Ant-colony optimization of event orders at shunting yards

Final project for the course Heuristic search methods

Sara Gestrelius

January 25, 2016

Abstract

In order to serve many origin-destination pairs freight transportation companies often operate a hub-and-spoke network. The hubs are shunting yards, where arriving trains are decoupled and their cars sorted into new departing trains. Shunting yards often consists of three sub-yards: an arrival yard, a classification bowl (where the cars are sorted) and a departure yard. Finding the most efficient sorting procedure in terms of effort is an NP-hard problem, even if only the classification bowl is considered. Optimization models for this classification bowl problem have been devised, but they require the arrival and departure yard schedules, including the times for when trains are rolled in and out of the classification bowl, to be pre-defined. In this project an ant colony heuristic for finding good arrival and departure yard schedules is constructed, and different pheromone update strategies are tested.

1 Introduction

Operating a hub-and-spoke network is often cost effective when serving many origin-destination pairs, and is used by e.g. airline companies and large freight companies, including some train freight companies.

For freight trains the hubs are shunting yards where cars from incoming trains are sorted into new departing trains. Shunting yards often consist of three sub-yards: the arrival yard, the classification bowl and the departure yard (see Figure 1). The incoming trains arrive to the arrival yard where they are decoupled and inspected before they are rolled into the classification yard where they are to be sorted into new outbound trains. The classification yard has two types of tracks: *train formation tracks* and *mixing tracks*. Train formation tracks are where the outbound trains are built. An outbound train is built on one and only one formation track, and while a train is being built on a formation track no cars belonging to other outbound trains may be rolled to that formation track. On the other hand, mixing tracks can host a mixture of cars, and are used to store cars whose departing train has yet to be allocated a train formation track. The cars on the mixing track are pulled back to the arrival yard at given points in time. They are then once again rolled into the classification yard, allowing for the cars to be rolled to their allocated train formation track. Mixing cars results in extra work and wears on the yard, and

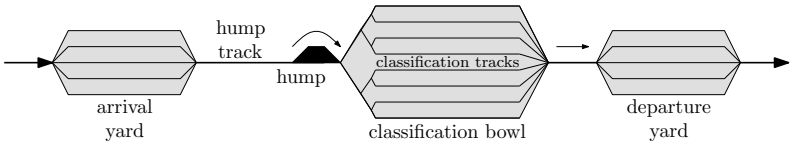

Figure 1: Lay-out of a shunting yard.

should be avoided if possible. The number of mixed cars can therefore be used as a measure of schedule inefficiency (i.e. the number of mixed cars should be minimized to get an efficient schedule). Once an outbound train has been completely built on the classification yard it can be rolled to the departure yard and wait there for its departure time. All sub-yards have a certain capacity, and all operations (inspections, rolling, decoupling and coupling) take a certain amount of time (see Section 1.1). A feasible shunting schedule must respect all capacity and time constraints.

The sorting in the classification bowl is the core of the shunting operation. Optimization models for this sorting problem have previously been developed [1]. However, the sorting problem on the classification bowl depends on the roll-in and roll-out order of the trains, and different roll-in and roll-out orders will result in different optimal values of the classification bowl problem. The aim of this report is to investigate the feasibility of using an ant-colony algorithm to generate roll-in and roll-out schedules that define a classification bowl problem that has a good optimal solution. The ultimate goal is to use the ant colony algorithm to generate arrival and departure yard schedules that are feasible with respect to capacity and timing constraints, and then use the optimization model to find the cost of the shunting given the arrival and departure yard schedules. However, in this report a simplified way of measuring the effectiveness of the arrival and departure yard schedules is used rather than the optimization model itself. This heuristic for determining the effectiveness of the arrival and departure yard schedules could be used in combination with the optimization model to limit the running time of the overall problem solving also in the final version.

1.1 Problem definition

Assume that the arrival yard has a tracks, the classification bowl c tacks and the departure yard d tracks. Assume that all tracks are long enough to host all trains.

Further, there are certain time constraints that must be fulfilled. First of all, the time needed for inspection and decoupling at the arrival yard means that the roll-in time of a train i, t_r^i must be at least a time interval δ_a later than the arrival time, t_A^i. Further, there must be at least δ_c between the last roll-in to a departing train d and its roll-out time, t_r^d (to allow time for rolling, coupling and inspection), and at least δ_d between the roll-out and the departure time (t_D^d) (time needed for car rolling and departure inspection). There must also be δ_r time between two roll-ins, and δ_o

2 Ant-colony algorithm

In an ant-colony algorithm *ants* generate solutions by, to some extent, following *pheromone trails* or *weights* from previous ants. The cost of a solution is used to decide if other ants should be encouraged to generate solutions that are similar to the current one or not. This encouragement is accomplished by updating the pheromone trails. In this section we first discuss the encoding of the problem, and then the solution construction method used by the ants. Finally the pheromone update process is introduced.

2.1 Encoding

The main feature defining the problem is the order of arrival, roll-in, roll-out and departure events. This permutation of events is an indirect encoding as timings of the events is also needed for a complete schedule. A complete schedule could be extracted from this indirect encoding by e.g. scheduling all events as early as possible while still respecting all timing constraints, and this is the method used in this report. The main decision will be which event to choose next, but at the same time an exact encoding based on the earliest possible time will be used to check the schedule feasibility.

2.2 Solution Construction

The ants construct the schedule in time order, i.e. the next event is always sampled from the current event's list of potential succeeding events, called *potential successors*. The first arrival is the first event in every schedule, but then an ant may chose any of this first arrival's potential successors as the next event (called *the chosen successor*). The next event then becomes the current event and an successor is chosen from the current events potential successors. Note the distinction between an event's potential successors (all events that can be chosen next) and the events chosen successor (the next event chosen by the ant). The chosen successors is sampled among the potential successors. The sample probability is based on the pheromone weights of the potential successors. In fact, the probability of sampling a successor e_i is

$$p(e_i) = \frac{w(e_i)}{\sum_{e \in P} w(e)}$$

where P is the set of potential successors.

The main problem when constructing the solution is to ensure that the schedule is feasible, i.e. that it respects the timing and capacity constraints from section 1.1. This is accomplished by restricting the set of potential successors depending on previously selected events, and also by back-tracking and making certain orderings tabu if an infeasible schedule has been constructed. Despite these feasibility maintaining tactics, the ant may end up in a situation where the solution is infeasible (most notably, it may end up having no potential successors). In this case the ant will start all over again (and we will record the failure).

2.2.1 Events

There are two types of events, *set events* that have exact timings (arrivals and departures) and *variable events* that can happened during a time interval (roll-ins and roll-outs). The time-intervals for variable events are calculated during the problem set-up and are based on the set events' exact times and the timing constraints from section 1.1. However, the intervals are not updated as the ant progresses. This is one of the reasons we end up in infeasible situations (see section 2.2.2).

Every event has a set of potential successors, but every event also has a set of forced predecessors. The set of forced predecessors include events that have to take place before the event in question is chosen. The sets are constructed as follows:

Arrival
Predecessors (forced): the set event before the arrival and any variable event whose latest time is before the time of this arrival but after the exact time of the previous set event.
Successors (potential): any variable event whose interval covers some part of the interval between the arrival and the next set event and also the next set event.
Roll-in
Predecessors (forced): the latest set event that has a time that is earlier than the earliest roll-in time (call it the *early limit event*), and any variable event whose latest time is earlier than this events earliest time, but later than the time of the early limit event.
Successors (potential): the first set event after the late roll-in time (call it the *late limit event*), and any set event that falls within the roll-in interval. Also, any variable event whose time interval overlaps with the interval between the early roll-in time and the time of the late limit event.
Roll-out
Predecessors (forced): all roll-in events that are associated with the roll-out event. The latest set event that has a time that is earlier than the earliest roll-out time (the *early limit event*). Finally, any variable event whose latest time is earlier than this events earliest time but later than the time of the early limit event.
Successors (potential): the first set event after the late roll-out time (call it the *late limit event*), and any set event that falls within the roll-out interval. Also, any variable event whose time interval overlaps with the interval between the early roll-out time and the time of the late limit event.
Departure
Predecessors (forced): the set event before the departure and any variable event whose latest time is before the time of this departure but after the exact time of the previous set event.
Successors (potential): any variable event whose interval covers some part of the interval between the departure and the next set event and also the next set event.

Events will remain invalid choices until all forced predecessors have been selected even if the current time is within a variable events time interval. Further, whenever there is only time for forced predecessors to be chosen (e.g. because

there is only a short period of time between two arrivals and many roll-ins have to happen before the next arrival) the list of potential successors will be restricted to the forced events. That is, the list of potential successors is updated as the ant progresses to support schedule feasibility.

The forced predecessors set will also be used for tabu-events to prevent overloading the arrival and departure yard, and for when the schedule constructed becomes infeasible with respect to timing constraints. This is explained in more detail in the next section.

2.2.2 Feasibility

Arrival and departure yard capacity
There can only be a certain number of trains on the arrival and departure yard at any point in time. That is, the number of planned arrivals minus the number of planned roll-ins must always be smaller or equal to the capacity of the arrival yard, a. Likewise, the number of planned roll-outs minus the number of planned departures must always be smaller or equal to the capacity of the departure yard, d. If the yard capacity is reached arrivals (roll-outs) are invalid choices until a roll-in (departure) has been chosen. If there are only arrivals (roll-outs) in the set of potential successors the ant has no valid choice, and will back-track until it finds a roll-in (departure) that can happen after the current event. The current event will then be added to the roll-in's (departure's) list of forced predecessors as a tabu event. That is, the ant will now chose the current event before the arrival (roll-out) and may thereby avoid the current overloaded yard situation.

Infeasible exact timings
Even though the selection order of events is restricted it is possible to pick event orders that are infeasible with respect to the timing constraints. This is an effect of the variable events time-intervals and the fact that a certain amount of time is needed between roll-ins and roll-outs. For example, three roll-outs a, b and c with intervals ending late may be chosen before a roll-out e with an interval ending early, and thereby making it impossible for e to be scheduled within its interval when exact timings are considered. To ensure feasibility the earliest possible time is allocated to each chosen event, and if this time is later than the latest allowed time of the event the ant will back-track until it finds an event that can be chosen as tabu, i.e. has no current connection to the failed event and that can be scheduled at the current (later) time, and then add the failed event as a forced predecessor to the event found when brack-tracking. The ant then restarts from the event before the event found when back-tracking. In our example above, e could be added as a forced predecessor to c ensuring that only a and b are now scheduled before e.

Non-recovered infeasbility
Despite the two recovery actions above, the ant may still end up in a situation where there's no available next event, or where no feasible event is found for tabu. For example, a late roll-in order may force a roll-out to be scheduled late. However, a late roll-out scheduling may not be possible due to capacity and time restrictions. The tabu search will try to find an event to schedule after the roll-out event, but the solution to the problem is really to schedule the roll-in

earlier, which is not a recovery strategy that has been coded. If an ant ends up in a situation where the feasibility can't be recovered using the strategies above, it will start all over again. We record the number of times an ant fails to generate a solution.

2.3 Pheromone updating

The core of ant colony optimization is the pheromone trail the ants produce, i.e. how the success of previous ants should affect a current ant's solution construction process. There are a number of different strategies. The pheromone trail could be updated after each ant has finished finding a solution, or after N ants have found solutions. The effect a solution should have on future ants could depend on the solution cost, or on how good a solution is in comparison with other solutions found.

In this report the pheromone weight of all orderings is set to 1.0 at the start of the algorithm. Each iteration then consists of N ants, and the pheromone weight is updated after each iteration. A the end of an iteration the N solutions are sorted in increasing cost order, and the weights of the event orderings in the n best solutions are proportionally increased by $1/i$, where i is the order of the solution. That is, the weight of the orderings in the best of the N solutions are increased by 1.0, and the weight of the orderings in the i^{th} best solution by $1/i$. We call the percentage of solutions used in the updating process, i.e. $\frac{n}{N}$, the *update percentage.*

The algorithm stops when there has been m iterations without any improvement of the best solution found.

2.3.1 Solution cost

The aim of the algorithm is to produce arrival and departure yard schedules that define a classification bowl problem with a low optimal solution value. One tactics would therefore be to find the optimal classification bowl schedule for each solution. However, in this report a simplified cost evaluation is used. The outbound trains are sorted according to their reversed roll-out times, and are then allocated to classification bowl tracks in this order. When scheduling an outbound train a before an outbound train b, some of b's cars may need to be mixed as they are rolled into the classification bowl before the roll-out time of a. The track that holds a train b that required the least mixing cars is therefore chosen for train a. In fact, we used the number of mixed cars multiplied by the time they need to be mixed as the cost. The exact timings of pull-backs is not considered.

3 Results

A schedule for one week was found, using real data for arrivals and departures. An iteration size of 50 ants were used, and three different update percentages, 100%, 50% and 0%. 0% were chosen to test if the heuristic outperforms random search. When no improvement in the cost had been found for 100 iterations the algorithm was stopped.

The results are shown in graph 2. An updating percentage of 100% gives the best results, and also converges the fastest. However, there are quite many

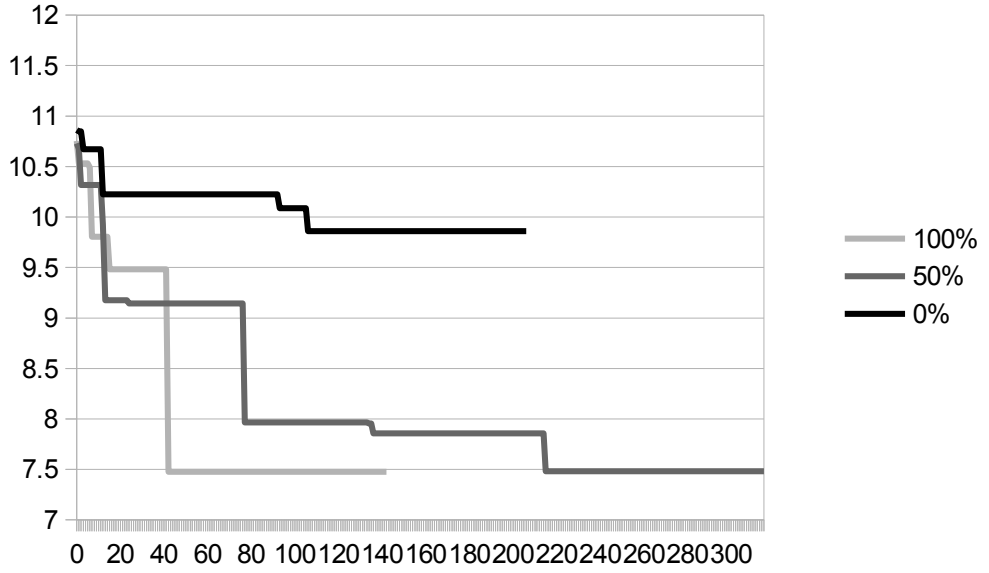

Figure 2: The improvement for the best solution value for different updating percentages.

iterations without an improvements for the 0% and 50% update percentages also before the 100 iterations that shuts down the algorithm. This indicates that stability for 100 iterations might not be an appropriate stopping criteria. Further, the update percentage of 0% clearly yields worse results than the other two, indicating that the pheromone updating is indeed effective.

Figure 3 shows the number of times an ant failed to generate a feasible solutions in each iteration for the different update percentages. As expected the iteration number does not seem to affect the number of failures for 0%, while failures become more common for higher iteration numbers for 100% and 50%. This might be because the good solutions are close to the capacity limit, and therefore ants following this pheromone trail more often suffer from capacity shortage. This indicates that the problem with failures may aggravate if the algorithm is run for more and more iterations to find better and better solutions.

The total running time for the algorithm for the different update percentages were 1h56min for 100%, 3h40min for 50% and 2h18min for 0%.

4 Conclusion

An ant colony optimization algorithm for the roll-in and roll-out schedule for a shunting yard has been developed. Ants generate solutions that are evaluated using a simplified cost function. Feasibility is supported but not guaranteed by using forced predecessors and including tabu event orders as needed. In case the ants end up in an infeasible situation despite these tactics the solution construction process is restarted for that ant.

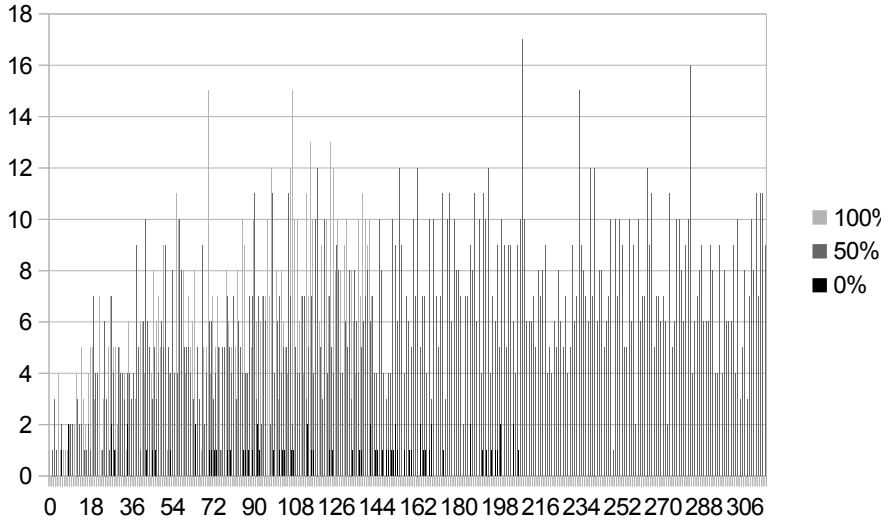

Figure 3: The number of failures in each iteration for different updating percentages.

The results show that the ant-colony algorithm with pheromone updating outperforms the same algorithm but without pheromones.

The algorithm would benefit from a number of improvements such as updating the time intervals of variable events as the ant progresses, and also updating the set of forced predecessors and potential successors. This would decrease the number of times ants fail to find a solution. It would also allow for more advanced feasibility tactics.

When it comes to solution quality testing more pheromone updating strategies, and also more iterations would be interesting. Improving the cost evaluation, which is currently quite crude, would also be beneficial. Finally, track lengths and pull-backs need to be included for a real-world schedule. If these are included the heuristic cost evaluation without optimization becomes less proper, and some sort of hybrid heuristic should probably be considered.

References

[1] Markus Bohlin, Sara Gestrelius, Florian Dahms, Matús Mihalák, and Holger Flier. Optimization methods for multistage freight train formation. *Transportation Science*, Printed online, 2015.