An effective heuristic for ready mixed concrete delivery

Verena Schmid[(1)], Karl F. Doerner[(1)], Richard F. Hartl[(1)], Martin W. P. Savelsbergh[(2)], Wolfgang Stoecher[(3)]

(1) Department of Management Science, University of Vienna, Bruenner Strasse 72, 1210 Vienna, Austria
{verena.schmid, karl.doerner, richard.hartl}@univie.ac.at
(2) School of Industrial and Systems Engineering, Georgia Institute of Technology, Atlanta, GA 30332-0205
mwps@isye.gatech.edu
(3) Profactor Produktionsforschungs GmbH, Steyr-Gleink, Austria
wolfgang.stoecher@profactor.at

1 Introduction

A medium-sized company in the concrete industry located in Upper Austria is facing the following scheduling problem on a daily basis. Concrete produced at several plants has to be delivered at customers' construction sites using a heterogeneous fleet of vehicles in a timely, but cost-effective manner. As the ordered quantity of concrete typically exceeds the capacity of a single vehicle several deliveries need to be scheduled in order to fulfill an order. The deliveries cannot overlap and the time between consecutive deliveries has to be small. Some vehicles can only be used for the delivery of concrete. Other vehicles, with specialized unloading equipment, may have to be present at a construction site to assist with the unloading operations of other vehicles. Such vehicles need to arrive first at a construction site and remain at the construction site until the complete order has been fulfilled.

We present a solution approach for the problem at hand which combines the power of integer programming solvers with the search schemes of meta-heuristics. Initial computational experiments have demonstrated that the approach is capable of producing high quality solutions in a reasonable amount of time (solutions that improve those produced by a simulated annealing heuristic currently employed by the company).

The meta-heuristic searches for compatible order fulfillment patterns, which are used to define an integer multi-commodity flow problem with side constraints to find optimal truck itineraries. Instances of the integer multi-commodity flow

problem with side constraints are rather easy to solve if the number of patterns is relatively small.

2 Problem Setting

Construction sites require concrete for their daily activities. Concrete consists of cement, mineral aggregates, such as gravel and sand, and water and is widely used in the construction industry. In order to satisfy the demand for concrete, full truckloads of concrete will be delivered at constructions sites. The concrete is produced at plants. A plant's loading rate determines the time it takes to fill up a truck. The fleet of delivery trucks is heterogeneous. Trucks do not only differ in terms of capacity, but also in terms of their equipment. Orders for a particular day are known in advance. An order not only specifies the required quantity of concrete, but also the time window within which the first delivery has to start, and special loading equipment, if any, that is required. Furthermore, an expected unloading rate at the construction site is provided. The objective is to minimize total cost, consisting of total travel cost, (small) penalties for delays between two consecutive unloading operations for an order, and (high) penalties for unfulfilled orders.

3 An Integer Multi-commodity Flow Formulation

The problem is modelled as an integer multi-commodity flow problem on a time-space network (with some similarities to the model proposed by Hoffman and Durbin [1]). Each type of delivery truck is modelled as a separate commodity. However, instead of considering all possible loading and unloading operations and all possible truck movements, only a limited number of options is considered. This done through the use of order fulfillment patterns. A fulfillment pattern for an order completely specifies which delivery trucks visit the construction site, in what order, and when unloading operations take place. Given a (small) set of fulfillment patterns for each order, the integer multi-commodity flow formulation selects a fulfillment pattern for each order and determines an itinerary for each of the delivery trucks. Note that a truck typically makes several full truckload deliveries during a day and can load concrete at any of the plants.

The time-space network has a node for every plant, time instant combination and every site, time instance combination. The integer multi-commodity flow formulation contains flow-balance constraints at every node of the time-space network for every truck. Starting and ending condition ensure that trucks depart from and return to their home plants. Side constraint capture the fact that exactly one fulfillment pattern has to be chosen for every order, otherwise penalty will accrue.

In most case, trucks leave a construction site immediately after finishing their unloading operation. However, if an order requires specialized unloading

equipment, the first truck to arrive needs to have the required equipment and remain at the site until the entire order has been fulfilled. That truck departs as soon as the last unloading operation of a fulfillment pattern has been completed.

4 Fulfillment Pattern Generation

Order fulfillment patterns specify different ways to satisfy the demand for concrete at a particular construction site. One fulfillment pattern has to be chosen for each order. A fulfillment pattern completely specifies which delivery trucks visit the construction site, in what order, and when unloading operations take place. Fulfillment patterns are generated iteratively inspired by concepts from meta-heuristics (see Glover and Kochenberger [2]).

First a number of base patterns is generated for every order. The integer multi-commodity flow formulation is solved with only these base patterns. Next, based on an analysis of the current solution, new patterns are generated and added to the pool of patterns. The integer multi-commodity flow formulation is solved with the patterns in the pattern pool and the process repeats.

Two types of patterns are generated: (1) completely new patterns, and (2) modified existing patterns. The generation of these patterns is guided by the characteristics of the orders as well as by the patterns selected in the last solution to the integer multi-commodity flow formulation. Care is taken to generate new patterns that are complementary to the patterns in the last solution (will have little or no overlap) and new patterns with start times for unloading operations that differ in peak delivery periods.

4.1 Generation of the base fulfillment patterns

As a first step, one pattern is generated randomly for every order. The start of the first unloading operation of the pattern is selected randomly within the time window associated with the order. Next, the truck performing the first unloading operation is selected randomly applying a roulette wheel method. Of course only trucks that can feasibly perform the unloading operation will be considered, e.g., for orders with special requirements concerning instrumentation only trucks with appropriate instrumentation will be taken into account. Next, additional unloading operations, and thus trucks, will be added to the pattern until the cumulative capacity of scheduled trucks exceeds the quantity of concrete ordered. We properly handle trucks that have already been used in the emerging pattern, i.e., we ensure that there is sufficient time to drive to the nearest plant, load concrete, and return to the construction site.

An additional $n - 1$ fulfillment patterns will be generated for each order, where we enforce that patterns do not overlap with existing patterns for other orders. The probability for selecting a truck for an unloading operation is inversely proportional to the number of patterns it would overlap with.

4.2 Generation of additional fulfillment patterns

Each time the integer multi-commodity flow formulation has been solved, a set of new patterns is generated and added to the pool of patterns.

As mentioned earlier, new patterns are generated in one of two ways. Either delays are inserted between consecutive deliveries of an existing pattern or a completely new pattern is generated.

To generate a modified pattern, a single delivery in the pattern is selected randomly and its start time is delayed (which forces all subsequent deliveries to be delayed as well). The selection probability for a delivery depends on the workload during the time interval of the delivery. The more concrete needs to be delivered during the time interval, the more likely it is that the start of the delivery will be delayed.

When generating a new pattern, we try not to generate a pattern that overlaps with the patterns of the current solution of the integer multi-commodity flow formulation. Again, all trucks scheduled to make a delivery in a pattern will be chosen sequentially. The selection probability is inversely proportional to the number of patterns it would overlap with. Any special requirements concerning instrumentation will be considered, such that only eligible trucks will be scheduled in the first position of a pattern.

5 Preliminary Results

So far, two representative instances for a medium-sized company in Austria have been solved. The first instance is relatively small with two plants, three construction sites, seven trucks, and five small to medium-size orders (quantities vary between 16 and 45 cubic meters). The second instance is larger and has 45 trucks, nine plants, and five large orders (quantities vary between 23 to 314 cubic meters). The capacity of trucks varies between 5 and 11 cubic meters.

First, we solved these instances by blindly enumerating a large number of patterns with and without delays. Using this pool of patterns the integer multi-commodity flow formulation was solved.

Next, we applied the iterative approach described above. Five base patterns were generated for each order. As long as the solution to the integer multi-commodity flow formulation did not fulfill all the orders, one existing pattern per order was modified by inserting delays, and three new patterns were generated per order.

Table 1: Instance 1

x	avg sol	avg time	avg # iterations	# patterns	# patterns BF
5	55.08	190.20	16	69	~ 200
10	54.00	377.08	15.4	71.6	~ 200
20	52.17	1762.31	22.4	109.6	~ 200

Table 2: Instance 2

x	avg sol	avg time	avg # iterations	# patterns	# patterns BF
5	712.56	79.13	1.8	12.2	~ 200
10	708.40	57.04	1.6	16.4	~ 200
20	655.84	31.49	0	20	~ 600

Each instance was solved five times. Table 1 and Table 2 show the average solution quality, average run times (in seconds), and the number of iterations when starting with x base patterns. The last two columns give the average number of patterns generated to obtain the solution and the number of randomly generated patterns that would have been required by the brute-force method to obtain a comparable solution.

All experiments were performed on Pentium 4, 3.2 GHz machine with 3 GB RAM.

References

[1] K. Hoffman and M. Durbin. "The Dance of the Thirty Ton Trucks," Operations Research. To appear.

[2] F. Glover and G.A. Kochenberger. "Handbook of Metaheuristics," Kluwer Academic Publishers, Norwell, 2003.