UNIVERSITÉ DU LUXEMBOURG

Parallel Interval Propagation

Parallel Computing

Goals

- ✯ Study a problem difficult to parallelize efficiently.
- ✯ **Relevant videos**: All videos from the task scheduler laboratory might be useful. The following ones cover more advanced concepts on parallelism.
  - Sequential consistency
  - Litmus tests
  - come back later for more...

Deliverables

1. The code on your Github repository generated by clicking here: `https://classroom.github.com/a/av4hHNIk`
2. Presentation of the results at the end of semester.

Rules and Information

1. You can discuss your design and your results on Discord or orally, but please don't share your code.
2. This is a solo or team project (between 1 and 3 students).
3. All members of the team must have unlocked this project before presenting.
4. **Automated leaderboard**

Exercise 1 – Parallel Interval Propagation

We study the problem of interval propagation. It is central in many fields including constraint reasoning, preprocessing of linear programming models, neural network verification and numerical analysis. However, it is rather difficult to parallelize efficiently.

The problem is defined as follows. Let X be a finite set of variables. Let C be a finite set of constraints of the form $x = y + z$ where $x, y, z \in X$. The domain of the variables is a function $d : X \to I$ where I is the set of intervals $\{[\ell, u] \mid \ell \in \mathbb{Z}, u \in \mathbb{Z}, \ell \leq u\}$. The goal is to reduce the interval of each variable as much as possible.

Example. Let $X = \{x_0, x_1, x_2, x_3\}$ and C be the following constraints:

$$x_0 = x_1 + x_2$$
$$x_1 = x_2 + x_3$$

Let the domain of each variable be $d(x_0) = [0, 1]$, $d(x_1) = [0, 2]$, $d(x_2) = [0, 3]$ and $d(x_3) = [0, 3]$. A solution to this system of equations is $x_0 = 0, x_1 = 0, x_2 = 0, x_3 = 0$. Another one is $x_0 = 1, x_1 = 1, x_2 = 0, x_3 = 1$. However, there is no solution where $x_1 = 2$ or $x_2 \geq 2$. Therefore, we can reduce the domains of x_1 and x_2 to $x_1 = [0, 1]$ and $x_2 = [0, 1]$.

In the following, we propose a simple *reduce* function which shrinks the domains of the variable for constraint of the form $x = y + z$. During the execution of our reduction algorithm, the domain of the variable should always become smaller and smaller. That is, the interval of each variable is only getting smaller. To ensure that, we update the domain of a variable by performing an interval intersection defined as follows:

$$[\ell, u] \cap [\ell', u'] = [\max(\ell, \ell'), \min(u, u')]$$

For instance, $[0, 10] \cap [5, 15] = [5, 10]$.

Domain reduction. The domains of the variables can be reduced using the following rules for each constraint $x = y + z$ ($x, y, z \in X$):

$$\begin{aligned} &reduce(d, x, y, z) = \\ &\quad d(x) = d(x) \cap (d(y) + d(z)) \\ &\quad d(y) = d(y) \cap (d(x) - d(z)) \\ &\quad d(z) = d(z) \cap (d(x) - d(y)) \end{aligned}$$

where $+$ and $-$ are operations defined using interval arithmetic as follows:

$$[\ell, u] + [\ell', u'] = [\ell + \ell', u + u']$$

$$[\ell, u] - [\ell', u'] = [\ell - u', u - \ell']$$

For the reduction algorithm presented next, we need to extend slightly this *reduce* function to know if a domain has changed or not.

$$\begin{aligned} &reduce(d, x, y, z) = \\ &\quad dx = d(x) \\ &\quad dy = d(y) \\ &\quad dz = d(z) \\ &\quad d(x) = d(x) \cap (d(y) + d(z)) \\ &\quad d(y) = d(y) \cap (d(x) - d(z)) \\ &\quad d(z) = d(z) \cap (d(x) - d(y)) \\ &\quad \mathbf{return}\ (dx \neq d(x) \vee dy \neq d(y) \vee dz \neq d(z)) \end{aligned}$$

Sequential reduction algorithm. A simple algorithm to reduce all the constraints is to reduce them one by one until nothing is changing anymore. This idea is captured in the following algorithm. Note that the domain function can be efficiently implemented using an array of intervals (each variable is an index in the array).

```
function reduction(d, C)
    b = true                          Indicate whether a domain changed.
    while b do                        We continue as long as at least one domain has changed.
        b = false
        for (x = y + z) ∈ C do        We apply the reduction to each constraint.
            if reduce(d, x, y, z) then   Check if the domain has changed or not.
                b = true
            end if
        end for
    end while
end function
```

Goal of the laboratory. The goal is to compute this reduction, and to avoid printing all the domains, we ask you to print the sum of the intervals' width, i.e., after applying $reduction(d, C)$, print $\sum_{x \in X} |d(x)|$, where $|[\ell, u]| = (u - \ell) + 1$.

Input Format. We specify the input format of the problem using the example above:

```
4
0 1
0 2
0 3
0 3
2
0 1 2
1 2 3
```

The first line indicates the number of variables N. The N next lines are the domains of the variables (here we have $d(x_0) = [0, 1]$). The next line is the number of constraints M. The M next lines are the constraints where `0 1 2` represents the indexes of the variables in the constraint $x_0 = x_1 + x_2$.

Output Format. You must output the sum of the width of the reduced domains of the variables as follows:

```
8
```

Dataset We provide instances of growing complexity to test the efficiency of your code here `https://uniluxembourg-sharepoint.com/:u:/g/personal/pierre_talbot_uni_lu/IQCntKC2OvRSTa98_llca1TDAeRnH1Euovpe=zBnC9p`

Command line. Provide the following options:

- `--input file.txt`: a file with the input specified above.
- `--variant [seq|par]`: the variant of the algorithm. At least, you provide a sequential version and your best parallel version. You can add different variants for your own testing, and document it in the README.