Strategic Gang Scheduling in the Railway Industry

Dengfeng Yang
Department of Industrial Engineering and Management Sciences
Northwestern University

Clark Cheng, Edward Lin, Kannan Viswanath, Jian Liu
Norfolk Southern Corporation

Diego Klabjan
Department of Industrial Engineering and Management Sciences
Northwestern University, Evanston, IL US
Email: d-klabjan@northwestern.edu

1 Introduction

The railway industry is an infrastructure intensive industry. Major Class-1 US railways have more than 20,000 miles of tracks. The tracks are heavily used and thus subject to wear and failures. Throughout a year, track maintenance, which encompasses both preventive and real-time maintenance, is an equipment and labor intensive process. Maintenance workers work in groups, called gangs, performing heavy labor duties such as installing track ties, driving track spikes, shoveling ballast, and other similar maintenance track related activities. Gangs work outdoors at almost all weather conditions to install and maintain the tracks properly. A gang consists of several members, typically between ten and one hundred, with different responsibilities such as machine operators, termite welders, assistant extra gang foremen, or extra gang foremen. A gang beat or a beat section is a regular section of a track where some form of a maintenance need to be performed. During a year a gang moves from one beat section to another upon completion of the work at the section. A beat section together with its attributes such as the type of work, the number of days to complete the work, and an underlying time window to perform the task is called a job.

Gang scheduling is to find a favorable gang schedule for each gang during a given planning horizon that is governed by several regulatory and union rules. A gang schedule consists of a sequence of jobs together with the underlying start time of each job. Typically there are three or four types of gangs (rail, tie and surface, surface, and dual) and more than 1,000 jobs around the U.S. Each job has an underlying required gang type, which implies that a particular job can only be carried out by particular gangs. Certain jobs must obey precedence constraints, e.g., work on ties must precede any

work on the rails. From the operational perspective, it is not desirable to perform two jobs geographically close to each other during the same period since it could substantially disrupt train operations. Few occurrences of such situations create a more robust schedule. At most railways the process of gang scheduling is an intensive manual effort often leading to costly and inefficient schedules.

The main goal of this study is to develop an algorithm to solve the gang scheduling problem considering all business requirements. The objective is to develop an algorithm capable of:

1. minimizing the total cost including the gang related cost of transitioning between jobs and the cost of equipment movement between two locations, as well as the travel allowance covering the transfer from a job location and the home domicile of the underlying gang,
2. obeying all business requirements, e.g., it is suitable for gangs to work at a southern area during the winter period, but jobs in the northern part must be performed during the summer season, and
3. computationally handling the large-scale instances arising in the industry.

The main contributions of this work are:

1. designing a network based model capturing job precedence and robustness, and
2. developing a construction and mathematical programming heuristic (math-heuristic) for solving the underlying model.

2 Model

We formulate a network model *G (N,A)* for each gang *g*, where *N* is the set of all job nodes and *A* the set of arcs in the model.

Node (*j*,*t*) of the network is encoded by job *j* eligible to be performed by gang *g* and the associated starting time *t*. Given time *t*, job *j*, and gang *g*, it is possible to calculate the completion time of the job (from the duration requirement of job *j* and productivity of gang *g*). Two nodes are connected by an arc if the completion time of the job associated with the tail plus the transition time from one job to the other one is less than or equal to the start time of the node associated with the head. Other requirements such as gangs not working during weekends and holidays can easily be directly incorporated in the network.

We next focus on the underlying node and arc cost. Cost is driven by travel allowance, and thus all costs are measured in monetary units, i.e., the money paid to a gang for traveling between the locations of two consecutive jobs or for weekend stays at the home domicile. The arc cost is composed of three components:

1. the travel cost between the location of job *j*, the home domicile of the gang if the transition time includes a weekend, and then return to the location of the adjacent job *k*,
2. the direct travel cost between the two locations if the weekend is not included, and

3. the equipment movement cost from the location of job *j* to the location of the adjacent job *k*.

In addition, each node bears the travel allowance cost for weekend travel to the home domicile over weekends. We note that job duration can span several weeks and thus it might require several weekend home trips.

By construction, each path in the network forms a gang schedule. The underlying mathematical program includes variables that assign a path (gang schedule) to each gang. The underlying constraints impose:

1. each job is assigned to exactly one selected path,
2. each gang is assigned one and only one path,
3. precedence constraints linking paths among the gangs,
4. equal number of work days for all gangs of a certain type,
5. robustness constraints likewise link various path.

3 Methodology

The problem as posed is very hard to solve to optimality since it is an NP-hard problem, and a pure mathematical programming approach does not work due to the sheer size of real-world instances. Either significant ad-hoc preprocessing is required or a branch-and-price algorithm developed. Computational tractability of the latter is questionable due to several unstructured constraints linking paths. For this reason, we resort to a heuristic. Instead of employing a traditional local search strategy, we combine very large-scale neighborhood search ideas with mathematical programming. To this end, we use a two-phase solution methodology. In the first phase a solution covering many jobs but not necessarily all of them is obtained. In the subsequent phase the uncovered jobs are inserted by means of a mathematical program. The solution quality of a feasible solution is then iteratively improved by removing jobs from the incumbent solution and then reinserting them back by means of the same mathematical program.

3.1 Initial Construction Heuristic

First, we designed a heuristic that generates several shortest paths using dynamic programming. Gangs are ranked and then a schedule is found for each gang sequentially by solving a shortest path type problem. Once a path is fixed for a gang, the network is accordingly modified so that each job is assigned at most once, and the precedence and robustness rules are warranted.

To obtain balanced workload within the same gang type we do not choose the exact shortest path with the maximum number of jobs and smallest cumulated cost at the last job node. Instead we choose a path with the total duration falling in some predetermined range. Since each gang has a limited number of working days for the year and each available job has a restricted time window, in this phase, the solution by the shortest path algorithm cannot schedule all possible jobs. An improvement phase is introduced to cover the unassigned jobs.

3.2 Improvement Strategy

The initial solution is then iteratively improved at each subsequent iteration by an interchange algorithm so as to schedule all of the jobs and to guarantee finding a no worse solution than the one we begin with. The basic idea of the improvement phase is to randomly select certain nodes to extract from the starting solution, then extract the nodes, and derive sequences of nodes including extracted nodes and uncovered nodes to reinsert into the short-cut solution. Next we formulate an integer program and solve it using an ILP solver to incorporate the results back into the solution. This process is repeated until an iteration limit is reached or the solution is of an acceptable quality.

3.2.1 Recombination

In this step, we use all the extracted/uncovered nodes to create a pool of subsequences to be potentially reinsterted into the solution. To make the problem simpler and improve the performance, we only consider the short sequences of nodes (1 or 2 jobs). The business rules, such as job precedence and robustness, are also considered in this step.

3.2.2 Reallocation

In this step, we form an insertion ILP. This ILP reinserts back the extracted/uncovered nodes. Note that the model is always feasible since the nodes can always be reinserted back to the original locations. Binary decision variables indicating whether certain sequence *s* is inserted at some insertion point *i*, and integer decision variables presenting the shift in time of the job sequences at the short-cut solution after nodes extraction, are introduced into the ILP model. It is critical to introduce the shift time variables since at some insertion point, the time window of the subsequent job sequence must be enlarged or shrinked to accommodate the to-be-inserted job sequence. The objective is to minimize the overall cost. Essentially, the ILP is an assignment problem, but with additional constraints, such as jobs precedence and robustness, and time window restrictions. The complete ILP has a large number of columns. We solve it by randomly choosing a predetermined, large number of columns from the entire set of columns to add to the problem.

3.2.3 Reinsertion

We incorporate the optimal results obtained from the reallocation ILP into the solution using specially designed insertion operations. This reinsertion step marks the end of an iteration, and the entire series of steps is repeated until all jobs are scheduled or an improved solution is obtained.