CoPPo: Hand-in assignment 3

Group 15
Yingjie Huang, Jingkai Zhou

Task I: The Tower of Hanoi

We follow the same method as defined in the attachment file, and denote $on(d, tw, t)$, $obj(d, t)$, $from(tw, t)$ and $to(tw, t)$ as $\mathbf{x}_{d,tw,t}$, $\mathbf{y}_{d,t}$, $\mathbf{m}_{tw,t}$ and $\mathbf{n}_{tw,t}$ respectively. On top of that, in order to turn this problem into an optimization problem, one more decision variable $\mathbf{z}_t$ is needed to count the steps. Moreover, we need some extra steps to enable optimization, so assume maximum number of steps $ts = 2^{ds}$.

Then basically all the constraints are the same as the ones in attachment, and interpretation of these constraints can be found in delivered codes. However, some minor modifications are required to make it an optimization problem as following.

1. Uniqueness of $obj/\mathbf{z}_t$: originally it says exactly one step has to be taken at every time instance, but in our optimization problem, it should instead be at most one step can be taken at every time instance, since extra steps are given, meaning that disks don't have to be moved as long as they all reach the final state. In Z3, this constraint can be interpreted as follwoing.

$$[\texttt{PbLe}([(\mathbf{y}_{d,t}, 1)\ \texttt{for}\ d\ \texttt{in}\ \texttt{range}(ds)], 1)\ \texttt{for}\ t\ \texttt{in}\ \texttt{range}(ts)]$$

2. Counting constraint: at each time step, we shall count the steps that we take even if we know it's either 1 or 0 so that we can set our objective later on. This constraint can be coded as follows in Z3.

$$\texttt{And}([\mathbf{z}_t == \texttt{sum}([\texttt{If}(\texttt{Or}([\mathbf{y}_{d,t}\ \texttt{for}\ d\ \texttt{in}\ \texttt{range}(ds)]), 1, 0)])\ \texttt{for}\ t\ \texttt{in}\ \texttt{range}(ts)])$$

3. Objective: in this task, we want to minimize the total number of steps, which can be simply expressed as below.

$$\textbf{MIN}\left\{\sum_{t=0}^{ts}\mathbf{z}_t\right\}$$

In this way, we can solve this optimization problem and get all results as expected.

Task II: A General Sorting Problem

2.1 - Single robot

2.1.1 - Bricks without classes

Firstly, the maximum number of steps in this task is $ts = \lceil 3 \cdot \frac{bs}{2} \rceil$, where bs is the number of bricks. This is because the most complicated situation is that two bricks need to switch their positions, which requires 3 steps to solve this pair, and there are $\lceil \frac{bs}{2} \rceil$ pairs of bricks at most.

Similar to task 1, bricks and positions can be regarded as disks and towers respectively, so we denote decision variables as $\mathbf{x}_{b,p,t}$, $\mathbf{y}_{b,t}$, $\mathbf{z}_t$, $\mathbf{m}_{p,t}$ and $\mathbf{n}_{p,t}$ for this task, where b and p represent some certain brick and some certain position.

As for the constraints, we can mostly reuse them from task 1 with subscript changes, but some modifications are also needed.

1. Precondition I: not needed anymore as no more than one brick can be placed at one position at the same time.

2. Precondition II: shall be modified as no more than one brick can be placed at one position at the same time, which can be interpreted as

$$\begin{cases} \mathbf{x}_{b,p,t} \wedge (\mathbf{x}_{1,p',t} \vee \dots \mathbf{x}_{b-1,p',t} \vee \mathbf{x}_{b+1,p',t} \vee \dots \mathbf{x}_{bs,p',t}) \Rightarrow \neg(\mathbf{y}_{b,t} \wedge \mathbf{n}_{p',t}), \\ b \in [1, bs], \quad p \in [1, ps], \quad p' \in [1, ps] \setminus \{p\}, \quad t \in [1, ts]. \end{cases}$$

3. Initial/final state: in task 1, every disk shall be moved to the last tower, but in this task, every brick has its unique target position, so this constraint can be modified as below

$$\begin{cases} \mathbf{x}_{b,p_{\text{b\_initial}},0} \wedge \mathbf{x}_{d,p_{\text{b\_target}},ts}, \\ b \in [1, bs], \quad p \in [1, ps]. \end{cases}$$

Finally, the objective function is the same as task 1, namely

$$\textbf{MIN} \left\{ \sum_{t=0}^{ts} \mathbf{z}_t \right\}.$$

With this setup, this model is scalable to any number of bricks and positions in theory. But in practice, the computational and memory burden have to be taken into consideration. So to figure out the feasible sizes for this solver, we randomly generate test cases in an ascending order of bricks and positions shown as table 1, from which we can see that execution time increases exponentially in the number of bricks and positions, and becomes dramatically large since 9 bricks as no solution is available within 30 minutes.

# bricks	1	2	3	4	5	6	7	8
# positions	2	3	4	5	6	7	8	9
Time	41ms	29ms	125ms	345ms	1”6ms	2”612ms	7”490ms	3’7”959ms

Table 1: Execution time in task II on RTX3060

2.1.2 - Bricks with classes

If the bricks are divided into different classes, the only change is the initial/final state constraint. Instead of pushing every single brick to move to its target position as defined above, this constraint shall be relaxed to accept every single brick to move to any places involved in its class' target positions as illustrated below.

$$\begin{cases} \mathbf{x}_{b,p_{\text{b\_initial}},0} \wedge (\mathbf{x}_{d,p_{\text{cls\_target\_1}},ts} \vee \dots \mathbf{x}_{d,p_{\text{cls\_target\_n}},ts}), \\ b \in [1, bs], \quad p \in [1, ps]. \end{cases}$$

2.2 - Multiple robots

When it comes to multiple robots, we reuse all decision variables as in single robot case, but $\mathbf{y}_{b,t}$ shall be modified as $\mathbf{y}_{r,b,t}$, where r represents some certain robot. Hence, all constraints involving $\mathbf{y}_{b,t}$ shall be modified accordingly, and some more changes are needed as following.

1. Precondition: if any brick is at some position p', then no bricks can be moved to p' by any robot.

$$\begin{cases} \mathbf{x}_{b,p,t} \wedge (\mathbf{x}_{1,p,t} \vee \dots \mathbf{x}_{b-1,p,t} \vee \mathbf{x}_{b+1,p,t} \vee \dots \mathbf{x}_{bs,p,t}) \Rightarrow \neg((\mathbf{y}_{1,b,t} \vee \dots \mathbf{y}_{rs,b,t}) \wedge \mathbf{n}_{p',t}), \\ b \in [1, bs], \quad p \in [1, ps], \quad p' \in [1, ps] \setminus \{p\}, \quad t \in [1, ts]. \end{cases}$$

2. Uniqueness of $from/\mathbf{m}_{p,t}$: if a brick is moved by any robot from position p, then $\mathbf{m}_{p,t}$ must be true, and no more than $rs - 1$ $\mathbf{m}_{p',t}$ among all the rest can be true.

$$\begin{cases} \mathbf{x}_{b,p,t} \wedge (\mathbf{y}_{1,b,t} \vee \dots \mathbf{y}_{rs,b,t}) \Rightarrow \\ \mathbf{m}_{p,t} \wedge \texttt{PbLe}([(\mathbf{m}_{p',t}, 1)\ \texttt{for}\ p'\ \texttt{in}\ \texttt{range}(ps)\ \texttt{if}\ p\ !=\ p'],\ rs - 1)), \\ b \in [1, bs], \quad p \in [1, ps], \quad t \in [1, ts]. \end{cases}$$

3. Uniqueness of $to/\mathbf{n}_{p,t}$: moving a brick to some position p is equivalent to no more than $rs - 1$ $\mathbf{n}_{p',t}$ among all the rest can be true.

$$\begin{cases} \mathbf{n}_{p,t} \Leftrightarrow \texttt{PbLe}([(\mathbf{n}_{p',t}, 1)\ \texttt{for}\ p'\ \texttt{in}\ \texttt{range}(ps)\ \texttt{if}\ p\ !=\ p'],\ rs - 1), \\ p \in [1, ps], \quad t \in [1, ts]. \end{cases}$$

4. Uniqueness of $obj/\mathbf{y}_{r,b,t}$: no more than rs bricks can be moved at each time instance.

$$\begin{cases} \texttt{PbLe}([(\mathbf{y}_{r,b,t}, 1)\ \texttt{for}\ b\ \texttt{in}\ \texttt{range}(bs)\ \texttt{for}\ r\ \texttt{in}\ \texttt{range}(rs)],\ rs), \\ t \in [1, ts]. \end{cases}$$

5. Non-moving bricks: if a brick is not moved by any robot, then it shall stay at the same position for the next time instance and not anywhere else.

$$\begin{cases} \neg(\mathbf{y}_{1,b,t} \vee \dots \mathbf{y}_{rs,b,t}) \wedge \mathbf{x}_{b,p,t} \Rightarrow \\ \mathbf{x}_{b,p,t+1} \wedge \neg\mathbf{x}_{b,1,t+1} \wedge \dots \neg\mathbf{x}_{b,p-1,t+1} \wedge \neg\mathbf{x}_{b,p+1,t+1} \wedge \dots \neg\mathbf{x}_{b,ps,t+1}, \\ b \in [1, bs], \quad p \in [1, ps], \quad t \in [1, ts]. \end{cases}$$

6. Distinct $from(\mathbf{m}_{p,t})$ / $to(\mathbf{n}_{p,t})$: this constraint is not valid on a holistic level since multiple robots can operate at the same time, but for a single robot, this constraint is still valid.

$$\begin{cases} \mathbf{m}_{p,t} \wedge \mathbf{y}_{r,b,t} \Rightarrow \neg(\mathbf{n}_{p,t} \wedge \mathbf{y}_{r,b,t}), \\ b \in [1, bs], \quad p \in [1, ps], \quad r \in [1, rs], \quad t \in [1, ts]. \end{cases}$$

7. Update: if any robot moves a brick to some position p', then it shall stay at p' for the next time instance and not anywhere else.

$$\begin{cases} (\mathbf{y}_{1,b,t} \vee \ldots \mathbf{y}_{rs,b,t}) \wedge \mathbf{m}_{p,t} \wedge \mathbf{n}_{p',t} \Rightarrow \\ \mathbf{x}_{b,p',t+1} \wedge \neg\mathbf{x}_{b,1,t+1} \wedge \ldots \neg\mathbf{x}_{b,p'-1,t+1} \wedge \neg\mathbf{x}_{b,p'+1,t+1} \wedge \ldots \neg\mathbf{x}_{b,ps,t+1}, \\ b \in [1, bs], \quad p \in [1, ps], \quad t \in [1, ts]. \end{cases}$$

8. Initial/final state: exactly the same as single robot case.

$$\begin{cases} \mathbf{x}_{b,p_{\text{b\_initial}},0} \wedge \mathbf{x}_{d,p_{\text{b\_target}},ts}, \\ b \in [1, bs], \quad p \in [1, ps]. \end{cases}$$

9. Counting constraint: since we want to minimize the total time, we shall increase the counter by only one as long as any robot moves.

$$[\texttt{And}(\mathbf{z}_t == \texttt{sum}([\texttt{If}(\texttt{Or}([\mathbf{y}_{r,d,t}\ \texttt{for}\ r\ \texttt{in}\ \texttt{range}(rs)\ \texttt{for}\ d\ \texttt{in}\ \texttt{range}(ds)]), 1, 0)])\ \texttt{for}\ t\ \texttt{in}\ \texttt{range}(ts))]$$

10. Objective: exactly the same as single robot case.

$$\textbf{MIN}\left\{\sum_{t=0}^{ts} \mathbf{z}_t\right\}$$

2.3 - New cost function

Let's first denote a position on a square platform with size $n \times n$ as $p_{[i,j]},\ i, j \in [1, n]$, where i and j represent $x-$ and $y-$ coordinates respectively. Afterwards, the counting constraint can be modified such that if any robot moves a brick, the cost will be the sum of Cartesian and Manhattan coordinate distance between current position and target position instead of 1. And at each time instance, we sum up the costs of every robot. The formula can be interpreted as below.

$$\texttt{And}([\mathbf{z}_t == \texttt{sum}([\texttt{If}(\texttt{And}(\mathbf{y}_{r,b,t}, \mathbf{m}_{p_{[i,j]},t}, \mathbf{n}_{p_{[i',j']},t}), cost, 0)\ \forall\ r \in [1, rs]\ \forall\ b \in [1, bs], \forall\ i, j \in [1, n]])\ \forall\ t \in [1, ts]])$$

where $cost = \sqrt{(i'-i)^2 + (j'-j)^2} + |i'-i| + |j'-j|$, i and j are coordinates of current time instance t, and i' and j' are coordinates of the next time instance $t+1$.

As for the objective function, it's still the same as following.

$$\textbf{MIN}\left\{\sum_{t=0}^{ts} \mathbf{z}_t\right\}$$