Available online at www.sciencedirect.com
ScienceDirect
European Journal of Operational Research 176 (2007) 1205–1218
EUROPEAN JOURNAL OF OPERATIONAL RESEARCH
www.elsevier.com/locate/ejor

O.R. Applications

Using aggregation to optimize long-term production planning at an underground mine

Alexandra M. Newman [a,*], Mark Kuchta [b]

[a] Division of Economics and Business, Colorado School of Mines, 1500 Illinois Street, Golden, CO 80401, USA
[b] Department of Mining Engineering, Colorado School of Mines, 1500 Illinois Street, Golden, CO 80401, USA

Received 21 October 2004; accepted 7 September 2005
Available online 18 November 2005

Abstract

Motivated by an underground mining operation at Kiruna, Sweden, we formulate a mixed integer program to schedule iron ore production over multiple time periods. Our optimization model determines an operationally feasible ore extraction sequence that minimizes deviations from planned production quantities. The number of binary decision variables in our model is large enough that directly solving the full, detailed problem for a three year time horizon requires hours, or even days. We therefore design a heuristic based on solving a smaller, more tractable, model in which we aggregate time periods, and then solving the original model using information gained from the aggregated model. We compute a bound on the worst case performance of this heuristic and demonstrate empirically that this procedure produces good quality solutions while substantially reducing computation time for problem instances from the Kiruna mine.

Keywords: Production scheduling applications; Mining and metals industries; Heuristic integer programming algorithms

1. Introduction and background

We develop a mixed integer programming model to determine a multi-time period extraction sequence of ore blocks in an underground mine. The goal is to minimize the deviation between actual and planned production quantities subject to operational constraints. Because the number of binary decision variables

* Corresponding author. Tel.: +1 273 3688; fax: +1 273 3416.
E-mail addresses: newman@mines.edu (A.M. Newman), mkuchta@mines.edu (M. Kuchta).

0377-2217/$ - see front matter
doi:10.1016/j.ejor.2005.09.008

representing whether or not to extract a given piece (or *machine placement*) of ore during a time period may be on the order of thousands, the size of the model precludes us from quickly determining underground mine schedules by solving the full, detailed problem. In this paper, we outline a procedure to increase the tractability of the scheduling model at the Kiruna mine. We solve smaller, aggregated models and use information gained from these models to restrict the size of the original model. We show that our method yields solutions with appropriate fidelity and of acceptable quality in a reasonable amount of time. Because our aggregation procedure is a heuristic, we compute a bound on its worst case performance.

Although the importance of using integer programming models to determine discrete production scheduling decisions in the mining industry has been identified (e.g., Winkler, 1996), many authors have been unsuccessful at modeling and solving realistic mining scenarios, e.g., Trout (1995) and Smith (1998). Earlier attempts at the Kiruna mine also failed to yield a production schedule of requisite duration in a reasonable amount of time, e.g., Almgren (1994), Topal (1998), and Dagdelen et al. (2001). Rather than determining an optimal schedule, these authors resort to shortening the schedule timeframe and/or to sacrificing schedule quality. Carlyle and Eaves (2001) formulate an integer programming model to maximize revenue from mining platinum and palladium at Stillwater Mining Company, and are able to obtain near-optimal solutions without using any special techniques to reduce solution time. Smith et al. (2003) construct a production scheduling model for a copper and zinc underground mine at Mount Isa, Australia. The decision variables in their model represent the time at which to mine each extractable entity (production block) to maximize net present value subject to operational constraints, e.g., ore availability, concentrator (mill) capacity, mine infrastructure production capacity, grade (mineral quality) limits, continuous production rules, and precedence relationships between production blocks. However, the authors are unable to solve all instances of their problem in a reasonable amount of time. Sarin and West-Hansen (2005) maximize net present value for an underground coal mine subject primarily to precedence, quality, and production smoothing constraints. They tailor a Benders decomposition technique to solve randomly-generated problem instances, and present a case study. All three of the afore-mentioned models apply to a mine that uses different underground mining methods than Kiruna. In Kuchta et al. (2004), we explain how the model contained in this paper was implemented at the Kiruna mine. Newman et al. (2005) present a different type of production scheduling model for the same mine, and a technique other than the one presented here to reduce solution time.

The paper is organized as follows: In Section 2, we describe the Kiruna mine and our integer programming production scheduling model. In Section 3, we present the aggregation procedure, along with related literature, and in the following section show how to compute a bound on the aggregation procedure's worst case performance. In Section 5, we provide numerical results. Section 6 concludes the paper.

2. The Kiruna mine and the production scheduling model

Our research is motivated by operations at Sweden's Loussavaara-Kiirunavaara Aktiebolag (LKAB) Kiruna mine, currently the second largest underground mine in the world. The mine contains a world-class high-grade magnetite deposit, approximately 4 km long and 80 m wide on average, and produces approximately 24 million tons of iron ore per year. To extract the ore, LKAB uses an underground mining method known as sublevel caving, and divides the orebody vertically into ten main *production areas*, about 400–500 m in length, each with its own group of *ore passes*; this group is known as a *shaft group*. The mine is also divided horizontally into *sublevels*, separated by about 29 m. Electric load haul dump units (LHDs) with a capacity of 25 tons transport the ore on a sublevel within each production area from the mined site to the ore passes. Trains operating on the transport (or main) level haul the ore from the ore passes to a crusher. At the crusher, the ore is broken into pieces small enough to be hoisted to the surface via vertical

shafts. The site on which each LHD operates, usually 200–500 m in length and containing from one to three million tons of ore, is referred to as a *machine placement* and consists of multiple, perhaps 10–20, *production blocks*. Once started, mining restrictions require continuous production of a machine placement until all available ore in each production block within the machine placement has been removed. This precludes the miners from needing to reblast the ore, and from suffering the aggravation of tracking partially-mined machine placements. Because we assume a specific order in which all production blocks within a machine placement must be mined, we make decisions in our model only at the *machine placement*, and not at the *production block*, level. Fig. 1 depicts the relationship between the production areas, ore passes, shaft groups, sublevels, and vertical shafts, and shows a load haul dump unit, a train, and the crusher.

The mine produces two raw ore types, *B* and *D* ore, having a phosphorous content less than 0.10%, and greater than 0.10%, respectively. A machine placement can contain both raw ore types and waste rock. Each production block contains a quantity of material extractable in a month. Within each production block, the relative quantities of each ore type and waste can be estimated from an in situ geologic model, the mine layout, and the true production data obtained after mining upper levels. Monthly production capability can be estimated from the haul distance and ore extraction rates of the LHDs. From the raw ore types extracted from each production block, mills produce fines and/or pellets, both of which are used in the steel-making industry. These mills must be run continuously and at capacity to meet contractual agreements; mine planners set their demand for the raw ore types accordingly. The mine holds virtually no stockpiles of ore due to lack of physical space, mine planners' aversion to rehandling ore, and the long-term nature of the contractual agreements. In order to adequately satisfy demand at the mills, Kiruna must carefully plan the extraction of each type of ore, which the mine does under the assumption of location- and time-independent extraction costs.

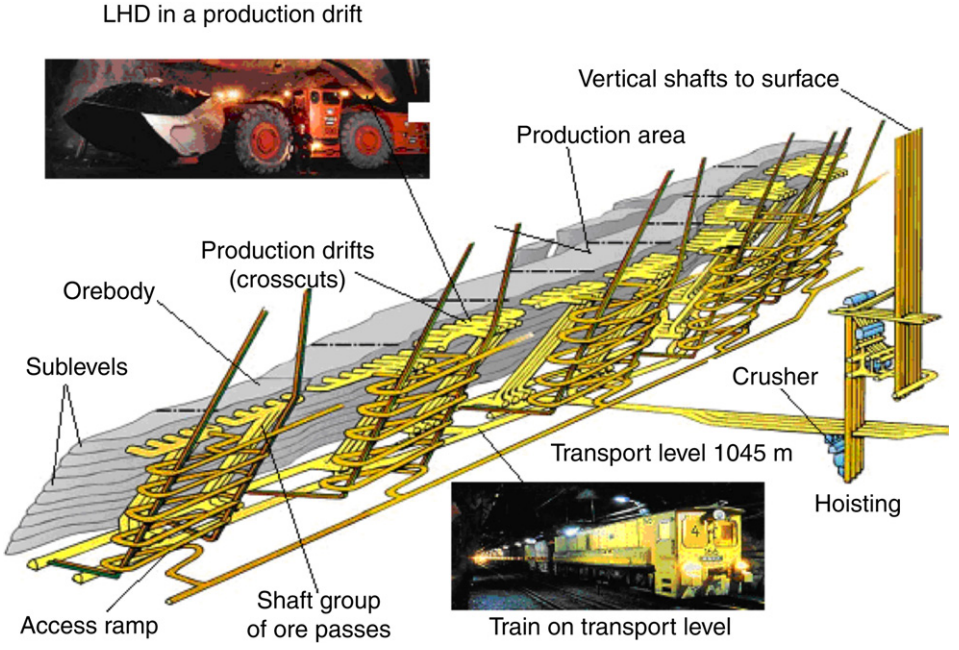

Fig. 1. The main body in the figure depicts the relationship between the production areas, ore passes, shaft groups, sublevels, the crusher and vertical shafts to the surface. The enlargement in the upper left hand corner shows a load haul dump unit, while the inset in the lower right hand corner depicts a train hauling ore to the crusher.

The goal of our scheduling model is to determine an operationally feasible sequence in which to mine the machine placements such that the amounts of B and D ore produced each month match the demanded ore quantities as closely as possible. In order for the extraction sequence to be operationally feasible, the following constraints must be observed: (i) *vertical sequencing*, i.e., an underlying machine placement cannot start to be mined until at least 50% of the machine placement directly above the underlying machine placement has been mined, (ii) *horizontal sequencing*, i.e., right and left machine placements directly adjacent to a given machine placement must start being mined as soon as 50% of the given machine placement has been mined, and (iii) no more than two or three machine placements within a shaft group can simultaneously be mined, depending on the allowable number of operational LHDs within that shaft group. If too many machines are simultaneously operational within a constrained space, the LHD trailing cables become intertwined.

Vertical sequencing constraints require that if machine placement a lies directly above machine placement b, then the earliest time at which machine placement b can start to be mined is the time at which 50% of machine placement a has been mined (subject to LHD availability). This prevents the overlying sublevel from caving onto the underlying sublevel. Horizontal sequencing constraints require that if machine placements c' and c'' are positioned directly next to machine placement a, then machine placements c' and c'' must start to be mined no later than the time at which 50% of machine placement a has been mined. This precludes adjacent machine placements from suffering blast damage caused by mining a neighboring machine placement on the same sublevel. Of course, it is impossible to start mining all the machine placements on a given sublevel at once. However, using this ''50% rule of thumb'' results in minimal blast damage. Naturally, we could change this percentage, and, in fact, the Kiruna mine has the flexibility in its production scheduling software to do so. However, we choose a value of 50% for this paper, which reflects the usual standard at the mine and lends itself towards ease of implementation for the miners. Fig. 2 shows the sequencing rules between several machine placements.

We can determine an earliest possible start time for each machine placement, i.e., a time before which a machine placement cannot have started to be mined, by applying vertical sequencing and LHD availability constraints to a given machine placement. We can similarly determine a latest possible start time for each machine placement, i.e., a time at which a machine placement must start to be mined, by applying horizontal sequencing constraints to machine placements on a sublevel with an active machine placement, i.e., a machine placement that is currently being mined. We can then use earliest and latest possible start

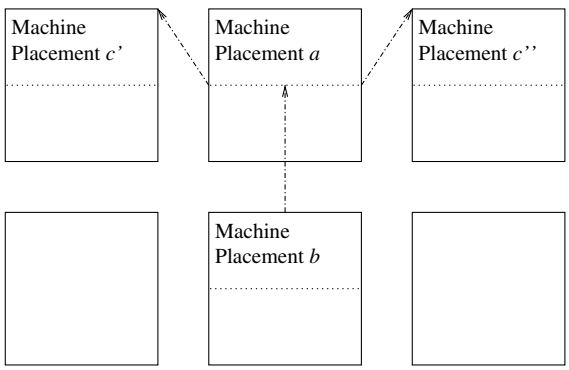

Fig. 2. Each of the six boxes represents a machine placement; a dotted line separates each machine placement into two equal halves. Dashed arrows between machine placement a and machine placements c' and c'' indicate that the latter two must be started after 50% of the former has been mined; the arrow between machine placement b and machine placement a indicates that the former may only start to be mined after 50% of the latter has been mined.

times in our model formulation as follows: (i) if the earliest and latest possible start times for a machine placement are equal, then we fix the variable value to one corresponding to starting to mine that machine placement at that start time; (ii) if a machine placement does not have a latest possible start time that occurs before the end of the horizon, then the machine placement may or may not start to be mined during the horizon, but certainly cannot start to be mined more than once during the horizon; (iii) if a machine placement has a latest possible start time that falls before the end of the horizon, then the machine placement must start to be mined during the horizon; and (iv) the set of time periods in which a machine placement can start to be mined lies between its earliest and latest possible start times.

Solving the following integer programming model provides values for the decision variables that correspond to a production schedule, i.e., whether or not to start mining a machine placement in a given time period. Our model is unique in that we address the operations of a sublevel caving mine (many underground models apply to sublevel stopping mines) with its own set of restrictions such as the inability to hold inventory and the necessity to adhere to specific sequencing constraints.

Indices

- a, a' = machine placement
- k = ore type, i.e., B, D
- t, t' = time period (month)
- v = shaft group, i.e., $1, \ldots, 10$

Sets

- T_a = set of time periods in which machine placement a can start to be mined
- A_t = set of machine placements that can be mined in time period t
- A_a^{V} = set of machine placements whose access is restricted vertically by machine placement a
- A_a^{R} = set of machine placements whose access is forced by right adjacency to machine placement a
- A_a^{L} = set of machine placements whose access is forced by left adjacency to machine placement a
- A_v = set of machine placements contained in shaft group v

Parameters

- $r_{at'tk}$ = reserves of ore type k available at (the start of) time t in machine placement a given that the machine placement started to be mined in time t' (ktons)
- d_{kt} = demand for ore type k in time period t (ktons)
- $\underline{t}_a$ = earliest start time for machine placement a
- $\overline{t}_a$ = latest start time for machine placement a
- $\mathscr{T}$ = length of the planning horizon
- LHD_v = the maximum number of simultaneously-operational LHDs in each shaft group v
- $\rho_{at't} = \begin{cases} 1 & \text{if machine placement } a \text{ is being mined in time period } t \text{ given that} \\ & \quad \text{it started to be mined in time } t' \\ 0 & \text{otherwise} \end{cases}$

Decision variables

- $y_{at} = \begin{cases} 1 & \text{if we start mining machine placement } a \text{ in time period } t \\ 0 & \text{otherwise} \end{cases}$
- $\bar{z}_{kt}$ = amount mined above the demand for ore type k in time period t (ktons)
- $\underline{z}_{kt}$ = amount mined below the demand for ore type k in time period t (ktons)

$$(P): \quad \min \quad \sum_{k,t} \underline{z}_{kt} + \sum_{k,t} \bar{z}_{kt}$$

$$\text{subject to:} \quad \sum_{a \in A_t} \sum_{t' \in T_a, \leqslant t} r_{at'tk} y_{at'} + \underline{z}_{kt} - \bar{z}_{kt} = d_{kt} \quad \forall k, t, \tag{1}$$

$$\sum_{t \in T_a, \leqslant t'} y_{at} \geqslant y_{a't'} \quad \forall a, a' \in A_a^{\mathrm{V}}, \ t' \in T_{a'}, \ a' \neq a, \tag{2}$$

$$\sum_{t' \in T_{a'}, \leqslant t} y_{a't'} \geqslant y_{at} \quad \forall a, a' \in A_a^{\mathrm{R}}, \ t \in T_a, \ a' \neq a, \tag{3}$$

$$\sum_{t' \in T_{a'}, \leqslant t} y_{a't'} \geqslant y_{at} \quad \forall a, a' \in A_a^{\mathrm{L}}, \ t \in T_a, \ a' \neq a, \tag{4}$$

$$\sum_{a \in A_v \cap A_t} \sum_{t' \in T_a, \leqslant t} \rho_{at't} y_{at'} \leqslant \mathrm{LHD}_v \quad \forall t, v, \tag{5}$$

$$y_{at} = 1 \quad \forall a \ni \underline{t}_a = \bar{t}_a, \tag{6}$$

$$\sum_t y_{at} \leqslant 1 \quad \forall a \ni \bar{t}_a > \mathscr{T}, \tag{7}$$

$$\sum_t y_{at} = 1 \quad \forall a \ni \bar{t}_a \leqslant \mathscr{T}, \tag{8}$$

$$\bar{z}_{kt}, \underline{z}_{kt} \geqslant 0 \quad \forall k, t, \tag{9}$$

$$y_{at} \quad \text{binary } \forall a, t. \tag{10}$$

The objective function minimizes the deviation from the demanded quantities of each ore type. Constraints (1) track the deviations from the demanded quantities for each ore type and time period. Constraints (2)–(4) enforce vertical and horizontal sequencing constraints. Constraints (5) ensure that no more than the allowable number of LHDs is active within a shaft group. Constraints (6) place active machine placements into the production schedule by fixing the associated variable values. Constraints (7) ensure that a machine placement starts to be mined no more than once during the time horizon if its latest possible start time occurs beyond the length of the time horizon. Constraints (8) require that a machine placement starts being mined at some point during the time horizon if its latest start time falls within the time horizon. Although redundant, the inclusion of constraints (6)–(8) greatly reduces solution time. Constraints (9) and (10) enforce nonnegativity and integrality of the variables, as appropriate. Typical model scenarios contain over 500 binary variables and more than 1000 constraints. Hereafter, for ease of exposition, we refer to the time at which a machine placement starts to be mined as a *placement start*.

3. Using aggregation to solve the production scheduling model

The principal variables in our formulation are binary variables indicating whether to start to mine a machine placement in a given time period, or not. We present an optimization-based heuristic to improve model tractability in which we solve an aggregated model to determine a set of "reasonably good" starting times for each machine placement, allowing us to restrict the model to a subset of start time choices beyond the restrictions from the early and late start algorithms. In this section, we describe the aggregation procedure and several of its variations, and conclude with a review of other aggregation techniques.

Our aggregation procedure consists of first "collapsing" our multi-period production-scheduling model to reduce its size. We aggregate demands and the amount of ore in each machine placement that can be mined in a single time period into data corresponding to *phases*, where each phase, τ, consists of an equal number of consecutive periods. So, for an original model containing 36 periods, an aggregation of two

consecutive periods into each phase would result in an aggregated model with 18 phases, where the first phase would contain time periods 1 and 2, the second phase, periods 3 and 4, etc. We term the model consisting of these aggregated phases the *aggregated model*.

$$(A): \quad \min \quad \sum_{k,\tau} \underline{z}_{k\tau} + \sum_{k,\tau} \bar{z}_{k\tau}$$

$$\text{subject to:} \quad \sum_{a\in A_\tau} \sum_{\tau'\in T_a,\leqslant\tau} r_{a\tau'\tau k} y_{a\tau'} + \underline{z}_{k\tau} - \bar{z}_{k\tau} = d_{k\tau} \quad \forall k,\tau, \tag{11}$$

$$\sum_{\tau\in T_a,\leqslant\tau'} y_{a\tau} \geqslant y_{a'\tau'} \quad \forall a,a' \in A_a^{\mathrm{V}},\ \tau' \in T_{a'},\ a' \neq a, \tag{12}$$

$$\sum_{\tau'\in T_{a'},\leqslant\tau} y_{a'\tau'} \geqslant y_{a\tau} \quad \forall a,a' \in A_a^{\mathrm{R}},\ \tau \in T_a,\ a' \neq a, \tag{13}$$

$$\sum_{\tau'\in T_{a'},\leqslant\tau} y_{a'\tau'} \geqslant y_{a\tau} \quad \forall a,a' \in A_a^{\mathrm{L}},\ \tau \in T_a,\ a' \neq a, \tag{14}$$

$$\sum_{a\in A_v\cap A_\tau} \sum_{\tau'\in T_a,\leqslant\tau} \rho_{a\tau'\tau} y_{a\tau'} \leqslant \mathrm{LHD}_v \quad \forall \tau, v, \tag{15}$$

$$y_{a\tau} = 1 \quad \forall a \ni \underline{\tau}_a = \bar{\tau}_a, \tag{16}$$

$$\sum_{\tau} y_{a\tau} \leqslant 1 \quad \forall a \ni \bar{\tau}_a > \mathscr{T}, \tag{17}$$

$$\sum_{\tau} y_{a\tau} = 1 \quad \forall a \ni \bar{\tau}_a \leqslant \mathscr{T}, \tag{18}$$

$$\bar{z}_{k\tau}, \underline{z}_{k\tau} \geqslant 0 \quad \forall k,\tau, \tag{19}$$

$$y_{a\tau} \quad \text{binary } \forall a,\tau. \tag{20}$$

Observe that the solution for (A) is feasible for (P). (At the end of Section 5, we discuss a few exceptions to this, which, in practice, are inconsequential.) Constraints (1) (in (P)) and (11) (in (A)) are elastic. Therefore, we can safely ignore any "inventory" or "backorders" carried between time periods within a phase. Constraints (2)–(4) (in (P)) impose the same constraints on machine placement sequencing as constraints (12)–(14) (in (A)) because we aggregate ore reserves in (A) by condensing *monthly production blocks*, not machine placements, into an amount that can be mined within a phase. The right hand side of (5) (in (P)) matches that of (15) (in (A)) because we do not aggregate LHD availability. Specifically, if LHD_v LHDs are allowed to be operational in shaft group v in each time period (in (P)), then LHD_v LHDs are allowed to be operational in shaft group v in each phase (in (A)). Constraints (6) (in (P)) relate to constraints (16) (in (A)) in that if a machine placement starts in a certain time period in the original model, then it must start in the corresponding phase in the aggregated model. Constraints (17) and (18) (in (A)) (modified from constraints (7) and (8) in (P)) require only mapping a late start time period for a machine placement to the corresponding phase. Therefore, a solution to (A) will satisfy the constraints in (P).

After solving the aggregated model, we add constraints to the original model, (P), to tighten the search space. We term the problem with the added constraints the *restricted problem*. We generate the additional constraints as follows: We note the phase in which each machine placement starts in the optimal solution for the aggregated model and require that the machine placement start in a time period within that phase. For example, suppose we have an aggregated model consisting of two time periods per phase. If, after solving the aggregated model to optimality, the machine placement in the aggregated model starts in phase 2, the additional constraint requires that the machine placement start in time period 3 or 4 (the two periods contained in that phase). Letting U_a be the set containing the time periods in the phase in which machine placement a starts in the aggregated model, we can state the restricted model as follows:

$$(R): \quad (P)$$
$$\text{subject to:} \quad \sum_{t \in U_a} y_{at} = 1 \quad \forall a. \tag{21}$$

Note that we could equivalently construct (R) by eliminating binary variables, rather than by adding constraints (21). Let:

- $\underline{\underline{t}}_a$ = the first time period in the phase in which machine placement a starts to be mined in the aggregated model,
- $\overline{\overline{t}}_a$ = the last time period in the phase in which machine placement a starts to be mined in the aggregated model.

Then, another form of the restricted model follows:

$$(R'): \quad (P)$$

replacing $T_a = \{\underline{t}_a, \ldots, \bar{t}_a\}$ with $\widetilde{T}_a = \{\underline{\underline{t}}_a, \ldots, \overline{\overline{t}}_a\}$ $\forall a$.

Note that for any a, $\underline{t}_a \leqslant \underline{\underline{t}}_a$ and $\bar{t}_a \geqslant \overline{\overline{t}}_a$. However, for the bounds provided in the set $\widetilde{T}_a$ to be useful, we seek some machine placements a such that: $\underline{t}_a < \underline{\underline{t}}_a$ and $\bar{t}_a > \overline{\overline{t}}_a$.

Because we are, in essence, changing the objective function in the aggregated model (from the original model) by eliminating penalties for deviations incurred during consecutive time periods within a phase, we cannot expect to obtain an optimal solution by solving (R) (or (R')).

We could use the solution from the aggregated model in a less restrictive way to enhance the solution quality of the original model. Specifically, we could allow a machine placement start in the time periods contained in the $n > 0$ phases adjacent to that in which the machine placement starts in the aggregated model, as well as the time periods in the phase in which the machine placement starts in the aggregated model. However, there is no guarantee that the optimal solution will improve by adding this flexibility in the restricted model. An integer program (unlike a linear program), does not adhere to convexity properties. That is, a near-optimal solution is not necessarily "close" (in a Hamming distance sense) to an optimal solution (Rardin, 1998, chapter 12). Nonetheless, we demonstrate empirically in Section 5 that some degree of relaxation provides near-optimal solutions. In Section 4, we show how to compute a bound on the worst case performance of the aggregation heuristic.

Improving the tractability of integer programming problems using aggregation is not new. Rogers et al. (1991) provide an excellent survey paper on the application of aggregation and disaggregation techniques to optimization models and specifically to integer programming models. The authors discuss how model attributes are "clustered," how to construct a solution to the original problem after the aggregated model is solved, and how to estimate errors resulting from the aggregation procedure. The authors cite Mathews (1897) who demonstrates how several constraints can be combined into a single equivalent constraint with the appropriate choice of constant multipliers for each disaggregated constraint. Related work, e.g., Glover and Woolsey (1972) and Kannan (1983), examine the computational burden associated with constraint aggregation.

Aggregation techniques depend highly on the structure of the problem, and, in general, are tailored specifically for a class of problems or even for a specific problem instance. Research subsequent to 1990 on aggregation techniques used to solve integer programs includes Daichendt and Grossmann (1994), who use aggregation to solve mixed integer nonlinear optimization problems. Specifically, they aggregate both variables and constraints, the latter by creating linear combinations of subsets of constraints in the original model, such that the aggregated model can be solved to global optimality. They use the results from the aggregated model to provide tight bounds on the global optimal solution of the original problem, which significantly helps to identify good solutions to ill-behaved nonlinear models. Kulkarni and Mohanty

(1996) aggregate a multi-time period model for siting warehouses and planning the distribution of goods from these warehouses to geographically dispersed customers. The model is first solved as an aggregated single-period model, and time period-specific decisions are then sequenced over the horizon. Babayev et al. (1997) reformulate the integer knapsack problem using constraint aggregation, and develop a highly efficient solution procedure for the reformulated, but equivalent, model.

Srinivasa and Wilhelm (1997) use constraint aggregation to help obtain solutions for a resource allocation problem, specifically, for a model that minimizes the amount of time required to clean up an oil spill. Iyer et al. (1998) determine decisions for scheduling an offshore oil field operation over multiple time periods. The authors solve a model in which oil wells and time periods are aggregated, and then successively disaggregated to produce a solution to the original problem. The disaggregation is myopic in that some decisions from the aggregated model are fixed in the disaggregated model, and the disaggregated model is solved over only portions of the entire time horizon. The authors devise bounds on the quality of the disaggregated solution and demonstrate empirically that their solutions are within about 10% of the optimal. Bertsimas and Patterson (2000) seek to minimize the cost of delay from rerouting aircraft due to dynamically changing weather conditions. Their problem is a large integer program, which they solve for real instances to near-optimality in several stages, first for aggregate aircraft movements, then for individual aircraft flight paths. Similar to Iyer et al., Harjunkoski and Grossmann (2001) use a disaggregation procedure to schedule steel production, a highly sequence-dependent process with multiple end products. The authors' procedure is based on several levels of disaggregation in which groups in the lowest level are scheduled independently, and then with respect to other groups. A linear programming improvement procedure is used to make minor modifications in the schedule. Results indicate that the authors are able to find solutions within only a few percentage points of optimality in a reasonable amount of time.

Researchers also use aggregation to help solve dynamic programs, e.g., Mohanty and Singh (1992) for (deterministic) hierarchical production planning in a steel plant, and Puelz (2002) for stochastic portfolio selection. In these cases, aggregation reduces the curse of dimensionality typical of dynamic programming problems.

Constraint aggregation can be used to develop a general class of constraints, *mixed-integer rounding inequalities*, whose addition to large mixed integer programming problems can reduce solution time (Marchand and Wolsey, 2001). Belvaux and Wolsey (2001) show that for a particular class of multi-product distribution and sales problems, the addition of mixed-integer rounding inequalities, obtained by aggregating constraints in the original model, enhances solution quality.

Our aggregation heuristic uses information gained from a model in which time periods and the associated demand and production data are aggregated into phases. However, our procedure differs from others mentioned here in that: (i) we solve a disaggregated model, but use information gained from the aggregated model to eliminate variables, and/or (ii) our procedure can be applied to any general multi-period production scheduling problem, see, e.g., Martin (1999, Section 1.3.4), and the references contained therein.

4. Worst-case performance

In this section, we show how to compute a bound on the worst case performance of our aggregation procedure for the implementation we describe in Section 3. This bound is computable, rather than theoretical, and relies on the solution generated from (A). The aggregated model only accounts for, i.e., minimizes, deviations *between* each phase, and ignores deviations that occur *within* a phase. In the worst case, for each phase and ore type, the optimal disaggregated schedule produces everything in the time period with the minimum demand, and nothing in the other time periods. By solving (A), we are able to compute $\mathscr{P}$, the production level in the aggregated model for one phase, and for a fixed ore type. Using this quantity, and letting d_t represent the demand in time period t, and represent the set of time periods contained in

the given phase, for each phase, the maximum deviation resulting from restricting the original model to mine a machine placement in one of the time periods t contained in the associated phase is

$$\left|\mathscr{P} - \min_{t\in\Upsilon}\{d_t\}\right| + \sum_{t\ni t\neq \operatorname{argmin}_{t\in\Upsilon}\{d_t\}} d_t, \tag{22}$$

where the first term represents the maximum overproduction, and the second term represents the maximum underproduction.

The maximum total deviation in the worst case is the sum of these expressions above, one for each ore type and for each phase of the aggregated model added to the deviation between the time periods that separate the phases. (The latter term consists of those deviations that appear in the objective function of the aggregated model.)

Consider the following single ore type, single phase example: Suppose that the phase consists of three time periods, with demands in periods 1, 2, and 3 of 1 (ton), 2 (tons), and 3 (tons), respectively. Suppose also that the solution to the aggregated model produces 4 (tons) in the single phase. In the worst case, the disaggregated model produces $\mathscr{P}$ tons, i.e., the total amount produced in the phase, in the period with the least demand (period 1), and nothing in the other two periods. This yields a (maximum) deviation of

$$\left|\mathscr{P} - \min_{t\in\{1,2,3\}}\{d_t\}\right| + \sum_{t\ni t\neq \operatorname{argmin}_{t\in\{1,2,3\}}\{d_t\}} d_t = |4 - \{d_1\}| + \sum_{t\in\{2,3\}} d_t = |4-1| + \{2+3\} = 8(\text{tons}).$$

Were our objective only to consider underproduction, the worst possible outcome would be to produce nothing in any time period, resulting in a 6 ton deviation, i.e., a deviation equal to the demand. However, the total worst-case deviation (8 tons) exceeds the total demand, because we penalize overproduction as well.

5. Numerical results

We illustrate empirically the benefits of our aggregation method using five actual data sets from the mine. Each data set contains a group of 36-month periods, and has two ore types. Ore reserves for each type are between zero and 180 ktons per production block. Demands are about 830 ktons of B and 1250 ktons of D ore each month. One, two, or three machine placements are allowed to be simultaneously active within each of the ten shaft groups.

We solve our mixed integer program using the AMPL programming language (Fourer et al., 1993; Bell Laboratories, 2001), and the CPLEX solver, Version 7.0 (ILOG Corporation, 2001). We use a Sun Ultra 10 machine with 256 MB RAM. To further reduce solution time, in each scenario we run, we implement priority branching, choosing the variables in increasing order of their time period index. Hence, we branch first on variables corresponding to a placement start in time period one, followed by variables corresponding to a placement start in time period two, etc. We implement our aggregation procedure after eliminating variables using the early start algorithm (Newman et al., 2005). We also eliminate variables that would violate the latest start time, $\bar{t}_a$, for each machine placement. Each phase consists of two time periods.

For each scenario, after solving the aggregated model, we solve the restricted model, that is, a corresponding disaggregated model in which we impose constraints of varying flexibility. The most restrictive set of constraints requires that the time period in which each machine placement starts be contained in the phase in which the machine placement starts in the aggregated model. We alternatively impose constraints of increasing flexibility that require that the time period in which each machine placement starts be contained either in the phase in which the machine placement starts in the aggregated model, or in one of the n phases immediately before or immediately after the phase in which the machine placement

starts in the aggregated model. For example, if $n = 1$ in an aggregated model which contains two time periods per phase, we allow a placement start in the following six periods: in the two periods before the phase starts, in the two periods contained in the phase, and in the two periods after the phase. We then form the set $U_a\ \forall a$ and add (21), as in (R). Equivalently, we could determine $\underline{t}_a$ and $\overline{t}_a\ \forall a$, and replace the set T_a with $\widetilde{T}_a$, as in (R').

In general, the quality of the solutions obtained with the restricted model increases at the expense of slower solution times. We seek a reasonable tradeoff between solution time and solution quality using our aggregation procedure. In Fig. 3, we present such a tradeoff for values of n ranging between 0 and 5. Solution quality is given as the ratio of the optimal objective function value to the objective function value obtained from the aggregation procedure. Solution time is given as the ratio of that required for the aggregation procedure to that required by (P). Each point on both the solution quality and solution time curves corresponds to average values across five scenarios for a given value of n. The solution quality curve exhibits diminishing marginal returns and the quality converges to 100% as the window size n increases. The solution time ratio increases sharply with the value of n. As the window size increases, solution times approach a significant fraction of those required by (P).

Fig. 3 indicates that for our model and data, the best tradeoff between solution quality and solution time is roughly where $n = 2$. Table 1 gives numerical values for each of the five scenarios for this best case implementation. Column 1 lists the scenario number; column 2 gives the objective function value for (P) and column 3 gives the objective function value provided by the aggregation procedure. Columns 4 and 5 list the solution times required for (P) and for the aggregation procedure, respectively. On average, solutions

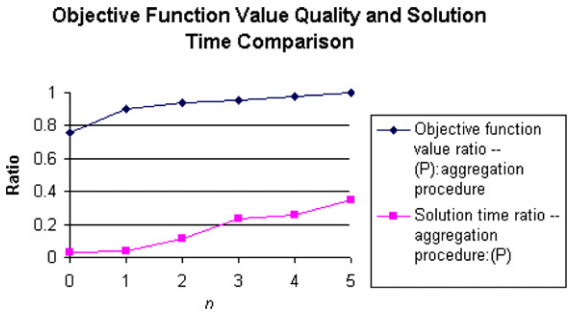

Fig. 3. A comparison of the objective function value quality of (P) compared to that obtained with the aggregation procedure is given in the top curve on the graph. As n increases, the aggregation procedure yields objective function values that approach optimal. In the bottom curve on the graph, the ratio of the solution time of the aggregation procedure to that required by (P) is given. Solution time increases with n. Each point is the average across five scenarios.

Table 1
Comparison of objective function values and solution time for (P) and using the aggregation procedure (two time periods per phase and $n = 2$)

Scenario	Objective of (P) (ktons)	Objective from aggregation procedure (ktons)	Solution time for (P) (seconds)	Aggregation procedure solution time (seconds)
1	3280	3784	1200	432
2	2852	2966	17,000	1330
3	1766	1890	130,000	3300
4	2034	2166	18,000	670
5	3536	3538	15,000	740

within about 5% of optimality can be obtained with the aggregation procedure in just over 10% of the solution time required to solve (P). That is, our aggregation procedure generates solutions of acceptable quality in an order of magnitude less time than that required to solve (P). For these instances, the aggregated models contain about 260 binary variables and 530 constraints; the restricted models contain about 350 binary variables and 740 constraints. Recall, instances of (P) contain over 500 binary variables, on average, and over 1000 constraints.

We emphasize that these conclusions hold for our particular model structure and representative data sets. For any general multi-period scheduling model and corresponding data sets, the best tradeoff between solution quality and solution time could well lie at another value for n.

The solution times for (P) used as the 100% measure in Fig. 3 result from running the model until optimality, i.e., until the gap is 0, although theoretically very good solutions may be found much earlier. However, if we allow (P) to run for the same amount of time as that required by the best implementation of our aggregation procedure, objectives from (P) are significantly worse, 19% on average, than those obtained by using our aggregation procedure.

Other (larger) models may lend themselves to the use of aggregated models with coarser granularity that are faster to solve but that provide looser restrictions on the corresponding original model, which increases solution times of the restricted model. In fact, we conducted numerical experiments aggregating four time periods into a phase. However, for our instances, we find the granularity with more than two time periods per phase is too coarse to provide precise enough information for use in the original model. And, this loss of information comes at the expense of solution quality, which, in our experiments, is not offset by overall faster solution times.

Although the aggregation procedure performs well on our data sets, for $n = 0$ the restricted model may be infeasible if the number of monthly production blocks within a machine placement is not evenly divisible by the number of time periods aggregated into a phase. These infeasibilities are caused by round-off errors in the disaggregated sequencing constraints. For example, if we have three monthly production blocks in a given machine placement and each phase consists of two periods (months), phase 1 will contain two monthly production blocks and phase 2 will contain 1. Sequencing constraints are based on a ''50% rule,'' yet it is not clear how to impose this rule on a single production block. Any choice will yield an inconsistency between (P) and the aggregated models. Because the case in which $n = 0$ is not likely to yield the best tradeoff between solution quality and solution time, in practice, the effect of these infeasibilities is negligible.

We were able to use this method to generate very long-term (10–15 year) plans for the Kiruna mine using different types of data sets containing two and three raw ore types. These schedules help guide several different strategic planning decisions: (i) the time at which the new main production level should start to be constructed, and (ii) the number of ore types to consider. The latter decision is evaluated by examining the tradeoff between a higher degree of differentiation among the quality of the ore types and a potentially higher deviation between desired and actual production quantities, for three versus two raw ore types. At the time of the writing of this paper, the mine plans to use these results to aid in its long-term strategic decisions; the determination of these decisions is still ongoing. Kiruna does use the long-term model (without the aggregation procedure) to generate three- to five-year schedules. Kuchta et al. (2004) describe Kiruna's implementation of this model over the shorter time horizon, and the resulting savings.

6. Conclusions and extensions

Faced with a large integer programming model for production scheduling, we use an aggregation procedure to first establish ''reasonable'' start times for each machine placement, and then impose constraints on (P) to restrict machine placement start times. While not an optimal procedure, we find empirically that we obtain schedules of reasonable quality when compared both with those schedules found by solving (P)

and with the quality necessary in practice. Long-term strategic planning is essential for large underground mines in order to best utilize available resources. The benefit of using our procedure is a substantial reduction in solution time without significant loss of solution quality which facilitates the rapid evaluation of alternative mining strategies.

Extensions to our work include improving the quality of our production schedules by, for example, incorporating constraints on the minimum or maximum allowable deviation between "phases" into the aggregated model, which could result in better information being passed to the restricted model. We could also explore other forms of the objective function, such as minimizing cumulative deviation, or minimizing successive time periods of underproduction and successive periods of overproduction. These objectives may smooth under- and over-production, although in our numerical experiments, we found Kiruna's mine geometry and ore body composition to be sufficiently restrictive so as to result in relatively few solutions, regardless of the form of the objective.

Acknowledgements

The authors would like to thank LKAB for the opportunity to work on this challenging project and for permission to publish these results. The authors would specifically like to thank LKAB employees Hans Engberg, Anders Lindholm, and Jan-Olov Nilsson for providing information, assistance, and support in this endeavor. The authors would also like to thank Professor Kevin Wood of the Naval Postgraduate School, Professor Candi Yano of the University of California, Berkeley, and several anonymous referees for helpful comments and suggestions on earlier drafts.

References

Almgren, T., 1994. An approach to long range production and development planning with application to the Kiruna mine, Sweden. Luleå University of Technology, Doctoral Thesis number 1994:143D.

Babayev, D., Glover, F., Ryan, J., 1997. A new knapsack solution approach by integer equivalent aggregation and consistency determination. INFORMS Journal on Computing 9 (1), 43–50.

Bell Laboratories, 2001. AMPL, Version 10.6.16.

Belvaux, G., Wolsey, L., 2001. Modeling practical lot-sizing problems as mixed-integer programs. Management Science 47 (7), 993–1007.

Bertsimas, D., Patterson, S., 2000. The traffic flow management rerouting problem in air traffic control: A dynamic network flow approach. Transportation Science 34 (2), 239–255.

Carlyle, M., Eaves, B., 2001. Underground planning at stillwater mining company. Interfaces 31 (4), 50–60.

Dagdelen, K., Kuchta, M., Topal, E., 2001. Linear programming model applied to scheduling of iron ore production at the Kiruna mine, Kiruna, Sweden. Transactions of the Society for Mining, Metallurgy, and Exploration, Inc. 312, 194–198.

Daichendt, M., Grossmann, I., 1994. Preliminary screening procedure for the MINLP synthesis of process systems—I. Aggregation and decomposition techniques. Computers and Chemical Engineering 18 (8), 663–677.

Fourer, R., Gay, D., Kernighan, B., 1993. AMPL: A Modeling Language for Mathematical Programming. Boyd and Fraser, Massachusetts.

Glover, F., Woolsey, R., 1972. Aggregating diophantine equations. Zeitschrift für Operations Research 16 (1), 1–10.

Harjunkoski, I., Grossmann, I., 2001. A decomposition approach for the scheduling of a steel plant production. Computers and Chemical Engineering 25, 1647–1660.

ILOG Corporation, 2001. CPLEX, Version 7.0.

Iyer, R., Grossmann, I., Vasantharajan, S., Cullick, A., 1998. Optimal planning and scheduling of offshore oil field infrastructure investment and operations. Industrial and Engineering Chemistry Research 37, 1380–1397.

Kannan, R., 1983. Polynomial-time aggregation of integer programming problems. Journal of the Association for Computing Machinery 30 (1), 133–145.

Kuchta, M., Newman, A., Topal, E., 2004. Implementing a production schedule at LKAB's Kiruna mine. Interfaces 34 (2), 124–134.

Kulkarni, R., Mohanty, R., 1996. Multilocation plant sizing and timing problem: A study of spatial–temporal decomposition procedure. Production Planning and Control 7 (5), 471–481.
Marchand, H., Wolsey, L., 2001. Aggregation and mixed integer rounding to solve MIPs. Operations Research 49 (3), 363–371.
Martin, R., 1999. Large Scale Linear and Integer Optimization. Kluwer Academic Publishers, Boston, MA.
Mathews, G., 1897. On the partition of numbers. Proceedings of the London Mathematical Society 28, 486–490.
Mohanty, R., Singh, R., 1992. A hierarchical production planning approach for a steel manufacturing system. International Journal of Operations and Production Management 12 (5), 69–78.
Newman, A., Kuchta, M., Martinez, M., 2005. Long- and short-term production at LKAB's Kiruna mine. Working Paper, Division of Economics and Business and Mining Engineering Department, Colorado School of Mines, Golden, CO, January.
Puelz, A., 2002. A stochastic convergence model for portfolio selection. Operations Research 50 (3), 462–476.
Rardin, R., 1998. Optimization in Operations Research. Prentice Hall, Upper Saddle River, NJ.
Rogers, D., Plante, R., Wong, R., Evans, J., 1991. Aggregation and disaggregation techniques and methodology in optimization. Operations Research 39 (4), 553–582.
Sarin, S., West-Hansen, J., 2005. The long-term mine production scheduling problem. IIE Transactions 37 (2), 109–121.
Smith, M., 1998. Optimizing short-term production schedules in surface mining: Integrating mine modeling software with AMPL/CPLEX. International Journal of Surface Mining, 149–155.
Smith, M., Sheppard, I., Karunatillake, G., 2003. Using MIP for strategic life-of-mine planning of the lead/zinc stream at Mount Isa Mines. In: 31st International APCOM Symposium Proceedings, Capetown, South Africa, pp. 465–474.
Srinivasa, A., Wilhelm, W., 1997. A procedure for optimizing tactical response in oil spill clean up operations. European Journal of Operational Research 102 (3), 554–574.
Topal, E., 1998. Long and short term production scheduling of the Kiruna iron ore mine, Kiruna, Sweden. Master of Science Thesis, Colorado School of Mines, Golden, Colorado.
Trout, L., 1995. Underground mine production scheduling using mixed integer programming. In: 25th International APCOM Symposium Proceedings, Brisbane, Australia, pp. 395–400.
Winkler, B., 1996. Using MILP to optimize period fix costs in complex mine sequencing and scheduling problems. In: 26th International APCOM Symposium Proceedings, Pennsylvania State University, University Park, Pennsylvania, pp. 441–446.