Integrated Mine Planner

Yuki Osada

Lyndon While
School of Computer Science and Software Engineering

Luigi Barone
CEED Client: SolveIT Software

Abstract

Planning is an important process in mining extraction to achieve long-term goals, or to fulfil contracts with their clients. Mining operations typically involve a number of operational units, each independently responsible for the delivery of a product subject to nominated performance specifications. In order to meet any performance targets for the overall operation, these operational units cannot work in isolation, and cooperation among them is required. To tackle this problem, the question we face is: how best to meet the performance targets of the overall mining operation subject to the limitations of the operational units. We will describe a decision support tool that could be used to simplify the process of coordinating the outputs of several operational units to meet a global target, and improve the quality of these plans.

1. Introduction

Mining companies typically have several mine-sites, and several contracts from their customers that they need to fulfil. These contracts are independent of the capabilities of the individual mines, and so it is not guaranteed that these contracts can be met with a single mine. In such a situation, if there is no coordination between the outputs of the different mines, there would be no way to ensure that the contracts will be met. This would be a major issue for these mining companies, costing them heavily, if they couldn't be sure that a contract could be met. To avoid this issue, planning is a requirement. If each mine had equal capability and the same limitations, one could evenly distribute the contract requirements over all the mines, but this is rarely the case. With each mine having their own properties, the requests made to each mine must be adjusted accordingly so that the different mines cover up for each other.

Coordinating the outputs of several mines can be a difficult task when planning out for the life of each mine. Although there are tools for mine planning (Runge 2012), they are mainly for planning out the life of a single mine. As these tools deal with single mines, the coordination of the outputs of each mine must be done separately. Currently, this coordination is mostly done manually, which makes this an unnecessarily long process due to the overheads involved, and can cost the company valuable resources. This planning process could be significantly improved with the aid of a decision support system. Not only could resources be saved and used elsewhere, it would also be possible to explore other future possibilities. For example, a mining company could do scenario planning, where what-if scenarios could be assessed more easily. It could better equip these companies to promptly respond to new opportunities or risks.

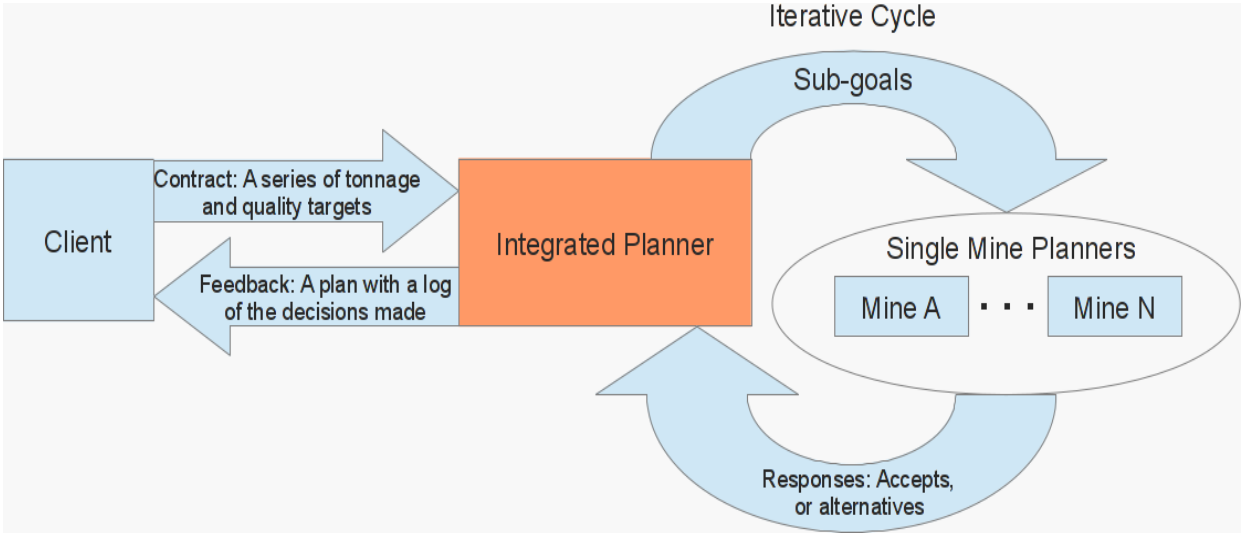

Figure 1 An overview of the integrated planner. The system first receives a contract from the user. After receiving the contract, the system enters a negotiation cycle with the single mine planners to find a plan that all mines are capable of, and best meets the given contract.

The idea of a computer system that deals with multiple mines is not new (CAE Mining 2010), but there is currently no known software system that takes a global contract, and distributes the load over several mines to set a high level target for each mine while respecting their limitations. One possible approach to this planning problem is to treat it as one large planning problem. While this approach could produce good results as all the information about any mine would be available to all the others, it can also require a lot of work to set up correctly. Another approach is to separate the planning process from the coordination process whereby a single mine planner is used to actually make the plans, and on top of that, a plan coordinator is placed to ensure that the combined plan fulfils a given contract. This is the approach that SolveIT Software has chosen, so that is what I will be looking at. More specifically, it is the coordination part that I will be focusing on.

The two main questions we want to answer are: is such a system feasible, and if so, how does the quality of the results change as we change the execution time allowed for the system. For the system to be feasible, there must be an advantage in using the system. This means it needs to produce good plans in an acceptable time frame for example. We may have to deal with long contract durations with many performance constraints, and with several mines, so this could make it very difficult to find a set of plans that meet a given contract. This could result in excessive computational time or human intervention that could nullify any advantages. The two main concerns with such a system are the time it takes to find a solution, and the quality of the solution, so it would be useful to know the trade off between these two objectives.

There may be no known software system for this task, but this type of problem is not unique to mine planning. If we can cast this problem into a more established framework, we will be able to apply past research, and prior knowledge to this new problem. One possible way to view this problem is as a multi-agent plan coordination problem (de Weerdt, M, ter Mors, A & Witteveen, C 2005), where a set of plans must be devised for a group of agents such that the combined result meets the goals of the system. An agent is a system that interacts with the world, so in our problem, the single mine planners are our agents. The agents in our problem also have differing capabilities reflecting the limitations of each mine.

2. Process

There are many ways for a set of plans to be constructed and coordinated in this problem, so it has been important to explore different ideas, and run them through several test cases. We first started with a static rule-based approach where we constructed a set of rules to handle every state it may encounter. This proved to be a difficult task due to the limited knowledge we had of mining operations. Even in our tests, constructing rules that worked "intelligently" was something of a black art. To construct these rules, we would lay out every combination of possible properties that an individual mine planner could respond with, and how they should be treated based on the responses of the other mines. A request to a single mine could span multiple years, and there could be many parameters defining a requested output for each of these years. Examples of these parameters may include total tonnage, iron percentage, and percentages of various contaminants. For each of these parameters, the response may meet them to different extents, and these extents define the properties of the response. Naturally, since all the rules were being hard-coded, it meant that the system operated under any arbitrary assumptions or prioritisations that we may have made when the rules were created. This constrained the system's ability to the ability of the programmer.

Other than a rule-based approach, we tried using optimisation techniques like hill climbing, and evolutionary algorithms (Bäck & Hammel & Schwefel 1997). Basically, the centre of the system is deciding what is the best next set of requests to make to the single mine planners. The rule-based approach would make decisions based on the arbitrary choices we may have made during the writing of the software, but these optimisation techniques allow a lot more exploration, and it can pick the strategy that returned the best result. The idea is to evaluate many simply constructed responses instead of crafting a perfect set of rules to produce the best response – evaluation is easier than innovation. Not only does this significantly simplify the system while giving it much more sophisticated behaviour, fewer choices are arbitrarily chosen by the programmer thus opening it up to producing higher quality results. This could reveal new strategies, and provide the users of the system with new ideas.

To test the coordination procedure, we needed to implement a simple single mine planner simulator. An actual single mine planner would be very difficult, so the simulator would just produce responses given to it as an input. This was useful in testing the behaviour of the system to different responses, but we had to be careful to keep the responses consistent. It is challenging to make realistic test cases that test the quality and performance of the system, so ideally we would want to test against an actual single mine planner with real mine data. This will be something we will do later on in the project.

3. Results and Discussion

Treating this coordination problem as an optimisation problem is expected to scale a lot better than the rule-based approach, which needed a rule defined for every possible state the system could be in. With our rule-based approach, we could have situations where the rules behave poorly and other situations where they perform well. This is due to the fact that there are a large number of possible states, and once we start generalising some states, or miss any states, then the system begins to perform poorly. Another difficulty with the rule-based approach was the utilisation of past responses from the single mine planners. As there is no noise in their responses, any plan that a single mine planner accepts will always be a valid plan.
The current proposed approach uses an optimisation approach which maintains a "best" plan during each iteration of the negotiation process. The general idea is as follows:

1. At each iteration, run an optimisation algorithm to find "plausible" requests that bring us closer to the target. These requests can be designed to work together to meet the global goal, or they can be individually designed to work with known previous responses of the other mines.
2. Make the requests to the single mine planners, and archive the responses together with the respective requests. This archive of past requests and responses give us information to evaluate the "fitness", or quality, of future requests. This evaluation of fitness is a requirement in any evolutionary algorithm. For example, a new request that is similar to an existing response has a higher chance of being accepted, so they may have a higher fitness than a request that is similar to a past request that had a poor response.
3. Identify the best set of responses in the archive that uses at least one response from the current iteration. If this plan is "better" than the previous running best, then store this plan. With a running best, it is possible for the user to terminate the system at any point in time, and still receive some result. It also demonstrates to the user that the system is still progressing if the user can see the result improving over time.
4. If the best plan meets the required contract, then we can return this to the user, otherwise move onto the next iteration.

This approach avoids all the little details that the rule-based system needed while producing more reliable requests in new situations.

3.1 Issues

This approach still requires some refining, and there are still some important issues to solve. These include the definitions for the "plausibility" of a request, and a "better" plan. The system should be able to produce acceptable results with any reasonable definition for these terms, but a complete and better definition will allow the system to converge onto preferred results in fewer iterations during the negotiation cycle. This should allow for better time performance, and thus better quality results.

3.1.1 The Initial Plan

Other unresolved issues include deciding what is an ideal initial plan, and how to rank request sets that bring us closer to our target. The first plan, or the first set of plans, can influence how the system will progress over time as the information we gather at these steps could determine how well and fast it'll converge on a solution. This can be considered the sampling phase to gather initial data on these mines. Examples include distributing the contract requirements equally over all the mines, giving each mine the whole contract, making a series of requests that focus on specific requirements, or possibly a combination of these. Each of these have their advantages and disadvantages. Distributing the contract equally could work well when each mine has similar capabilities thus only requiring minor tweaks to the requests. Stressing the mines by giving them the whole contract, or making very specialised requests allow us to get an idea of the limits of a mine. Taking more initial samples will give the system more information to make intelligent decisions, but it also has the cost that more requests are needed before we can even start. Finding the balance here is important in avoiding over-complication of the system. Another solution is to have the user provide an initial plan. This expert-provided information could potentially reduce the time needed to find a good solution, but it could also hinder the exploration of very different solutions.

3.1.1 What Makes a Good Set of Requests

Ranking request sets is also tricky problem that could influence the rate at which the system converges onto solutions. The difficulty comes from the fact that some requests could be good when targeting past responses in the archive rather than a set of requests that work together. In this situation, we could have a set of requests which contain good and bad requests. This means such a request set has a chance of being very useful as well as very useless, but disregarding it could be a waste. What we want would be to rank this request set high enough, so it is improved upon in the optimisation algorithm, but we want the more reliable ones to be ranked higher when it comes to actually selecting one.

3.2 As a Decision Support Tool

Interactivity of the system is another goal of the system to make it a useful decision support tool. The approach we outlined above allows fully autonomous operation of the system, but it also has the possibility of semi-autonomous operation. At every iteration of the negotiation with the single mine planners, it is possible for the user to intervene at step 1, where the next set of requests are constructed. The user of the system could supply their own set of requests at this step instead of letting the system decide if they wished to do so. This can also be beneficial to the system as the responses to those requests have a good chance of being useful when the user is an expert in this problem.

4. Conclusions and Future Work

With the aid of a computer system, it should make managing the plans of multiple mines easier. It can also provide all the benefits of a decision support system as the current approach outlined above supports interactive use. This means the user could interact, and watch the system display rationales for different actions to reveal new approaches to thinking about the problem. This is also only possible because we are not limiting the system's capabilities to the programmer as it uses optimisation techniques like evolutionary algorithms to explore and evaluate different solutions.

The current implementation does behave well, and has the potential to perform well on real test data, but the next step would be produce concrete results. This includes testing how the quality of plans change as we vary the number of requests made to the single mine planners, how much time it takes to find solutions in realistic data where we are guaranteed to have a plan that meets the requirements perfectly. This would require implementing a more complete simulator, or getting access to an actual single mine planner.

There are many strategies for this problem, and this problem can also be cast into other classes of problems. The approach we are taking may not be the best approach, so for future work, it would be useful to test even more approaches including other standard planning strategies to further improve the performance and quality of an integrated planner.

5. References

CAE Mining 2010, *Multimine Scheduler,* CAE. Available from: <http://www.cae.com/en/mining/_pdf/Mine%20Planning/MultimineScheduler_Datasheet_English_201003.pdf>. [20 August 2012].

Runge 2012, *XPAC*. Available from: <http://www.runge.com/mining-software/xpac>. [3 September 2012].

Bäck, T, Hammel, U & Schwefel, H.-P 1997, 'Evolutionary computation: comments on the history and current state', *IEEE Transactions on Evolutionary Computation*, vol. 1, no. 1, pp. 3-17.

de Weerdt, M, ter Mors, A & Witteveen, C 2005, 'Multi-agent planning: An introduction to planning and coordination', *Handouts of the European Agent Summer School*, pp. 1-32. Available from: CiteSeerX. [3 September 2012]