of vehicles and drivers belonging to some carrier \(K\). A plan for \(A, V,\) and \(D\) maps each assignment

\(^1\)dropoff location may also be called delivery location
a ∈ A to a tuple of the form (v, d, r, t) where v identifies a vehicle in V, d identifies a subset of drivers included in D, r identifies a route, and t identifies a departure time. Note that in general v may point to a set of vehicles but we make the assumption that an assignment is carried out by exactly one vehicle and thus that no reload is made (as for example when a truck leaves a trailer halfway to be delivered at the dropoff location by a second truck). We assume in this formulation that there may be more than one driver carrying out an assignment (albeit with the same vehicle). The transport assignment data used as input for the modelling in Section 3.1 did however not necessitate the use of more than one driver. Consequently, both the model and the implementation in Section 3 are made under the assumption that an assignment is carried out by exactly one driver\(^2\). (We call this the unique driver assumption below). Further, we do not assume that vehicles and drivers need to return to base in the end of the shift. By these assumptions, we avoid overly complicated optimization models in the initial stages. We shall for the final report lift the assumptions and for example study transport assignment data that require more than one driver per assignment.

Let \(\Pi_{A,K}\) be the set of plans for a set \(A\) (of assignments) with vehicles and drivers from a carrier \(K\). (we write \(\Pi\) when \(A\) and \(K\) are given by context.) The cost for a plan \(p \in \Pi\) depends as a function on how each assignment is mapped by the plan. Thus, our problem is to find a plan in \(\Pi\) with the minimal cost. As long as \(\Pi\) is non-empty, assuming non-negative costs, we are (trivially) guaranteed the existence of such an element (not necessarily unique though).

The characteristics (and the size) of \(\Pi_{A,K}\) depends on the degrees of freedom for the choices of the terms in \((v, d, r, t)\) associated with the assignments in \(A\). The degrees of freedom are in turn determined by constraints and assumptions made in the formalization of the problem. Thus different formalizations can give rise to different solutions. We can compare the quality of such solutions. The crucial part of modelling is to make \(\Pi\) large enough to contain a solution with high enough quality, while not make making it too large. If \(\Pi\) is too large, the problem of finding the optimal plan can become practically impossible.

3 Formalization and Implementation

In this section we first give a formalization of the problem. Then we describe the implementation briefly. The section ends with a discussion about the model and its complexity.

3.1 Optimization Model

The assignment planning use case assumes that there are finite non-empty sets \(A\) of assignments; \(V\) of vehicles; and \(D\) of drivers\(^3\). The problem is to assign a

\(^2\)The final report will however cover cases where more than one driver is needed

\(^3\)In a more general setting, a route should also be associated to each assignment. In this setting, it is not necessary to do so because, as remarked earlier, we assume that there is a
vehicles and a driver to each assignment while minimizing the cost for fulfilling the assignments.

\[ V \text{ set of vehicles} \]
\[ D \text{ set of drivers} \]
\[ A \text{ set of assignments} \]

We assume that there are \( N \) assignments and that each vehicle has a unique driver. The modelling of the assignment planning use case is based on the approaches in the crew pairing literature: we first construct a set of feasible pairings (fulfilling all the constraints put on the problem, such as for example driving regulations) and then use a mathematical programming approach to find an optimal set of feasible pairings\(^4\). In particular, we model the use case as a set partitioning problem \([6,7,8]\), where a solution is a set of feasible pairings that minimizes cost and covers the assignments without overlap.

**Defining the set of Feasible Pairings** The assignments determine the plans. Thus the following intuitive definitions, somewhat informally, extracts information from the assignments to be used in the optimization model.

\[ i \text{ for } i < N \text{ denotes the } i\text{-th assignment} \]
\[ \hat{a}_i \text{ earliest pickup time of assignment } i \]
\[ \hat{b}_i \text{ latest dropoff time of assignment } i \]
\[ \hat{a}_i^+ \text{ earliest pickup time of assignment } i \text{ plus 1 hour (for allowing a break)} \]
\[ \hat{a}_i^- \text{ earliest pickup time of assignment } i \text{ minus 1 hour (for allowing a break)} \]
\[ s_i \text{ pickup location of assignment } i \]
\[ e_i \text{ dropoff location of assignment } i \]
\[ \ell_i \text{ leg induced by assignment } i; \text{i.e. the route from the pickup location to the dropoff location of assignment } i \]
\[ u_{ij} \text{ empty leg from dropoff location of assignment } i \text{ to pickup location of assignment } j \]

For \( N \) assignments, enumerate the empty legs \( u_{ij} \), starting with index \( N \), and denote them by \( \ell_N, \ell_{N+1}, \ldots, \ell_{N+m-1} \) for some \( m \). Then, for \( i < N + m \), a hop is defined as a tuple:

\[ (\ell_i, v_j, d_k, a_i, \delta_i), \]

where \( \ell_i \) is a leg (an empty leg when \( i \geq N \)); \( v_j \) is a vehicle; \( d_k \) is a driver; \( a_i \) is the departure time of the leg \( \ell_i \); and \( \delta_i \) is the duration of the leg. Since we

\(^4\)See reference [1]
assume vehicles to have unique drivers, we shall accordingly write \((\ell_i, v_j, a_i, \delta_i)\) for the hop \((\ell_i, v_j, d_k, a_i, \delta_i)\), letting the driver be implicitly given by the vehicle \(v_j\). The duration \(\delta_i\) is assumed to be given as a predicted\(^5\) duration for driving from the pickup location to the dropoff location of assignment \(i\). Next follows a number of notations.

**Definition.** Let \(\tau = (\ell_i, v_j, a_i, \delta_i)\) be a hop. Then:

1. \(\ell(\tau) = \ell_i\) (the leg in \(\tau\))
2. \(a(\tau) = a_i\) (departure time of \(\tau\))
3. \(v(\tau) = v_j\) (vehicle used for \(\tau\))
4. \(\delta(\tau) = \delta_i\) (duration of \(\tau\))
5. \(b(\tau) = a(\tau) + \delta(\tau)\) is the (predicted) arrival time of \(\tau\).
6. If \(\tau\) is a non-empty hop, then \(i(\tau)\) denotes the transport assignment \(i\).
7. \(\hat{a}(\tau)\) is the earliest start of \(\tau\) and defined by the earliest pickup time of the \(i\)-th assignment in the case \(i < N\). For \(i \geq N\) the leg \(\ell_i\) is empty in which case \(\hat{a}(\tau)\) shall be undefined. (A hop with an empty leg shall be called an empty hop).

**Definition.** A *pairing* is a non-empty tuple of hops. A singleton pairing \((\tau,)\) is identified with its single hop \(\tau\).

**Definition.** Let \(p\) be a pairing:

1. \(|p|\) denotes the number of hops in (or length of) \(p\)
2. \(s(p)\) and \(e(p)\) denote the start respectively the end locations of \(p\). That is: \(s(p)\) is \(s_{i_0}\) of the first hop in \(p\), and \(e(p)\) is \(e_{i_{|p|-1}}\) of the last hop of \(p\).
3. \(a(p)\) and \(b(p)\) denote the timepoints for the departure of the first hop in \(p\) respectively the arrival of the last hop in \(p\)
4. if \(\tau\) is a hop, then \(p\tau\) denotes the extension of \(p\) with \(\tau\)
5. \(p^{-1}\) denotes the last hop in a pairing \(p\)
6. \(i(p)\) denotes the set of transport assignments covered by the (non-empty) hops in \(p\)

Finally, the notation \(loc(v_i)\) means the location of the vehicle \(v_i\). Note below that no assignments in the data required road-side breaks as long as breaks are taken at pickup or dropoff locations. With the notation defined, we are now ready to define the set of feasible pairings.

We define the set \(P\) of feasible pairings inductively as a subset of the set of all pairings. Assume that \(p\) is a feasible pairing and \(\tau\) is a hop such that

---

\(^5\)Predicted by a routing service such as Digital Reality
$b(p) \leq \hat{a}(\tau)$ and $e(p) = s(\tau)$ (That is; we assume conservatively that $p$ has to end before the earliest departure of $\tau$ and that $p$ ends at the same location as $\tau$ starts.)

1. if $\tau = (\ell_i, v_j, t, \delta_i)$ for some timepoint $t$ and $\ell(\tau)$ is non-empty, then $\tau \in P$ if $loc(v_j) = s_i$ and $\delta_i \leq 4.5$ and

   (a) $t = \hat{a}_i$ and $t + \delta_i \leq \hat{b}_i$, or
   
   (b) $t + \delta_i = \hat{b}_i$ and $\hat{b}_i - \delta_i \geq \hat{a}_i$, or
   
   (c) $t = \hat{a}_i^+$ and $t + \delta_i \leq \hat{b}_i$, or

   (verbally: the non-empty hop (a singleton pairing) $\tau$ is a feasible pairing if its vehicle is located at the start location of $\tau$ and the duration is less than 4.5 hours and

   a) $\tau$ starts at the earliest pickup time and ends before the latest dropoff time; or
   
   b) $\tau$ ends at the latest dropoff time and starts after the earliest pickup time; or
   
   c) $\tau$ starts one hour after the earliest pickup time and ends before the latest dropoff time)

2. if $\tau = (\ell_i, v_j, t, \delta_i)$ and $\ell(\tau)$ is empty and assuming that $\ell_i = u_{xy}$ for some $x, y$, then $\tau \in P$ if $loc(v_j) = e_x$ and $\delta_i \leq 4.5$ and

   (a) $t = \hat{a}_y$, or [the code has equality in this and the following two lines.]
   
   (b) $t + \delta_i \leq \hat{a}_y^-$, or

   (verbally: the empty hop $\tau$ is a feasible pairing if, for some transport assignments $x$ and $y$, its vehicle is located at the end location of assignment $x$ and the duration is less than 4.5 hours and

   a) the vehicle arrives at the location for the transport assignment $y$ at the pickup time for assignment $y$.
   
   a) the vehicle arrives at the location for the transport assignment $y$ an hour before the pickup time for assignment $y$ (allowing a break before the next leg))

3. if $\ell(\tau)$ is an empty hop, then $p\tau \in P$ whenever the following conditions hold

   (a) $v(\tau) = v(p^{-1})$
   
   (b) $\ell(p^{-1})$ is non-empty
   
   (c) there is an assignment $\alpha \in A$ with pickup location $e(\tau)$ such that $\alpha \not\in i(p)$ (that is, the assignment $\alpha$ has not already been dealt with in $p$)
(d) \( b(\tau) - a(p^{-1}) < 4.5 \) or \( a(\tau) - b(p^{-1}) \geq 1 \).

(verbally: a feasible pairing \( p \) can be extended with a hop to form a feasible pairing whenever

(a) \( \tau \) has the same vehicle as the last hop of \( p \)
(b) either \( \tau \) is non-empty or \( p \) doesn’t end with an empty hop
(c) \( \tau \) ends at a location which is the pickup location for a transport assignment that is not covered by \( p \)
(d) either the sum of the durations of \( \tau \) and the last hop in \( p \) is less than 4.5 hours, or \( \tau \) starts at least one hour after the end time of \( p \)

4. if \( \ell(\tau) \) is non-empty, then \( p\tau \in P \) if

(a) \( v(\tau) = v(p^{-1}) \)
(b) \( i(\tau) \not\in i(p) \)
(c) \( b(\tau) - a(p^{-1}) < 4.5 \) or \( a(\tau) - b(p^{-1}) \geq 1 \).

(verbally: covered be verbal explanation to item (3) above)

**Model**

In this section, we build a model of the problem using the pairings constructed in the previous section. In particular, we build a model suitable for mixed-integer programming. (The model is summarized in a more terse way in the next section.)

The goal is to minimize the total cost for carrying out all the transport assignments in our set \( A \). The total cost is the sum of the costs of the elements in the optimal set of pairings we are aiming to find. Without going into detail, a monetary cost for a pairing can be accrued from fuel and salary costs in connection to driving from the start to the end locations for the hops in the pairing. Note that the costs used in the model are assumed to be obtained from the predictive analysis performed in WP3 based on historical transportation data and performance records for the vehicles in the set \( V \). The present model does not depend on the type of cost: it can be monetary costs or time.

In the sequel, let \( P \) be the set of feasible pairings. For \( p \in P \), let \( c_p \) denote the cost of \( p \).

We shall define the objective function. For that, observe that for each \( p \in P \), \( p \) will either be an element of the solution or not. Therefore, we define, for each \( p \in P \), a decision variable \( x_p \) which takes the value 1 if \( p \) is in the solution and the value 0 if not. Thus, a valuation of the variables \( x_p \) for \( p \in P \) determines a subset of \( P \) (i.e. \( \{ p \in P | x_p = 1 \} \)). Note that if \( x_p = 0 \) then the cost of \( p \) doesn’t add to the total cost, so clearly \( \sum_{p \in P} c_p x_p \) will give the cost of the set of pairings for which the decision variables valuate to 1. The goal is to minimize the total cost, which is achieved by

\[
\text{minimize } \sum_{p \in P} c_p x_p
\]
as objective function.

The first constraint is that each assignment should be covered by exactly one pairing in the solution. In order to formalize this, let \( p_{ip} \) be a binary variable where \( a_{ip} = 1 \) if the pairing \( p \) covers the assignment \( i \in A \), and where it is 0 otherwise. (Note\(^6\) that when \( P \) is given and constructed as in the previous section, \( a_{ip} \) is constant relative to \( P \).) Thus we can formalize the constraint as:

\[
\sum_{p \in P} a_{ip} x_p = 1 \quad i \in A
\]

The second constraint is to limit the number of vehicles for each pairing to at most 1. Let therefore \( v_{ip} \) be a binary variable that takes the value 1 if the \( i \)-th vehicle occurs in a hop in \( p \) and 0 otherwise. Note again that \( v_{ip} \) is constant relative \( P \). Thus

\[
\sum_{p \in P} v_{ip} x_p \leq 1 \quad i \in V
\]

Third, we must make explicit that the decision variables can vary in the set \( \{0, 1\} \):

\[
x_p \in \{0, 1\} \quad p \in P
\]

Finally, we make a restriction, for keeping the complexity low, that each pairing must cover less than 4 assignments (of course, this number can be varied according to needs, time, and computing power): let \( a_p \) denote the number of assignments covered by pairing \( p \), then

\[
a_p < 4 \quad p \in P
\]

Note that further assumptions and restrictions on the model (for example those for driving regulations) have been imposed already in the definition of the set of feasible pairings.

**Summary of Model**

- \( P \) set of feasible pairings
- \( x_p \) decision variable for pairing \( p \)
- \( c_p \) cost for pairing \( p \in P \)
- \( a_{ip} \) assignment \( i \) is covered by pairing \( p \)
- \( v_{ip} \) vehicle \( i \) is used in pairing \( p \)
- \( a_p \) number of assignments covered by pairing \( p \)

\(^6\)The reason for this remark is that the constancy of the term is a necessary condition for the linearity of the model.
\[
\begin{align*}
\text{minimize} & \quad \sum_{p \in P} c_p x_p \\
\text{subject to} & \quad \sum_{p \in P} a_{ip} x_p = 1 \quad i \in A \quad (1) \\
& \quad \sum_{p \in P} v_{ip} x_p \leq 1 \quad v_i \in V \quad (2) \\
& \quad x_p \in \{0,1\} \quad (3) \\
& \quad a_p < 4 \quad p \in P \quad (4)
\end{align*}
\]

### 3.2 Implementation

DoIT aims for illustrating the results in its workpackages through implementation and integration into a demonstrator. In this subsection, we give a brief overview of the implementation of the optimization model. Let us for subsequent brevity fix the notation M for implementation of the optimization model.

The code is written in Python 2.7 and it runs on the standard Python interpreter\(^7\). In particular, we have used the Python module PuLP\(^8\) for modelling and for calling a solver. Pulp supports several different solvers including the commercial CPLEX\(^9\) and the free Lpsolve\(^10\). In the problem based on the sample data described in this report, Lpsolve and CPLEX perform equally well. However, for larger problems, the recommendation is to use CPLEX. In the forthcoming final WP4 report, we will be able to show empirical results for comparing the two solvers on project related problems. There are also resources online for similar comparisons [2].

There are three inputs to M: the assignments; vehicle data; and cost estimates for each trip that can be taken for fulfilling the assignments.

1. Assignments are given via a json interface
2. Vehicle data is given as a csv-file
3. Cost estimates are given via calls to a routing service and the fuel cost prediction model developed in the project

In subsequent versions of M, we will also consider driver data as input. Currently, we make a unique association between a vehicle and its driver, so there is no point in including driver data. Note though that M (and the optimization model) nonetheless take driving regulations into account.

---

\(^7\)www.python.org
\(^8\)https://pypi.python.org/pypi/PuLP
\(^9\)http://www-03.ibm.com/software/products/en/ibmilogcpleoptimistud/
\(^10\)https://sourceforge.net/projects/lpsolve/
The output of M is a json-file representing the optimal plan for carrying out the given assignments. The implementation is basically an imperative version of the declarative construction given in the previous subsection. The implementation consists of four steps depicted by the diagram in Figure 1 and detailed below.

**Data preparation** Input is read and stored internally in the module M. Based on the stored assignments, we create a database (a Python dict, to be precise) containing distance and predicted fuel consumption for every possible leg in the problem.

**Generation of Feasible pairings** We define the set of feasible pairings by first creating all possible legs based on the stored data. The legs are subsequently combined to form the feasible pairings. Note that it is possible to put restrictions on which legs we consider possible. In this implementation, following the model as described above, we include two variants of each leg: one that starts at the earliest time for pickup and one that ends on the time for latest delivery. The restrictions will affect the complexity of the model and it is important to choose restrictions that limit the complexity while not significantly reducing the possibility of finding a solution of high quality.

The pairings are generated inductively in accordance with the model described above with the legs just described as base case.

**Optimization** We use the syntax given in PuLP for setting up the optimization model, again as described above in Section 3.1, resulting in an instance A of the PuLP class OptimizationModel. The class has methods for calling external solvers whenever the solvers are installed on the system running where the instance resides. For this report, we have used both Lpsolve and CPLEX as solvers, as mentioned above.
Note that PuLP merely provides a *pythonic* syntax for encoding optimization models. There are several other (semi-)formal ways of encoding optimization models.

**Formatting of results** The format of the result is determined to a great extent by the modelling and the tools used when performing the optimization. In general, this is often different from the form expected by the user or the application (in our case the demo front end). The optimization/planning module returns the results of the computation reformatted according to specification of the user.

### 3.3 Solution and Complexity

In this subsection we make some theoretical and practical remarks about the solution to the problem.

First we note that the optimization model we have presented is a variant of the Set Cover Problem, which is NP-complete. Therefore, we can expect to find the best solution only under limited circumstances. As the problem scales up, it becomes virtually intractable. Factors in our model that affect the scale of the problem include:

1. number of assignments and vehicles (and drivers)
2. the (local) density of assignments during the planning period
3. degrees of freedom in departure time (allowed every hour, minute, or second)
4. degree of realism in the model (e.g. inclusion of speed profiles)
5. the number of assignments allowed in a pairing

The enumerated factors all contribute to determine the size $S$ of the set of feasible pairings. The size $S$ in turn corresponds exactly to the number of decision variables in the optimization model. The number of decision variables is a good indicator of the complexity of the restricted problem at hand. The model described in Section 3.1 has a limited number decision variables (ca 10,000) allowing the computation to be made on a laptop computer in seconds or up to a minute. However, increasing for example the number of allowed assignments per pairing by 1, increases the number of decision variables by an order of a million. This would make the computation much more time and resource consuming (on a laptop the time required for optimization increased to more than 4 hours.) It should thus be clear the importance of knowing for certain that we obtain a significantly better solution when we want to increase the depth of the investigation. Continuing increasing the depth of the investigation further, which may be desirable as a way of increasing the realism of the model, we eventually reach a point where all feasible pairing cannot be generated efficiently. For real-world scale problems, this is usually the case and there are many techniques for
3.4 Empirical Results

The model and the implementation described in subsections 3.1 and 3.2 have been tested on the set of assignments extracted from the data provided for the project. We tested four setups based on choice of fuel consumption model and the choice of locking all or none decision variables. We recorded for each case, the total driving time and the total fuel consumption. The results of the test are given in Table 1. If we accept total driving time as an indicator of salary cost, we thereby cover the two principal sources of cost: salary and fuel. The results with all decision variables locked can be seen as a reference case as no optimization was made for that. For tests with the all decision variables unlocked, we tested minimizing fuel consumption and minimizing time. In the case of the DoIT fuel model, note that minimizing fuel and minimizing time has almost the same effect with respect to fuel and time costs. Regardless of objective function, it seems that both calculations have found the (same) global minimum. The global minimum can easily be reached in the optimization process when the number of alternatives is low. That it happens here is likely due to the sparsity of the data as the number of assignments was low.

We tested also the effect on the complexity by restricting the problem in various ways. First, even in our small problem, the effect of allowing pairings to have initial empty legs had a huge impact on the complexity. In our setting, allowing pairings with empty first legs increased the number of pairings by a factor of 28! (the factorial of 28), rendering the computation infeasible on a laptop. Disallowing empty first legs and varying the restriction on the number of assignments per pairing we got the following results:

- **max 2 assignments** 1372 decision variables
- **max 3 assignments** 24496 decision variables
- **max 4 assignments** 286816 decision variables

Thus, increasing the number of allowed assignments per pairing by 1, increases the complexity of the problem with a factor of 10, roughly.

Table 1: Empirical results from test with DOIT cost models and optimization addressing it, including various forms of heuristic search in the space of pairings (see [9] for an example of approaching this problem). We leave the discussion of handling such cases for the final report.
3.5 Discussion

One may wonder why we chose to model the problem as a variant of a NP-complete problem in the first place. Note first that planning problem in general are NP-complete. Further, the choice of the Set Cover formulation allows sufficiently high degree of realism for enabling us to capture the problem described in the Transport Assignment planning use case—in particular when the problem formulation is scaled up. On the other hand, restricted variants of the use case can be modelled with considerably less decision variables and thus lower complexity. For example, if we consider only to match vehicles to transport assignments, we can model the problem as a minimal cost flow problem. This restriction is described in the forthcoming and final report from WP4. In the clean minimal cost flow problem, the number of decision variables is even 0. However, it is rarely the case that real-world problems can be modelled as clean minimal cost flow problems.

4 Conclusion and Further Work

We have with this document provided the first report of two for planning and optimization in the DoIT project. In the report, we have described an initial model for the transport assignment use case and discussed its complexity. We also described the implementation of the model, which is the contribution from WP4 to the demonstrator of the project.

The model and its implementation was tested with a set of assignments extracted from one month of operational data from Scania trucks around Europe. It is likely that the extraction tool captured only a fraction of all assignments, as the input set of assignments to the planner was very sparse in time. We shall in the sequel of the project make sure to test the models on more dense sets of assignments.

Below, the model referred to is the optimization model described in this document. The work ahead in the planning and optimization work package is planned as follows:

1. Up to reference group meeting in June 2017
   (a) Study and prepare acquired data from a commercial carrier
   (b) Extend model wrt the new data
   (c) Implement extended model
   (d) Test extended model with suitable use cases

2. August to December 2017
   (a) Investigate industrially relevant extensions of the problem (e.g., wrt fleet size, assignment volume, or cargo characteristics) and implications for the solution and modelling (in relation to the model developed in the project)
(b) Survey and evaluation of optimization methods wrt selected problem types with focus on Scania’s customers’ needs

(c) Final report. This will include a model for a second use case as well as further remarks on the model in the present report.

References


