CoPPo: Hand-in assignment 2

Group 15
Yingjie Huang, Jingkai Zhou

Task I: The classical Job Shop Problem

MILP in Gurobi

To solve this task using Gurobi, we first create a non-negative integer variable $\mathbf{x}_{k,j}$ to represent the start time of j-th job on k-th machine. Then a 3-dimensional binary variable $\mathbf{y}_{k,i,j}$ is needed to represent mutual exclusion between i-th job and j-th job on machine k, i.e., every pair of jobs on every machine, assuming that $\mathbf{y}_{k,i,j} = 0$ if i-th job is prior to j-th job on machine k and vice versa. Finally, another integer variable $\mathbf{z}$ represents makespan, which is also the objective that we want to minimize.

To make it simple, we reuse all the notations in the description of task 1, consequently, the processing time on h-th operation of j-th job can be expressed as $p_{\sigma_h^j,j}$. Since MILP is implemented, the big-$\mathbf{M}$ constant is required. In this case, we select it as $J \cdot M \cdot \mathbf{max}\{p_{\sigma_h^j,j}\}$, meaning that the maximum possible makespan if all operations are done in a row, which is big enough.

Thus constraints can be formulated as following:

1. $\mathbf{x}_{\sigma_h^j,j} + p_{\sigma_h^j,j} \leq \mathbf{x}_{\sigma_{h+1}^j,j}, \quad \forall\ j \in J,\ \forall\ h \in M \backslash \{m\}$;
2. $\mathbf{x}_{k,i} + p_{k,i} \leq \mathbf{x}_{k,j} + \mathbf{M} \cdot \mathbf{y}_{k,i,j}, \quad \forall\ i,j \in J,\ i \neq j,\ \forall\ k \in M$;
3. $\mathbf{x}_{k,i} \geq \mathbf{x}_{k,j} + p_{k,j} - \mathbf{M} \cdot (1 - \mathbf{y}_{k,i,j}), \quad \forall\ i,j \in J,\ i \neq j,\ \forall\ k \in M$;
4. $\mathbf{x}_{\sigma_m^j,j} + p_{\sigma_m^j,j} \leq \mathbf{z}, \quad \forall\ j \in J$.

As mentioned before, the objective function to be minimized is simply the makespan $\mathbf{z}$:

$$\mathbf{MIN}\{\mathbf{z}\}.$$

CSP in Z3

Using the same notations as in Gurobi, but remove the decision variables $\mathbf{y_{k,i,j}}$, we can model this problem in a CSP way as below:

1. $\mathbf{x}_{\sigma_h^j,j} + p_{\sigma_h^j,j} \leq \mathbf{x}_{\sigma_{h+1}^j,j}, \quad \forall\ j \in J,\ \forall\ h \in M \backslash \{m\}$;
2. $\mathbf{x}_{k,i} + p_{k,i} \leq \mathbf{x}_{k,j} \quad \texttt{OR} \quad \mathbf{x}_{k,i} \geq \mathbf{x}_{k,j} + p_{k,j}, \quad \forall\ i \in J,\ \forall\ j \in J \backslash \{i\},\ \forall\ k \in M$;
3. $\mathbf{x}_{\sigma_m^j,j} + p_{\sigma_m^j,j} \leq \mathbf{z}, \quad \forall\ j \in J$.

The objective function to be minimized in this case is still the same:

$$\textbf{MIN}\{\mathbf{z}\}.$$

Results

For both Gurobi and Z3, we obtain the same optimal objectives as table 1.

Test case	FT06	FT10	LA01	LA02	LA03	LA04	LA05
Optimal objective	55	930	666	655	597	590	593

Table 1: Optimal objectives in task I

However, the execution time of these two solvers are quite different as illustrated in table 2. In general, MILP using Gurobi is faster than CSP using Z3 on average, especially when solving FT10, but Z3 is slightly faster in some test cases.

Solver \ **Test case**	FT06	FT10	LA01	LA02	LA03	LA04	LA05
Gurobi	73ms	20”517ms	565ms	3”82ms	1”746ms	1”609ms	407ms
Z3	65ms	3’43”639ms	366ms	4”429ms	736ms	855ms	7”399ms

Table 2: Execution time in task I on RTX3060

Task II: The Extended Job Shop Problem

To implement A* algorithm to find the shortest distances for every pair of nodes, a graph has to be initialized first. Eight nodes, denoted as V, are created such that 0 represents warehouse, 1-6 represent machines and 7 represents delivery. Then we have to manually build 28 edges, denoted as E, directly connecting every two nodes with initial distances such that no third node is involved in the path. To make the implementation simple while programming, the non-directed graph is converted to a bi-directed graph, i.e., another 28 edges are added with the same corresponding value but reversed start node and end node. Finally, a heuristic, denoted as h, has to be built. Since we want to compute the shortest distances for every two nodes, meaning that the source node and sink node will be each node in turn, thus the heuristic can be simply selected as the initial edge values in the graph divided by a constant to avoid overestimate.

A binary decision variables $\mathbf{x}_{i,j}$ representing the selection of each edge from node i into node j is needed, where 1 means an edge is selected in the optimal path while 0 means not selected. Since a legal path can contain only one single edge from the source node, and also only one single edge into the sink node, and for all other intermediate nodes, the number of in-edges must be equal to the number of out-edges, meaning that if you go into an intermediate node, you must get out from that node in order to reach the sink node in the end. So the constraints are formed as following:

1. $\sum_j \mathbf{x}_{\text{source},j} = 1, \quad \forall\, j \in V \backslash \{\text{source}\}$;

2. $\sum_i \mathbf{x}_{i,\text{sink}} = 1, \quad \forall\, i \in V \backslash \{\text{sink}\};$

3. $\sum_i \mathbf{x}_{i,j} = \sum_i \mathbf{x}_{j,i}, \quad \forall\, i \in V \backslash \{j\}, \quad \forall\, j \in V.$

As for the objective function, it has to be minimized over the sum of all selected edge values and corresponding heuristic, which can be organized as below:

$$\textbf{MIN}\left\{\sum_i \sum_j \mathbf{x}_{i,j} \cdot (e_{i,j} + h_{j,\text{sink}}), \quad \forall\, i \in V,\ \forall\, j \in V \backslash \{i\},\ \forall\, e \in E\right\}.$$

By setting every pair of nodes as source node as sink node in turn, the solver is able to find all shortest paths as table 3 below:

Node / Node	Warehouse	1	2	3	4	5	6	Delivery
Warehouse	0	90	70	142	64	28	155	161
1	90	0	54	52	64	62	65	71
2	70	54	0	72	120	42	119	91
3	142	52	72	0	116	114	81	53
4	64	64	120	116	0	78	91	135
5	28	62	42	114	78	0	127	133
6	155	65	119	81	91	127	0	100
Delivery	161	71	91	53	135	133	100	0

Table 3: Shortest paths in task II

To illustrate A* searching algorithm, here is the pseudo code. The main idea of this algorithm is to maintain two lists, namely `open_set` and `closed_set`, and decide on the next node with minimum estimate path value ($\text{path\_cost}_{\text{source},i} + h_{i,\text{sink}}$).

Denote the obtained shortest distance between node i and node j as $d_{i,j}$, where node 0 represents warehouse, 1-6 represent machines and 7 represents delivery, and reuse all other notations in task 1, we can now model this problem with the following constraints using a float variable $\mathbf{x}_{k,j}$ instead of an integer as in task 1:

1. $\mathbf{x}_{\sigma_1^j,j} \geq \frac{d_{0,\sigma_1^j}}{5}, \quad \forall\, j \in J;$

2. $\mathbf{x}_{\sigma_h^j,j} + p_{\sigma_h^j,j} + \frac{d_{\sigma_h^j,\sigma_{h+1}^j}}{5} \leq \mathbf{x}_{\sigma_{h+1}^j,j}, \quad \forall\, j \in J,\ \forall\, h \in M \backslash \{m\};$

3. $\mathbf{x}_{k,i} + p_{k,i} \leq \mathbf{x}_{k,j} + \mathbf{M} \cdot \mathbf{y}_{k,i,j}, \quad \forall\, i,j \in J,\ i \neq j,\ \forall\, k \in M;$

4. $\mathbf{x}_{k,i} \geq \mathbf{x}_{k,j} + p_{k,j} - \mathbf{M} \cdot (1 - \mathbf{y}_{k,i,j}), \quad \forall\, i,j \in J,\ i \neq j,\ \forall\, k \in M;$

5. $\mathbf{x}_{\sigma_m^j,j} + p_{\sigma_m^j,j} + \frac{d_{\sigma_m^j,7}}{5} \leq \mathbf{z}, \quad \forall\, j \in J.$

The objective function is actually the same as in task 1:

$$\textbf{MIN}\{\mathbf{z}\}.$$

Feeding above conditions into the solver, we obtain a new optimal objective for FT06 as 177.

Algorithm 1 A-star Algorithm

initialize `open_list` := empty list, `closed_list` := empty list, g(`source_node`) := 0
add `source_node` to `open_list`
while `open_list` is not empty **do**
  q := node with MIN$\{$g(q)+$h_{q,\text{sink\_node}}\}$ in `open_list`
  for each q' that can be directly reached from q **do**
    if q == sink_node **then**
      end this algorithm
    else
      f(q') := g(q) + $e_{q,q'}$
    end if
    if q' in `closed_list` **then**
      continue
    end if
    if q' in `open_list` **then**
      if f(q') $\leq$ g(q') **then**
        g(q') := f(q')
      end if
    else
      g(q') := f(q')
      add q' to `open_list`
    end if
  end for
  remove q from `open_list`
  add q to `closed_list`
end while

Task III: Finding the Shortest Path using Z3

The idea is pretty similar to task 2, but as only boolean variables are allowed in this task, two helper functions in Z3, which are `PbEq` and `PbLe`, are needed. Moreover, due to a slightly different constraint expression for intermediate nodes, a non-directed graph shall be fed into the solver instead of a bi-directed graph as used in task 2.

To begin with, create two boolean decision variables, $\mathbf{x}_i$ representing whether node i is selected or not, and $\mathbf{y}_{i,j}$ representing whether an edge connecting node i and node j is selected or not. Note that we use a non-directed graph in this task, so the order of i and j doesn't matter.

'Exactly one' constraint shall be added to both source and sink nodes, but for all other nodes, the new constraint now is 'exactly two'. We first illustrate these constraints in a mathematical way as below:

1. $\sum_i \mathbf{y}_{i,\text{source}} = 1, \quad \forall\, i \in V \backslash \{\text{source}\};$
2. $\sum_i \mathbf{y}_{i,\text{sink}} = 1, \quad \forall\, i \in V \backslash \{\text{sink}\};$
3. $\sum_j \mathbf{y}_{i,j} + 2 \cdot (\sim \mathbf{x}_i) = 2, \quad \forall\, i \in V \backslash \{\text{source, sink}\},\ \forall\, j \in V \backslash \{i\}.$

The mathematical objective function can formed as below:

$$\textbf{MIN} \left\{ \sum_i \sum_j \mathbf{y}_{i,j} \cdot e_{i,j}, \quad \forall\, i \in V,\ \forall\, j \in V \backslash \{i\},\ \forall\, e \in E \right\}.$$

However, boolean variables can't be summed up in Z3, thus the corresponding constraints can be interpreted as follows in Z3:

1. $\texttt{PbEq}([(\mathbf{y}_{i,\text{source}}, 1) \quad \forall\, i \in V \backslash \{\text{source}\}],\ 1);$
2. $\texttt{PbEq}([(\mathbf{y}_{i,\text{sink}}, 1) \quad \forall\, i \in V \backslash \{\text{sink}\}],\ 1);$
3. $\texttt{PbEq}([(\texttt{Not}(\mathbf{x}_i), 2)(\mathbf{y}_{i,j}, 1) \quad \forall\, j \in V \backslash \{i\}],\ 2) \quad \forall\, i \in V \backslash \{\text{source, sink}\}.$

There's also some trick to implement the objective function in Z3 as follows:

$$\textbf{MIN}\{\texttt{sum}([\texttt{If}(\mathbf{y}_{i,j}, e_{i,j}, 0) \quad \forall\, i \in V,\ \forall\, j \in V \backslash \{i\},\ \forall\, e \in E])\}.$$

With above conditions, Z3 can find the shortest distance and exact path between machine 5 and machine 3, but we have to add one more constraint after each path is found to ban this path by not allowing this specific combination of path. Denote the set of nodes involved in the newly found shortest path as $\tilde{V}$, thus the new constraint can be formed as below, which means that not all the edges that are involved in the newly found shortest path can be selected at the same time when searching a new one:

$$\texttt{PbLe}([(\mathbf{y}_{i,j},\ 1) \quad \forall\, i \in \tilde{V},\ \forall\, j \in \tilde{V} \backslash \{i\}],\ |\tilde{V}| - 1).$$

By adding this new constraint after each iteration, it's possible to find all paths between machine 5 and machine 3, and the 10 shortest paths and corresponding distances are listed as table 4:

No.	Distance	Path
1	114	$5 \to 1 \to 3$
2	114	$5 \to 2 \to 3$
3	148	$5 \to 2 \to 1 \to 3$
4	156	$5 \to 3$
5	170	$5 \to 0 \to 3$
6	170	$5 \to 0 \to 1 \to 3$
7	186	$5 \to 1 \to 7 \to 3$
8	186	$5 \to 2 \to 7 \to 3$
9	188	$5 \to 1 \to 2 \to 3$
10	194	$5 \to 4 \to 1 \to 3$

Table 4: 10 shortest paths in task III