ABSTRACT

Manual short-term scheduling in underground mines is a time-consuming and error-prone activity. We use Constraint Programming to automate the scheduling process: deciding what to do where and when. We extend previous work by including fleet travel times, and by introducing a new model based on solving a related scheduling problem and transforming its solution back to the original domain. In addition, a neighborhood definition is introduced to optimize using Large Neighborhood Search. Results show that the proposed method scales to realistic problem sizes, and that the solutions obtained are of high-quality.

KEY RESULTS

• The mine scheduling problem resembles a rich variant of a \( k \)-stage flow shop, with a mix of interruptible and uninterruptible jobs, periodically induced machine unavailabilities, after-lags in some stages, sharing of (certain) machines between stages, and sequence-dependent setup times due to the travel times of the mobile machines [1].

• Underground mines can have road networks spanning several hundreds of kilometers. Therefore, to ensure that schedules are feasible to operationalize, we extend previous work [2] by including travel times of the mobile machines in the constraint model.

• In addition, we propose a new approach based on first generating solutions to a modified uninterruptible scheduling problem without blast windows. A post-processing step inserts blast windows and transforms the solutions to solve the original problem. To further improve the obtained schedules, Large Neighborhood Search is used with a domain-specific neighborhood definition based on relaxing all variables corresponding to jobs scheduled at a random subset of production areas.

• We can find high-quality schedules to realistic instances, generated using data from an operational mine, including more than 200 jobs. Compared with a common constructive heuristic [3], solutions are found within minutes exhibiting \( \sim 7\% \) lower objective value. Studying the optimal solution to a relaxed problem, we note that on a realistic instance we are at most \( \sim 12\% \) away from optimality.

REFERENCES:

