Metaheuristics for scheduling production in large-scale open-pit mines accounting for metal uncertainty - Tabu search as an example

Amina Lamghari
Outline

• Introduction

• Overview of Tabu Search

• Adaptation of Tabu Search to solve the open-pit mine production scheduling problem with metal uncertainty

• Conclusions

Not an exhaustive presentation
Only an introduction to some basic ideas and principles
The big picture

Decision problem
Want, should, cannot

Modeling

Optimization model
Minimize (max) some function of decision variables
subject to some constraints

Solving

Solution analysis
Verification, validation, sensitivity analysis
Optimization models

Minimize (maximize) some function of decision variables subject to some constraints

<table>
<thead>
<tr>
<th>Continuous</th>
<th>Discrete (Integer, binary)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Single criterion</td>
<td>Multi-criteria (conflicting objectives)</td>
</tr>
<tr>
<td>Unconstrained or few constraints</td>
<td>Many constraints (difficult to find a feasible solution)</td>
</tr>
<tr>
<td>Linear</td>
<td>Nonlinear</td>
</tr>
<tr>
<td>Deterministic data</td>
<td>Stochastic data</td>
</tr>
</tbody>
</table>

Its probability distribution is known
The big picture

Decision problem
Want, should, cannot

Modeling

Optimization model
Minimize (max) some function of decision variables
subject to some constraints

Solving

Solution analysis
Verification, validation, sensitivity analysis
# Solving optimization problems

<table>
<thead>
<tr>
<th>Easy</th>
<th>Difficult</th>
</tr>
</thead>
<tbody>
<tr>
<td>Continuous</td>
<td>Discrete (Integer, binary)</td>
</tr>
<tr>
<td>Single criterion</td>
<td>Multi-criteria (conflicting objectives)</td>
</tr>
<tr>
<td>Unconstrained or few constraints</td>
<td>Many constraints</td>
</tr>
<tr>
<td>Linear</td>
<td>Nonlinear</td>
</tr>
<tr>
<td>Deterministic data</td>
<td><strong>Stochastic data</strong></td>
</tr>
</tbody>
</table>

**Exact solution procedures**
(Simplex, Branch & Bound…)

**Approximate solution techniques**
(Heuristics / metaheuristics…)

Not efficient for difficult large-scale problems (prohibitive CPU time, memory…)

A heuristic: fits a specific problem and take advantage of its structure

A metaheuristic: provides general structure and strategy guidelines

**Used to rapidly come to a solution hoped to be close to the optimal**
The descent method

At each iteration, Select the **best improving solution** $x'$ in $N(x)$ as the new current solution

The method stops at the **first local optimum** reached from the **starting solution**
The local optimum issue

Assume neighbour solutions are adjacent points in this Solution Space. No more improvement in the neighbourhood!

Note that the global optimum is also a local optimum

Tabu search

To **avoid being trapped in a local optimum**, **move** to a new solution $x'$ in $N(x)$ **even if** this leads to **no improvement** or a **deterioration** of the objective function

At each iteration, select the **best improving solution** $x'$ in $N(x)$
The idea is nice but …

At each iteration, select the **best solution** $x'$ in $N(x)$

Risk of cycling back to previously visited solutions
Avoiding cycles

- Use a list to keep track of the $T$ recently visited solutions (history of the search)

- When considering a new solution, first check if it is in this list. If so, discard it

- Pro: instantly eliminates any possibility that a cycle of length $T$ occurs

- Con: requires a lot of storage

- Con: checking whether a potential solution is in the list or not may be computationally costly
Avoiding cycles

- Record the recent moves performed on the current solution and forbid moves that reverse their effect.

- These “forbidden” moves are declared Tabu and are disregarded when exploring the neighborhood for some number of iterations, called the Tabu tenure of the move.

- Tabus are stored in a short-term memory of the search, called Tabu list.

- Should be long enough to prevent cycling and short enough to not exclude more moves than necessary.

- An aspiration criterion is used to override the Tabu status of some attractive solutions during the process.
Exploring the neighborhood: decision tree

For each move leading to a neighboring solution of the current solution

Is the move in the Tabu list?
- Yes
- No

Does the move satisfy the aspiration criterion?
- Yes
  - Include the resulting solution in the candidate list
- No
  - Consider the next move

Include the resulting solution in the candidate list

Select the best solution in the candidate list as the new current solution
Update the Tabu List
Stopping criteria

- A fixed number of iterations
- A fixed amount of CPU time
- After some number of iterations without an improvement in the objective function value
- When the objective reaches a pre-specified threshold value
An illustrative problem

The Classical Vehicle Routing Problem

A depot at which a vehicle of limited capacity Q is based
A set of customers that need to be serviced
Each customer has a demand to be collected by the vehicle

Find a set of routes such that:

- Each route begins and ends at the depot
- Each customer is visited exactly once by exactly one route
- The total demand of the customers assigned to each route does not exceed Q
- The total distance traveled by the vehicle is minimized

The Stochastic Vehicle Routing Problem

The demand of each customer is uncertain

Typically formulated as a recourse model:

- Whenever the residual load of the vehicle reaches a specified threshold, the vehicle returns to the depot to unload and resumes collections at the next planned customer
- Minimize the expected distance traveled by the vehicle
A simple neighborhood

Current solution

Or

Provided that route R2 has sufficient residual capacity
Tabu list: some possibilities

• Moving 1 back from R2 to R1 is Tabu
  Include (1, R2, R1) in the Tabu list

• Moving 1 back to R1 is Tabu (without consideration for its current route)
  Include (1, R1) in the Tabu list

• Moving 1 to any route is Tabu
  Include (1) in the Tabu list

• Will not constrain the search much
  Cycling may occur if 1-->R3 then 1-->R1

• Stronger
  May prohibit a solution with a value better than that of the best-known solution
  Aspiration criterion will allow the move

• Even stronger
  May lead to an overall stagnation of the searching process
Basic Tabu search is good but…

- “Too local”: tends to spend most of its time in a restricted portion of the search space
- Although **good solutions** may be obtained, one may fail to **explore the most interesting parts of the search space**

- **Diversification**: a mechanism to force the search into previously unexplored parts of the search space
- Usually based on some form of **long-term memory** such as frequency memory
- Two major techniques
  - Continuous diversification
  - Restart diversification
Continuous diversification

- Integrate diversification considerations into the searching process

- Bias the evaluation of possible moves by adding to the objective a small term related to the component frequencies (minimization problem)

**Modified move value = Move value + diversification parameter * frequency measure**

- A possible frequency measure in the SVRP application would be the number of times the customer involved in the move has been moved from its current route

\[ f(x_1) = 25 \text{ and } \text{freq } (1,R_1) = 2 \]

\[ f(x_2) = 26 \text{ and } \text{freq } (6,R_2) = 0 \]

Diversification parameter = 0.75
Restart diversification

- Force a few rarely used components in the **current solution** or the **best-known solution**
- Restart the search from **this point**
- A possible strategy in the SVRP application would be to force customers that have not yet been moved frequently into new routes

**Current solution**

```
1 2 3
4 5 6
```

freq (1,R1) = 3, freq (1,R2) = 2
freq (2,R1) = 3, freq (2,R2) = 2
freq (3,R1) = 5, freq (3,R2) = 0
freq (4,R1) = 2, freq (4,R2) = 3
freq (5,R1) = 0, freq (5,R2) = 5
freq (6,R1) = 1, freq (6,R2) = 4

**New initial solution**

```
5 6 1 2
4 3
```

Current solution or best-known solution?
To what extent?
Feasibility?
Random, greedy?
The big picture

Decision problem
Want, should, cannot

Modeling

Optimization model
Minimize (max) some function of decision variables
subject to some constraints

Solving

Solution analysis
Verification, validation, sensitivity analysis
Evaluating metaheuristics

- **Quantitative criteria**
  - **Efficiency**: the method should provide *optimal or near-optimal* solutions to *realistic problem instances* (comparison with optimal solutions if known, with Lower/Upper bounds if known, or with other metaheuristics)
  - **Effectiveness**: *good solutions* should be achieved *within a reasonable amount of time*
  - **Robustness**: *efficiency and effectiveness* should prevail *for a variety of problem instances* (not just fine-tuned to some training set and less good elsewhere)

- **Qualitative criteria**
  - **Simplicity**: *simple* and *clear* principles should govern the method
  - **Flexibility**: the method should be *easily adapted* to deal with new problem variants
  - **User-friendliness**: the method should be easy to understand and *most important easy to use*. This implies it should have as few parameters as possible
Basic questions to ask when implementing Tabu Search

<table>
<thead>
<tr>
<th>Tabu search</th>
</tr>
</thead>
<tbody>
<tr>
<td>Initial solution</td>
</tr>
<tr>
<td>Neighborhood structure</td>
</tr>
<tr>
<td>Tabu (attributes, tenure, classification)</td>
</tr>
<tr>
<td>Aspiration criteria</td>
</tr>
<tr>
<td>Diversification strategy?</td>
</tr>
<tr>
<td>Stopping criteria</td>
</tr>
</tbody>
</table>

And then test and check ...

- Efficient?
- Effective?
- Robust?
A variant of the Stochastic OPMPS

• Determine the extraction order of blocks from a mineral deposit over a given time horizon

• Constraints
  - Reserve constraints
  - Slope constraints
  - Mining constraints (lower and upper limits)
  - Processing constraints (lower and upper limits)
  - Metal production constraints (lower and upper limits)

• Objective function
  - Maximize the expected NPV
  - Minimize deviations from production targets
Basic questions to ask when implementing Tabu Search

<table>
<thead>
<tr>
<th>Tabu search</th>
<th>Initial solution</th>
</tr>
</thead>
<tbody>
<tr>
<td>Neighborhood structure</td>
<td></td>
</tr>
<tr>
<td>Tabu (attributes, tenure, classification)</td>
<td></td>
</tr>
<tr>
<td>Aspiration criteria</td>
<td></td>
</tr>
<tr>
<td>Diversification strategy?</td>
<td></td>
</tr>
<tr>
<td>Stopping criteria</td>
<td></td>
</tr>
</tbody>
</table>

And then test and check ...

- Efficient?
- Effective?
- Robust?
Initial solution

Select a block randomly

Candidate blocks

A new period (t+1) is initiated whenever the total weight of the blocks extracted in t is greater or equal than 1/2(maxToExtract + minToExtract)

<table>
<thead>
<tr>
<th>1</th>
<th>2</th>
<th>.....</th>
<th>i</th>
<th>...</th>
<th>j</th>
<th>.......</th>
<th>k</th>
<th>N</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>2</td>
<td></td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td>T+1</td>
<td>T+1</td>
</tr>
</tbody>
</table>
Basic questions to ask when implementing Tabu Search

<table>
<thead>
<tr>
<th>Tabu search</th>
</tr>
</thead>
<tbody>
<tr>
<td>Initial solution</td>
</tr>
<tr>
<td>Neighborhood structure</td>
</tr>
<tr>
<td>Tabu (attributes, tenure, classification)</td>
</tr>
<tr>
<td>Aspiration criteria</td>
</tr>
<tr>
<td>Diversification strategy?</td>
</tr>
<tr>
<td>Stopping criteria</td>
</tr>
</tbody>
</table>

And then test and check ...

- Efficient?
- Effective?
- Robust?
Neighborhood Structure

**Neighborhood**

Shift block $i$ from period $t_1$ to another period $t_2$

The solution generated belongs to the neighborhood if it is feasible
Basic questions to ask when implementing Tabu Search

And then test and check ...

- Efficient?
- Effective?
- Robust?

Tabu search

<table>
<thead>
<tr>
<th>Initial solution</th>
</tr>
</thead>
<tbody>
<tr>
<td>Neighborhood structure</td>
</tr>
<tr>
<td>Tabu (attributes, tenure, classification)</td>
</tr>
<tr>
<td>Aspiration criteria</td>
</tr>
<tr>
<td>Diversification strategy?</td>
</tr>
<tr>
<td>Stopping criteria</td>
</tr>
</tbody>
</table>
Tabu and aspiration criteria

Do not allow reversing recent shifts (are declared Tabu for a certain number of iterations)

Except if they lead to a solution better than the best solution found so far (aspiration)
Basic questions to ask when implementing Tabu Search

<table>
<thead>
<tr>
<th>Tabu search</th>
</tr>
</thead>
<tbody>
<tr>
<td>Initial solution</td>
</tr>
<tr>
<td>Neighborhood structure</td>
</tr>
<tr>
<td>Tabu (attributes, tenure, classification)</td>
</tr>
<tr>
<td>Aspiration criteria</td>
</tr>
<tr>
<td>Diversification strategy?</td>
</tr>
<tr>
<td><strong>Stopping criteria</strong></td>
</tr>
</tbody>
</table>

And then test and check ...

- Efficient?
- Effective?
- Robust?
Stopping criterion

The local optimum issue

Assume neighbour solutions are adjacent points in this Solution Space.

No more improvement in the neighbourhood!

Note that the global optimum is also a local optimum

$n_{itermax}$ successive iterations where the objective does not improve
Basic questions to ask when implementing Tabu Search

- Efficient?
- Effective?
- Robust?

And then **test** and check ...

<table>
<thead>
<tr>
<th>Tabu search</th>
</tr>
</thead>
<tbody>
<tr>
<td>Initial solution</td>
</tr>
<tr>
<td>Neighborhood structure</td>
</tr>
<tr>
<td>Tabu (attributes, tenure, classification)</td>
</tr>
<tr>
<td>Aspiration criteria</td>
</tr>
<tr>
<td><strong>Diversification strategy?</strong></td>
</tr>
<tr>
<td>Stopping criteria</td>
</tr>
</tbody>
</table>
Diversification strategy

Restart the search from a **new point** (more extensive search)

Apply successively a sequence of shifts in order to generate the **new initial solution**
Diversification strategy

• Apply successively a sequence of shifts in order to generate the new initial solution

• Applied to

<table>
<thead>
<tr>
<th>TS</th>
<th>VNS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Current best solution (generated during the last search)</td>
<td>Best-known solution (found so far)</td>
</tr>
</tbody>
</table>

• Choose the shifts
  Trying to move the search towards unexplored or less explored areas of the solution space

  Trying to obtain a good quality solution
Solution procedure

Initial solution

Tabu search
to increase the expected NPV and reduce constraints violations

Diversification strategy
to generate a new initial solution

Repeat as long as the elapsed time is less than a specified maximum time
Basic questions to ask when implementing Tabu Search

<table>
<thead>
<tr>
<th>Tabu search</th>
</tr>
</thead>
<tbody>
<tr>
<td>Initial solution</td>
</tr>
<tr>
<td>Neighborhood structure</td>
</tr>
<tr>
<td>Tabu (attributes, tenure, classification)</td>
</tr>
<tr>
<td>Aspiration criteria</td>
</tr>
<tr>
<td>Diversification strategy?</td>
</tr>
<tr>
<td>Stopping criteria</td>
</tr>
</tbody>
</table>

And then test and check ...

- Efficient?
- Effective?
- Robust?
Numerical results

- 2 sets of problems $P_1, P_2$. In each set, 5 problems having different sizes
  - $P_1$: from a copper deposit
  - $P_2$: from a gold deposit
- Each problem is solved 10 times using different initial solutions

<table>
<thead>
<tr>
<th>Set</th>
<th>Problem</th>
<th>Number of blocks ($N$)</th>
<th>Number of periods ($T$)</th>
</tr>
</thead>
<tbody>
<tr>
<td>$P_1$</td>
<td>C1</td>
<td>4,273</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>C2</td>
<td>7,141</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>C3</td>
<td>12,627</td>
<td>7</td>
</tr>
<tr>
<td></td>
<td>C4</td>
<td>20,626</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td>C5</td>
<td>26,021</td>
<td>13</td>
</tr>
<tr>
<td>$P_2$</td>
<td>G1</td>
<td>18,821</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td>G2</td>
<td>23,901</td>
<td>7</td>
</tr>
<tr>
<td></td>
<td>G3</td>
<td>30,013</td>
<td>8</td>
</tr>
<tr>
<td></td>
<td>G4</td>
<td>34,981</td>
<td>9</td>
</tr>
<tr>
<td></td>
<td>G5</td>
<td>40,762</td>
<td>11</td>
</tr>
</tbody>
</table>

Table 1: Problems data
Basic questions to ask when implementing Tabu Search

<table>
<thead>
<tr>
<th>Tabu search</th>
</tr>
</thead>
<tbody>
<tr>
<td>Initial solution</td>
</tr>
<tr>
<td>Neighborhood structure</td>
</tr>
<tr>
<td>Tabu (attributes, tenure, classification)</td>
</tr>
<tr>
<td>Aspiration criteria</td>
</tr>
<tr>
<td>Diversification strategy?</td>
</tr>
<tr>
<td>Stopping criteria</td>
</tr>
</tbody>
</table>

- **Efficiency:** the method should provide optimal or near-optimal solutions to realistic problem instances (comparison with optimal solutions if known, with Lower/Upper bounds if known, or with other metaheuristics)

- **Effectiveness:** good solutions should be achieved within a reasonable amount of time

- **Robustness:** efficiency and effectiveness should prevail for a variety of problem instances (not just fine-tuned to some training set and less good elsewhere)

And then test and check ...

- Efficient?
- Effective?
- Robust?
Numerical results: Copper deposit

\[ \% \text{Average Gap} = \frac{Z_{LR} - Z_{\text{average}}}{Z_{LR}} \times 100 \]

Worth using metaheuristics

**CPLEX:** much more CPU
- C5 (N=26,021, T=13): fails to solve the LR within 4 weeks VS 2 hours
- C4 (N=20,626, T=10): 9 days VS 1 hour

**Metaheuristics:** small gap
- In average: < 4% (1.76% and 3.26%)

**TS outperforms VNS**
- Improves %Gap over VNS by 46%
- Performance differences increase with the problem size
Numerical results: Problem C5

Cross-sectional view of the best schedule generated by **TS**

Cross-sectional view of the best schedule generated by **VNS**

**TS** improves the value of objective function over **VNS** by 20%
Numerical results: Gold deposit

\[
\% \text{Average Gap} = \frac{Z_{LR} - Z_{\text{average}}}{Z_{LR}} \times 100
\]

Worth using metaheuristics

<table>
<thead>
<tr>
<th>Metaheuristics</th>
<th>CPLEX: much more CPU</th>
</tr>
</thead>
<tbody>
<tr>
<td>G5 (N=40,762, T=11): fails to solve the LR within 4 weeks VS 2.5 hours</td>
<td></td>
</tr>
<tr>
<td>G4 (N=34,981, T=9): 11 days VS 2 hours</td>
<td></td>
</tr>
</tbody>
</table>

**TS:** very small gap

In average: < 2% (1.83%)

**TS outperforms VNS**

Improves %Gap over VNS by 96%

Differences more pronounced %Gap in [1.15, 2.40] VS [16.64, 57.17]
Cross-sectional view of the best schedule generated by TS

Cross-sectional view of the best schedule generated by VNS

**TS improves the value of objective function over VNS by 126%**
TS and VNS require significantly shorter CPU than CPLEX (LR)

**TS** more effective and more robust than **VNS**
Conclusions

• Metaheuristics are powerful algorithmic approaches which have been applied with great success to many difficult optimization problems including Stochastic Combinatorial optimization problems.

• They allow dealing with large size problems having high degree of complexity, and generate rapidly very good solutions.

• Optimality is not guaranteed in general and few convergence results are known for special cases.
Conclusions

<table>
<thead>
<tr>
<th>Tabu search</th>
</tr>
</thead>
<tbody>
<tr>
<td>Initial solution</td>
</tr>
<tr>
<td>Neighborhood structure</td>
</tr>
<tr>
<td>Tabu (attributes, tenure, classification)</td>
</tr>
<tr>
<td>Aspiration criteria</td>
</tr>
<tr>
<td>Diversification strategy?</td>
</tr>
<tr>
<td>Stopping criteria</td>
</tr>
</tbody>
</table>

**Further readings:**

Conclusions

- Metaheuristics cannot be applied blindly to any problem. Generally, *ad hoc* adaptations are used to deal with specific applications.

  Significant knowledge and understanding of the problem at hand is absolutely required to develop a successful metaheuristic implementation.