First & last name	Cyril Bousmar
NOMA UCLouvain	63401800

LINFO2266: Advanced Algorithms for Optimization

Project 6: MDD

Exercise 1 Implementing the Solver

The first task of this assignment is for you to implement the vanilla algorithm as it has been explained during the lecture (see slide 67). To help you, we provide you with a well-documented framework that defines and implements all the abstractions you will need to implement a generic solver. Additionally, and because the BaB-MDD framework parallelizes *VERY* well, we provide you with a parallel implementation of the algorithm (ParallelSolver). Digging into that code, understanding it, and stripping away all the parallel-related concerns should finalize to give you a thorough understanding of the sequential algorithm. Before diving into the code, answer the following few questions to make sure you get a good grasp of the basics behind BaB-MDD algorithm.

Exercise 2 Definitions: Exact, Relaxed, and Restricted MDD

1. Give the formal definition of an *exact MDD* encoding the problem $\mathcal{P}$

   An exact MDD is a layered directed acyclic graph that encodes all feasible solutions to a problem, with each root-to-terminal path representing a valid solution.

2. Give the formal definition of a *restricted MDD* for the problem $\mathcal{P}$

   A restricted MDD is an approximation of the exact MDD, limited to a fixed width W (pruning intermediate states), representing a subset of feasible solutions.

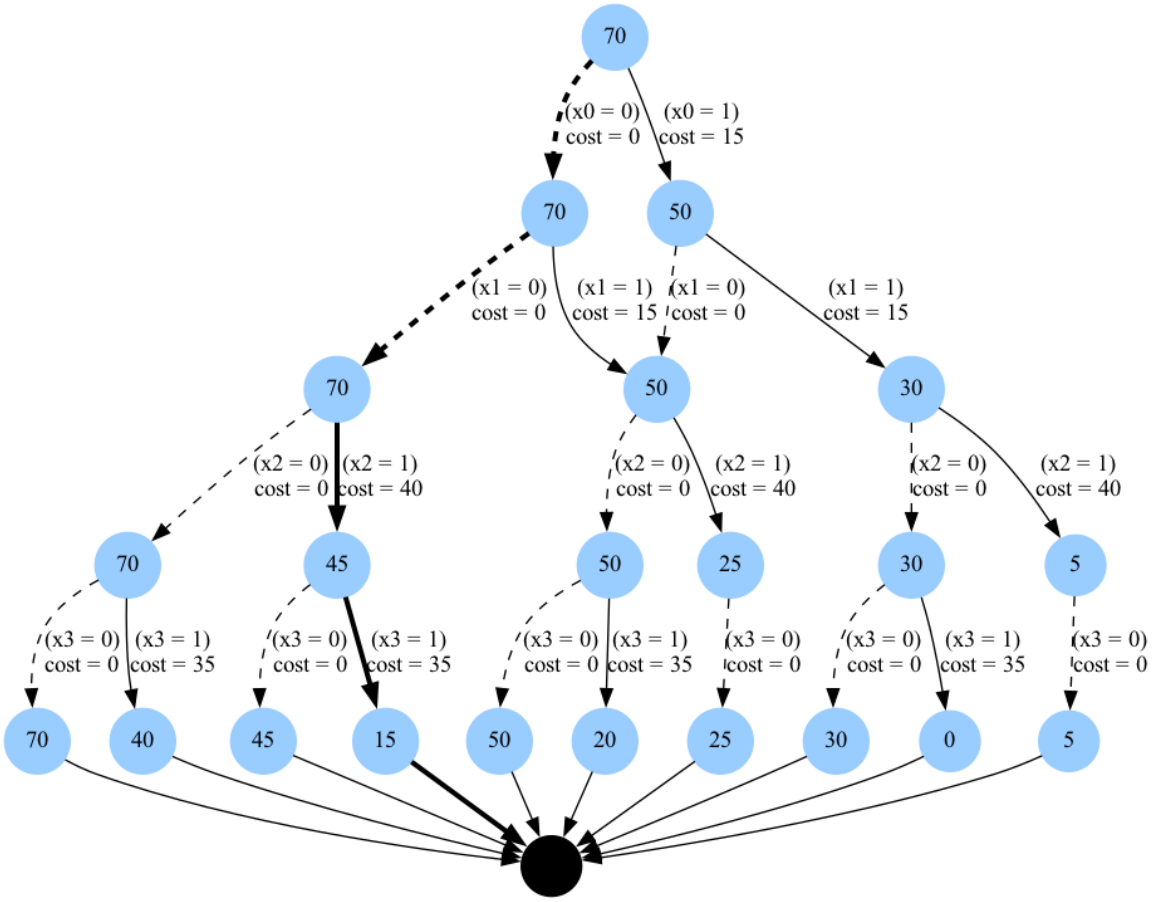
3. Give the formal definition of a *relaxed MDD* for the problem $\mathcal{P}$

   A relaxed MDD is a compact over-approximation of the exact MDD, merging nodes to fit within a width W, possibly including infeasible solutions.

Exercise 3 Application to Knapsack

Consider the following figure: it depicts an exact DD encoding all solutions to a knapsack problem instance (it uses the same model as in the slides). In this instance, the sack has a capacity of 70, and the costs-and-profits of each item are given as follows:

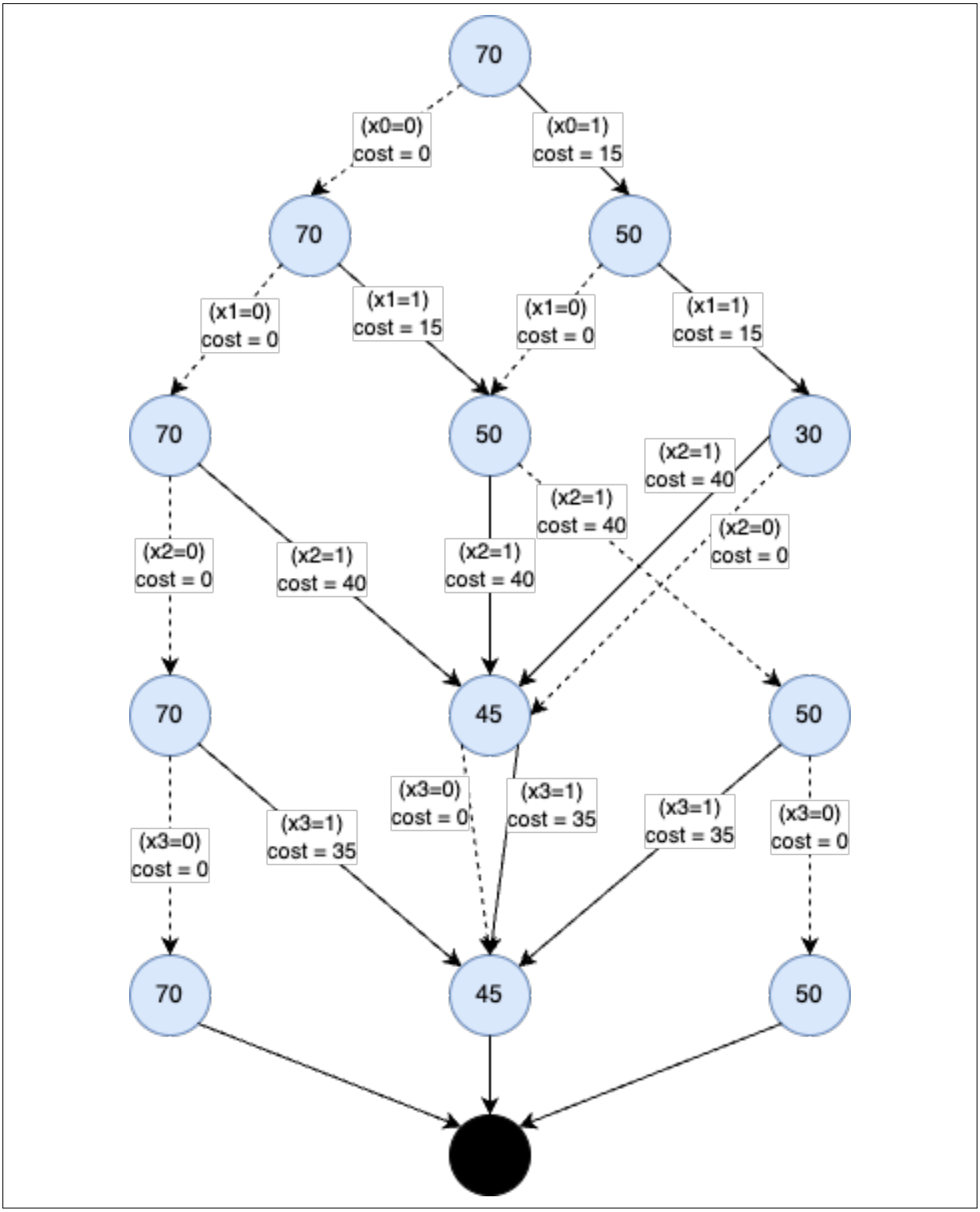
	x_0	x_1	x_2	x_3
weight (w_i)	20	20	25	30
profit (p_i)	15	15	40	35



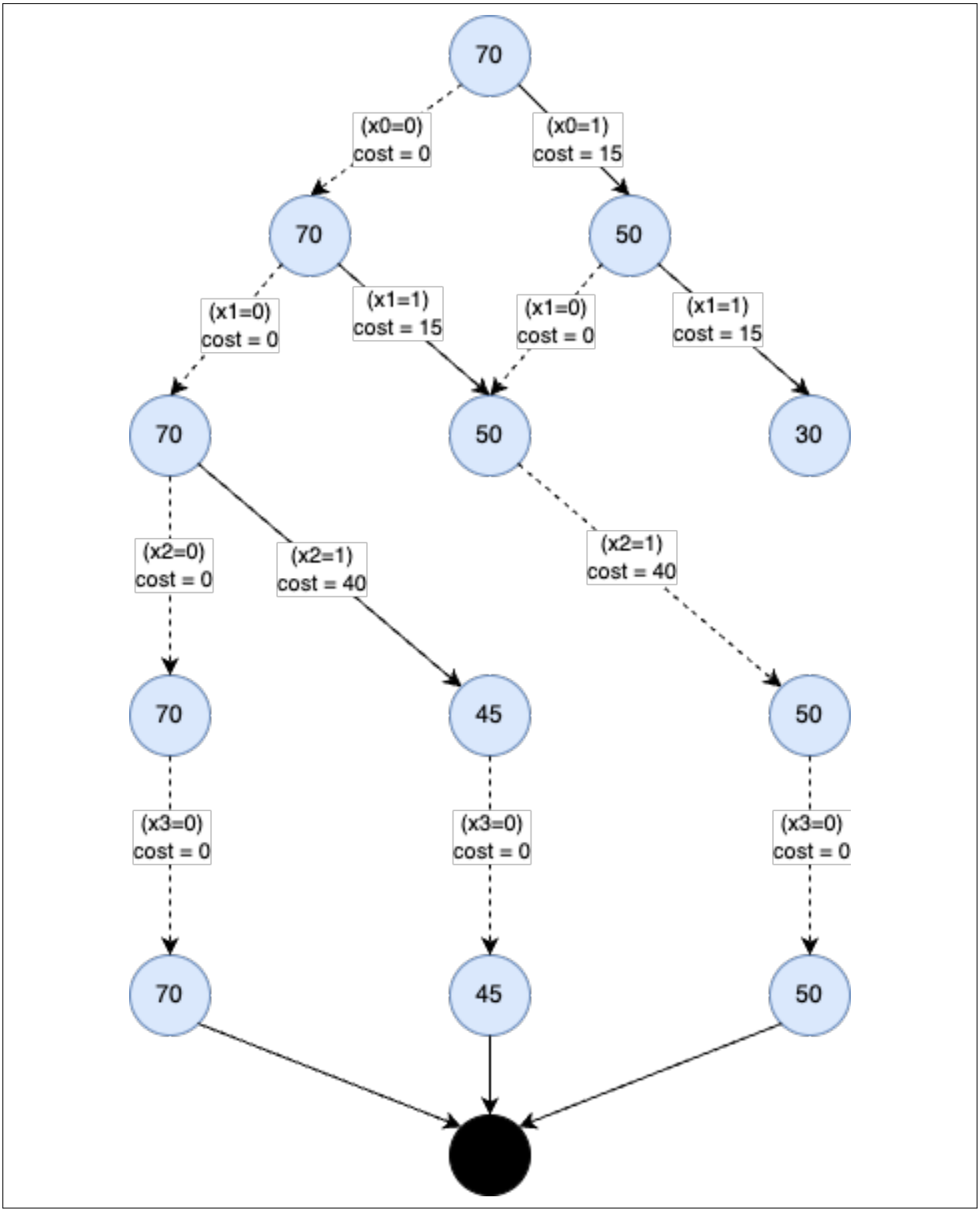
1. What is the best solution encoded in this DD, and what is its objective value ?

Best solution: $x_0 = 0$, $x_1 = 0$, $x_2 = 1$, $x_3 = 1$.
Objective value: 75

2. Draw the relaxed DD corresponding to the example instance with $W = 3$ (Your selection heuristic must decide to merge the nodes with lowest remaining capacity).



3. Draw the restricted DD corresponding to the example instance with $W = 3$ (Your selection heuristic must decide to delete the nodes with lowest remaining capacity).



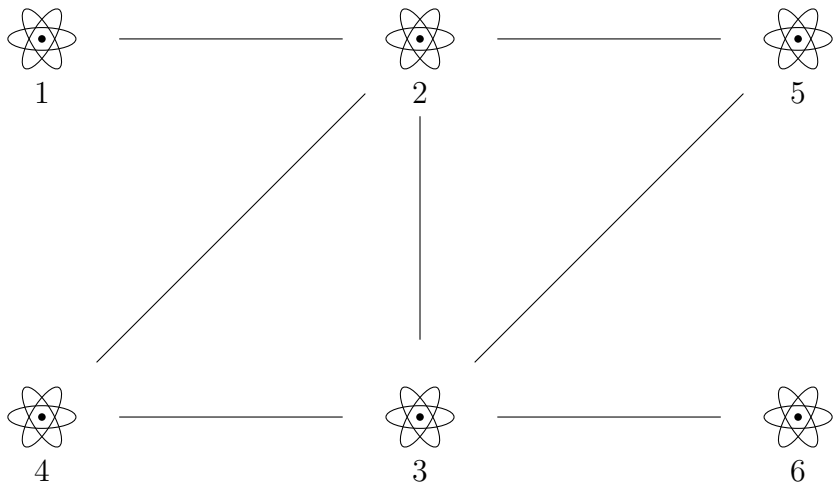
Exercise 4 Building Nuclear Plants

To decarbonate the European economy, Mrs. From-the-Streets, minister of energy, has decided to refresh electric infrastructure and build new nuclear plants all over the continent. This, however, has to be done in a principled way and no two power plants can be located too close to one another as there is a risk it would cause excess tension. Given a map of the infrastructure with candidateOld build sites, you are asked to find the set of locations where to construct power plants without violating safety constraints.

In practice, the map will be given to you in the form of a sizeable *undirected* graph. In that graph, each node represents a candidateOld build site, and two sites are connected with an edge if they are too close to one another (or more generally, if building a power plant at both sites causes a potential safety hazard). The problem which you are asked to solve is thus the following: you must find a maximum subset of the nodes in the graph such that no two construction sites are connected with an edge in the underlying graph.

Example:

The below graph depicts a situation where 6 candidateOld construction sites have been identified. It also shows that it would not be safe to build a nuclear plant at site 1 and 2 at the same time, or to build some at both sites 3 and 5. In this example, a solution that maximizes the energy production while guaranteeing safety constraints is the set $\{1, 4, 5, 6\}$.



1. What information is contained in the states of the model?

A state is a valid representation of the problem: $S = \{S_0, S_1, ..., S_n\}$. It contains already selected sites in a BitSet with respect of the constraints (initial is empty). And therefore, implicitly contains information on all not selected sites as well.

It also contains the cost of the path to reach that state, which represent the total amount of production (initial is zero).

2. Give the transition function of your problem

Transition function, $\tau_i : S_i \times D_i \to S_{i+1}$, for $s_i \in S_i$ and $d_i \in D_i$.

If $x_i = 1$ and contraints are satified, add site i to the state, move to the next state withtout including i otherwise.

3. Give the transition cost function of your problem.

Transition cost function is defined as $h_i : S_i \times D_i \to \mathbb{R}$, where

$$h_i(s^i, x_i) = \begin{cases} \text{energy}(i), & \text{if } x_i = 1 \text{ and constraints are satisfied,} \\ 0, \text{otherwise.} \end{cases}$$

4. Give the definition of the relaxation operator $\oplus$ which you will be using.

$$\Theta(S_1, S_2) = \{u \cup v : u \in S_1, v \in S_2, \text{ and no safety constraints are violated}\}$$

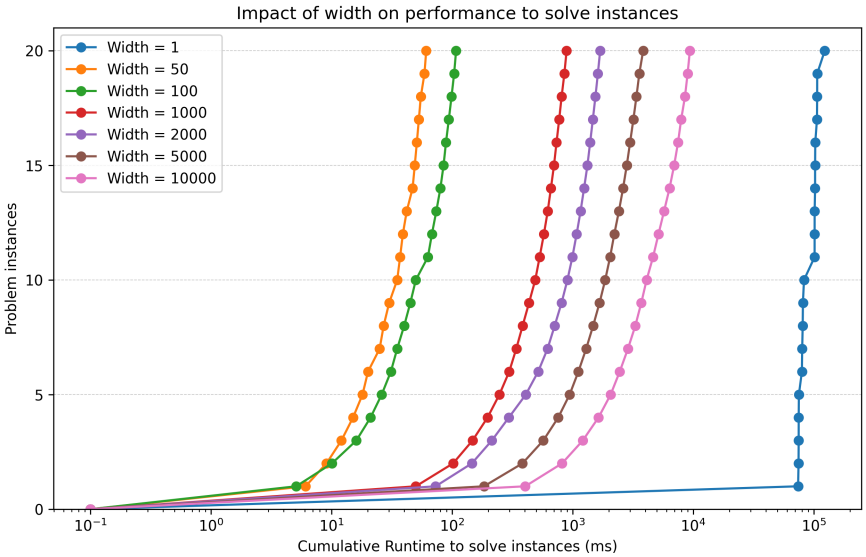
Exercise 5 Experimental comparison of widths

We are interested to compare experimentally the time required by the MDD solver to solve the instances, width different widths provided for the MDD. We can perform time measurement on benchmark instances and report them graphically. Given the 20 instances `decarbonation_100_XXX.dimacs` within the `data/decarbonation/instances` directory, analyze the running time of the solver with the widths:

- 1
- 50
- 100
- 1000
- 2000
- 5000
- 10 000

You need to use the **sequential implementation of your solver** when running the experiments.

1. Give the cactus plot of the running time to solve the 20 instances for each of the 7 widths provided. Your x axis should be the maximum time allowed (possibly in log scale, it's up to you to choose) and the y axis the cumulated number of problems solved within the maximum time (i.e. your curves for each width need to reach y=20 when all instances are solved, and the y value for each curve is incremented each time a new instance has been solved). You can find one example of such a plot here.



2. What is the best choice of width in your opinion?

In terms of speed, 50 is the best, but it outputs underperforming results. 100 is nearly as fast and produces optimal solution most of the time. 1000 is the fastest that gives only optimal solutions.
So, there is a trade of between 100 and 1000, depending on the priority: speed, or reliability. 1000 is 10 times slower than 100 width.