FIT5216: Modelling Discrete Optimization Problems
Assignment 1: Animal Capture

1 Overview

For this assignment, your task is to write a MiniZinc model for a given problem specification.

- Submit your work to the MiniZinc auto grading system (using the submit button in the MiniZinc IDE).

You have to submit by the due date (Friday 22nd March 2024, 11:55pm), using MiniZinc to receive full marks. You can submit as often as you want before the due date. Late submissions without special consideration receive a penalty of 10% of the available marks per day. Submissions are not accepted more than 7 days after the original deadline.

This is an **individual assignment**. Your submission has to be **entirely your own work**. We will use similarity detection software to detect any attempt at collusion, and the **penalties are quite harsh**. Note that we will compare all your saved models against others. You may not use **large language models** such as ChatGPT for any part of this assignment. If in doubt, contact your teaching team with any questions!

Learning outcomes from this assessment include:

- model a discrete optimisation problem using a mix of basic and more advanced modelling techniques in a high level modelling language;
- identify and fix errors in models;

2 Problem Statement

You are charged with setting up an animal monitoring program in a forested region. You need to set up a wireless network of camera traps to detect as much of the wildlife as possible given your budget restrictions.

Input data is given in MiniZinc data format:

```
LOC = ⟨ the set of locations where you can place traps and the base ⟩;
base = ⟨ the base location where you collect information ⟩;
n = ⟨ The number of camera traps available to use ⟩;
wild = ⟨ Wildlife density at each location ⟩;
cost = ⟨ cost to setup a trap at each location ⟩;
d = ⟨ distance matrix from each location to another ⟩;
move = ⟨ animal movement distance ⟩;
link = ⟨ wireless link distance ⟩;
mind = ⟨ minimum distance between two traps ⟩;
opcost = ⟨ operating cost for each trap ⟩;
budget = ⟨ budget for setting up system ⟩;
```

Note that the base location is always the first in `LOC`. If the cost to setup a trap at a location is negative then we are not able to set up a trap there.

Here is a sample data set:

```
LOC = { BASE, A, B, C, D, E, F, G, H };
base = BASE;
n = 3;
wild = [ 0, 10, 7, 3, 2, 8, 6, 4, 9 ];
cost = [ 0,  6,  4,  5, -1,  3,  2,  2,  4 ];
d =   [| 0,  4,  8, 12, 16, 18, 19, 14,  5
       | 4,  0,  5,  9, 12, 17, 20,  7,  9
       | 8,  5,  0,  5,  8, 12, 14, 15, 12
       |12,  9,  5,  0,  3,  6,  8, 10, 11
       |16, 12,  8,  3,  0,  9,  2,  6,  8
       |18, 17, 12,  6,  9,  0,  5,  8, 15
       |19, 20, 14,  8,  2,  5,  0,  8, 12
       |14,  7, 15, 10,  6,  8,  8,  0,  9
       | 5,  9, 12, 11,  8, 15, 12,  9,  0 |];
move = 7;
link = 6;
mind = 3;
opcost = 8;
budget = 26;
```

There are 9 locations, the first location is the BASE of operations, where no camera traps can be placed. There are three camera traps available for use. Each location has a wildlife density and cost to set up a trap there. Note that since the cost for D is -1 we are not able to set up a trap there. The distance matrix is symmetric, and has 0s on the diagonal (the distance to a location from itself is always 0). Animals can move up to distance 7, while the wireless link has range 6. Each pair of traps must be placed at least 3 distance apart. Operating each trap costs 8, and a total budget for operating and setting up the system is 26.

There are two decisions to be made

```
array[0..n] of var LOC: x;  % where traps are placed, but x[0] = base
array[1..n] of var 0..n: s; % send location (only used in part C)
```

The aim is to cover the most possible wildlife. A location is "covered" if there is a trap at a location at most `move` from this location.

Part A - Using all the traps

Create a model `animal.mzn` that takes data in the format specified above and decides on exactly `n` different camera trap locations. For the moment we ignore the budget constraint.

So the aim is to select n different locations in `x[1..n]`. The 0th location must be set to `base` and no other location set to `base`. For part A and part B, just set `s[i] = 0` for all `i`.

Remember you can use the expression `d[`u`,`v`]` to find the distance between two locations, *even if* the locations u and v are decisions. You will need to decide which locations are covered, and you may want to build an auxilliary decision variable to store this information, or to count for each locations how many traps cover it.

Here is a sample solution.

```
x = [0: BASE, 1: H, 2: C, 3: A];
s = [0, 0, 0];
total_wild = 43;
```

We elected to place traps at locations $\{A, C, H\}$. The total wildlife that is covered by this setup is 43, being the wildlife at locations $\{A, B, C, E, F, G, H\}$ (which are within 7 of one of the traps). Note that no two traps are less than distance 3 apart, and no traps are set up at locations with negative cost.

Note that you will not be able to obtain many marks by just answering part A. Some problems will have no solution, whereas using part B they have a solution.

Part B - Possibly using less traps

Modify your model `animal.mzn` to treat `n` as a bound on the maximal possible number of equipment. We will use the `base` location as a dummy value. So if `x[i] = base` then this indicates no trap placed. We must force all the dummy locations to be at the end of the `x` array (except that `x[0] = base` always).

Now you must take into account the budget constraint: that is the total operating cost of traps installed plus the install cost must be no more than the budget.

Note that you should endeavour to only have one way of representing each possible set of installed traps. This will usually make the model more efficient.

Here is a sample solution for part B.

```
x = [0: BASE, 1: B, 2: F, 3: BASE];
s = [0, 0, 0];
total_wild = 36;
```

Now we only place traps at locations $\{B, F\}$. The final entry in the `x` array indicates we do not place a third trap. The total wildlife covered is 36 being the wildlife at locations $\{A, B, C, D, E, F\}$ (which are within 7 of one of the traps). The two traps are 14 apart, well outside the minimum distance. The total budget used is 16 in operating cost (running two cameras) plus $4 + 2 = 6$ setup costs, fitting within the budget of 26. Note that the total cost for the previous solution $\{A, C, H\}$ is $3 \times 8 + 6 + 5 + 4 = 39$ over the given budget.

Note that you will not be able to obtain full marks by just answering parts A and B, but you can get a good mark. For full marks you need to correctly complete part C but it is designed to be challenging.

Part C - Connecting the network

The camera traps have to send the photos to the base for the system to work. To do this each trap must send its information to the base directly, or to another trap which then sends on the information further. To represent this network, we use `s[i]` to refer to the place (from 0 to n) where the camera at the i^{th} place sends its information. Note that sending to place 0 represents sending to the base (`x[0] = base`). To ensure that the network is a tree we require that the place where location i sends its info is a place less than i. Note that we require the distance between the location sending and receiving information is no more than `link`.

For dummy locations i where `x[i] = base` we should set the send place to 0, but there is no distance constraint, since we are not actually setting up a camera.

A solution for part C is given by

```
x = [0: BASE, 1: A, 2: B, 3: BASE];
s = [0, 1, 0];
total_wild = 24;
```

Again we only use two camera traps at $\{A, B\}$. The trap at A sends its info to location 0, the base, at distance 4; while the trap at B sends its info to location 1, A, at distance 5 (which will then be sent on to the base by A); hence the link constraints are satisfied. Note that the previous solution $\{B, F\}$ is no longer valid since F is at distance 19 from $BASE$ and 14 from B, so no send link is available. The total wildlife covered is 24 consisting of $\{A, B, C, G\}$. The budget constraints is satisfied with cost $2 \times 8 + 6 + 4 = 26$.

3 Instructions

Edit the provided `mzn` model files to solve the problems described above. You are provided with some sample data files to try your model on. Your implementations can be tested locally by using the *Run+check* icon in the MiniZinc IDE. Note that the checker for this assignment will only test whether your model produces output in the required format, it does not check whether your solutions are correct. The grader on the server will give you feedback on the correctness of your submitted solutions and models.

4 Marking

The marks are automatically calculated. With only Part A you can get full marks for a few instances, most will get 0. With Part A and part B you can get full marks for many instances, and otherwise a max of 0.75. The autograder will grade instances as: 0.25 for any solution, 0.5 for a reasonable solution, 0.75 for a good solution, and full marks for the optimal solution. Because part C adds constraints which can removes solutions, part B solutions that ignore part C may give superoptimal answers (violating the C constraints), these will get a maximum of 0.75 marks. To get maximum marks your model must be efficient as well as correct. Ways to improve efficiency are:

- Make sure there is only one (or at least as few as possible) ways of representing the same solution (set of traps placed).
- Express the constraints you need in the simplest possible form

The submission has 10 marks for locally tested data and 10 for model testing, for a total of 20 marks. For model testing you will only get feedback of marks for each test, you will not be able to see the test data. Concentrate on getting the locally tested data working first, since this is easier to debug.