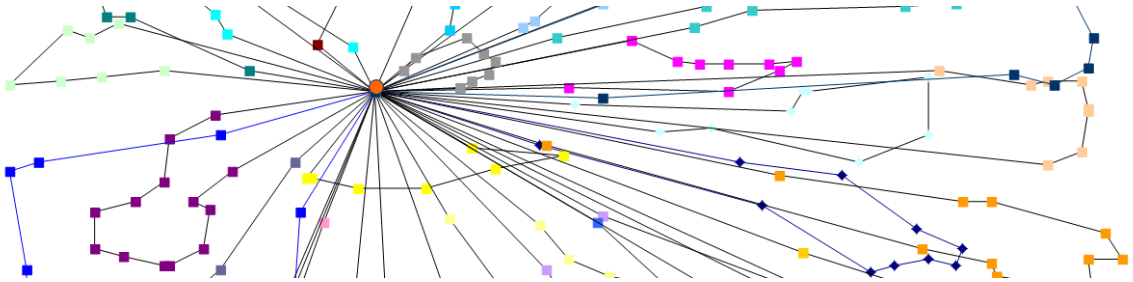
Spider

A software component for optimized transportation planning and vehicle routing

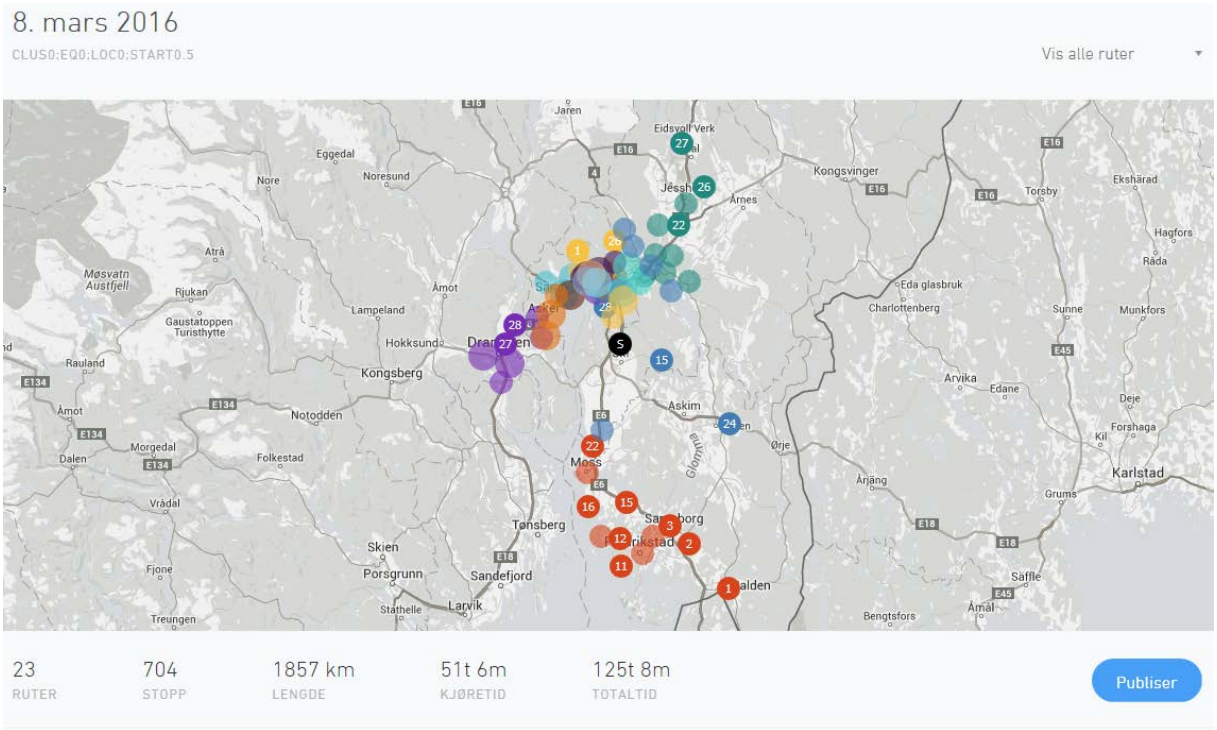


Spider is a software tool for planning transportation or service operations for a fleet of vehicles. By applying modern mathematical optimization techniques, it offers a huge potential for savings in transportation costs, increased revenues, and higher efficiency. As well as dynamic fleet management and fastest route calculations, Spider can be used in transportation network design and fleet composition. Good co-ordination of transportation activities through optimization can improve customer service and give both environmental and economical savings.

Applications

Spider (http://www.sintef.no/en/software/spider/) has been developed by SINTEF Digital, in close R&D collaboration with industrial users over many years. The result is a versatile and robust software product that easily extends to meet special requirements, making it an ideal tool as a planning kernel for apps and services.

As an example, Spider is integrated in the solution of Distribution Innovation AS (DI) for planning of carrier routes in the media product business, and in DI's solution Plan & Go for more dynamic transportation planning.



Ruter

RUTE	TILDELT	STOPP	LENGDE	START-FERDIG	KJØRETID	TOTALTID	MIN. PR. STOPP
Hetti, Netti og Letti	Tildel	22	258 km	15:29-21:42	5t 6m	6t 12m	16,9
Rikerud	Tildel	35	171 km	15:48-21:57	4t 20m	6t 10m	10,6

Screenshot from the Plan & Go solution of Distribution Innovation AS. Spider is the optimization engine.

Mathematical basis

The algorithms used for optimal planning in Spider are based on state-of-the-art approximation methods - so-called metaheuristics - for the solution of Vehicle Routing Problems (VRP), and efficient optimization methods for the Dynamic Shortest Path Problem (SPP). The VRP is about how to best assign customer requests to each of the available vehicles, and for each vehicle, what is the best sequence of customer visits. The goal is to find the best transportation plan according to revenues and economical and environmental transportation costs. The plan must obey constraints, such as vehicle capacity and opening hours at the customers.

Technical components and interfaces

C++ library

At the core of Spider is a C++ class library. The library is available for all recent Windows platforms. A port for FreeBSD is also available, and ports to other Unix variants will be provided on request.

Spider Server

Spider Server is a VRP optimization server built on top of the Spider core library. Running this server, the client may add, modify, examine and remove VRP instances and use the tool to calculate optimized plans for each problem instance.

Programming against Spider Server can be done using a client library in Java or .Net. The client library exposes classes that represent VRPs and associated objects. Communication with the server is handled transparently by the client library, over a TCP/IP connection.

Model

Spider solves an extended version of a problem type known as the Vehicle Routing Problem (VRP). The problem involves planning how to use a set of transportation resources to fill a set of demands for transportation.

Tours

The resources available for transportation are expressed as Vehicles, Drivers and Tours.

- A Vehicle represents the physical vehicle that will transport the goods. Its most important attribute is its capacity, which may be expressed in any number of dimensions. E.g. a vehicle may have a capacity of 800kg, 10 cubic meters and three pallets, and the capacity may not be exceeded in any dimension. The number and identity of dimensions is decided by the user.
- A Driver represents the person driving a vehicle. Drivers may be used in conjunction with constraints.
- A Tour represents a certain time interval in which a vehicle is available for transportation. The tour must be assigned a vehicle and a driver that perform the tour. It is given a start location at which to begin at a given start time, and an end location, where it must finish before a given end time. A given driver or vehicle may be used in several tours, provided that the tours have non-overlapping time intervals. It is possible to relax the requirement for non-overlapping time intervals. Tours that use the same driver and vehicle may be linked up. This forces the second tour to start after the first one ends, even if this is not implied by the time intervals.

Orders

An order represents a demand for transportation of some goods. It consists of one or two tasks, which represent something that has to be done at a certain location as part of some tour. Usually, but not necessarily, the things to be done are picking up goods at their origin and delivering them at their destination. Each task has a location. The order has a size, describing the amount of goods, which is given in the same dimensions as the capacities of vehicles. There are four types of orders in Spider:

- Delivery orders must be loaded at the start of a tour (usually the depot) and may be delivered at any point in the tour.
- Pickup orders may be picked up at any point in the tour, and must be unloaded at the end of the tour (usually the depot).
- Direct orders are the most general type, which may be picked up and delivered at any point in the tour (but in that order).
- Single visit orders do not have a size, and have only a single associated task. These orders do not represent goods to be transported, but the fact that a certain location should be visited.

Plans

Once the orders and tours of a VRP have been defined, you may instruct Spider to create a plan for the VRP. A Plan assigns a (possibly empty) sequence of tasks to each tour, thus determining which goods and what route each tour should take. If there are insufficient resources, some orders may be left unserviced, i.e. their tasks are not performed by any tour.

You may easily inquire about any aspect of a plan, e.g. what tour services a certain order, when is a certain task performed, or how much goods do a certain vehicle carry at a certain point.

Constraints

You may add constraints to the VRP to restrict what plans are acceptable. In Spider, the following types of constraint are available:

- Time window constraint. Each task may be given a time window, which determines the earliest and latest time that the task may be performed. Multiple time windows are also supported, i.e. the times when a task may be performed may be a set of disjunct intervals.
- Capacity constraint. This constraint ensures that at any time, the total size of the goods on each vehicle does not exceed the vehicle's capacity.
- Compatibility constraints. You may define compatibility rules that tell which orders/tasks may be performed by which drivers/tours/vehicles.
- Same tour constraint. You may specify pairs of orders that are not allowed to be serviced by different tours.
- Different tour constraint. You may specify pairs of orders that are not allowed to be serviced by the same tour.
- Alternative locations. It is possible to specify alternative locations for a given task. It is then up to the optimizer to select the location that gives the best plan. If both tasks of an order have alternative locations, it is possible to set a flag on the order that forces corresponding locations in the two tasks, i.e. both tasks must use the first or second or alternative location. It is also possible to specify alternative locations for the start and end of a tour, and a similar flag can be set to make the start and end locations of the tour correspond.
- Alternative time windows. It is possible to specify alternative time windows (each of which may be multiple) for a given task. It is then up to the optimizer to select the time window that gives the best plan. It is possible to specify sets of tasks (from different orders) whose selected time windows must correspond. This can be used e.g. to say that a given set of orders must be serviced on the same day.
- Capacity over several tours. In addition to the regular constraint that the capacity of a vehicle must not be exceeded at any point, it is possible to specify a set of tours over which the total picked up and delivered must not exceed a given quantity.

Objectives

By setting the objectives of the VRP, you define how the cost of a given plan is calculated. When optimizing, SPIDER searches for a plan with the lowest possible cost. The following objectives are available:

- Travel cost objective. This objective measures the cost of travelling between locations. The cost may be per distance or time unit, or a combination.
- Tour cost objective. For each tour, you may specify the cost of actually using the tour (that is, putting one or more tasks on it) and the additional cost per order serviced by the tour. This may be used to encourage using one tour over another, or encourage empty or non-empty tours.
- Order serviced objective. Each order has a cost which is incurred if the order is not serviced.
- Waiting time objective. A cost per time unit may be assigned to idle time in the plan caused by waiting for time windows. If desired, this cost may also be applied to the time spent performing tasks.
- Work regulations. Require that the driver have enough rest on long hauls. Work regulations within the EEA are modelled, and the objective can be configured to model many variants of these rules. The objective can report on violations of the work regulations in a plan, and can be active in optimization to avoid such violations.
- Tour leveling. Penalizes plans where tour durations vary.
- Tour overlap. Penalizes plans where tours overlap geographically.

You may give weights to the different objectives to set their relative importance.

VRP and plan dynamics

The main function of Spider is to automatically create and optimize a plan, i.e. modifying it so as to minimize the plan's objective. You may also manually modify plans, e.g. move tasks, insert and remove orders. Plans can be duplicated, and any number of plans may be kept in memory simultaneously. Plans can be saved to file and restored. At any point, you may instruct Spider to further optimize an existing plan.

Having obtained a plan, you may lock part of it. Spider will not change a locked aspect of a plan when optimizing, but manual alterations are still permitted. Locks are typically used for parts of the plan that have been or will shortly be executed. The following locks are available:

- Task time lock. This lock fixes the time when the task will be, or has been, performed.
- Order assignment lock. If an order is serviced by a tour, you may use this lock to prevent the order being moved to another tour or unserviced.
- Order unassignment lock. You may lock an order that is unserviced in the plan to remain so.
- Tour lock. You may lock an entire tour to prevent any modification of the tour. You may also lock the initial part of a tour, up to and including a given task. This is useful for tours that are being executed.

Any existing lock may be removed.

Spider allows you to modify the VRP after having created plan(s) for it. You may change virtually anything, e.g.:

- Add and remove objectives, constraints, orders and tours
- Change time windows, locations and other attributes
- Change the topology

After having changed the VRP, you must ask Spider to update existing plans. Spider will modify a plan as little as possible to make it consistent with the new VRP definition.

Topology

Spider has a topology module, whose task it is to determine the time and cost for travelling between locations. There are two topology models available: Euclidean topology and road topology.

Euclidean topology

This is a simple topology that measures Euclidean distance. Locations are given as a pair of coordinates (x,y), and distance is calculated using Pythagoras' formula. Travel time is calculated using a configurable average speed.

Road topology

The road topology calculates travel time by finding the cheapest path in a network of roads and intersections. The following attributes and restrictions are available:

- Roads have a length and a speed limit, which are used to determine distance and travel time. If available, average speed may be used instead of the speed limit.
- Roads may be one way.
- Roads may have a physical barrier in one or both ends.
- Intersections may have turning restrictions, preventing a vehicle arriving from one road turning onto a certain other road.
- There may be restrictions on the height, length or weight of a vehicle for using a certain road.

Spider can import road network data in several formats:

- Elveg, in SOSI- or Shape-version. Data covering Norway are available from Geonorge
- MultiNet Shape, available from tomtom
- Navteq, available from HERE Map Data

You may define a number of vehicle classes with different characteristics. These characteristics determine per vehicle class if a road may be taken, and if so, how long time it takes to travel along it:

- The height, length and weight.
- The speed may be given as a constant and/or as a percent of each road's speed limit. The speed specification may be overridden for given areas or single roads.
- A vehicle class may be allowed to ignore physical barriers and one-way restrictions.

The cost of a road is measured using a metric, which may be any combination of distance and time. Thus you may instruct Spider to calculate shortest paths or quickest paths or something in between. The path calculation is done by an extremely efficient algorithm, which is a variant of Dijkstra's algorithm. Caching is also used to improve the efficiency.

Locations are given as a pair of coordinates (x,y), which are snapped to the closest point on a road link. It is possible, and usually desirable, to avoid snapping to road links that represent tunnels or bridges.

A special location, Anywhere, is supported. This location allows tasks to be specified that have no particular location. When a task at Anywhere is part of a tour, it will not contribute to the travel time of the tour. The task is assumed to be performed at some unspecified point between the previous and the next task in the tour.

Optimization

Spider can construct initial plans for a given VRP using the following construction heuristics:

- Sequential insertion of each order in the best place
- Solomon's construction heuristic
- The Savings construction heuristic
- Constructor - a further development of the Solomon heuristic
- Clusterer - a construction heuristics for creating a routing plan with balanced, non-overlapping routes

The construction heuristics are extended to fully support the rich, industrial VRP model of Spider. Plans may be improved using an iterative improvement algorithm based on a combination of the Iterated Local Search and Variable Neighborhood Search metaheuristics. The algorithm supports the First improvement and Best improvement strategies. Neighborhoods are created using a configurable combination of some 15 local search operators, including:

- Insert
- Relocate
- Two-opt
- Exchange
- Cross
- Tour depletion
- Three-opt
- Or-opt

The operators are extended to fully support the rich, industrial VRP model of Spider.