CPSC 322, Practice Exercise
SLS for CSP

1 Directed Questions

- In local search, how do we determine *neighbours*?
- What is the difference between random walk and random restart?
- What is the key weakness of stochastic local search?

2 Traffic Flow

Consider the following scenario. You are on a city planning committee and must decide how to control the flow of traffic in a particular residential neighbourhood. At each intersection, you have to decide whether to install a 4-way stop, a roundabout, or an uncontrolled intersection (in an uncontrolled intersection, the streets intersect without signs, roundabouts or other traffic conrols). There are several restrictions in how the intersections can be controlled. Figure 1 gives an overview of the neighbourhood.

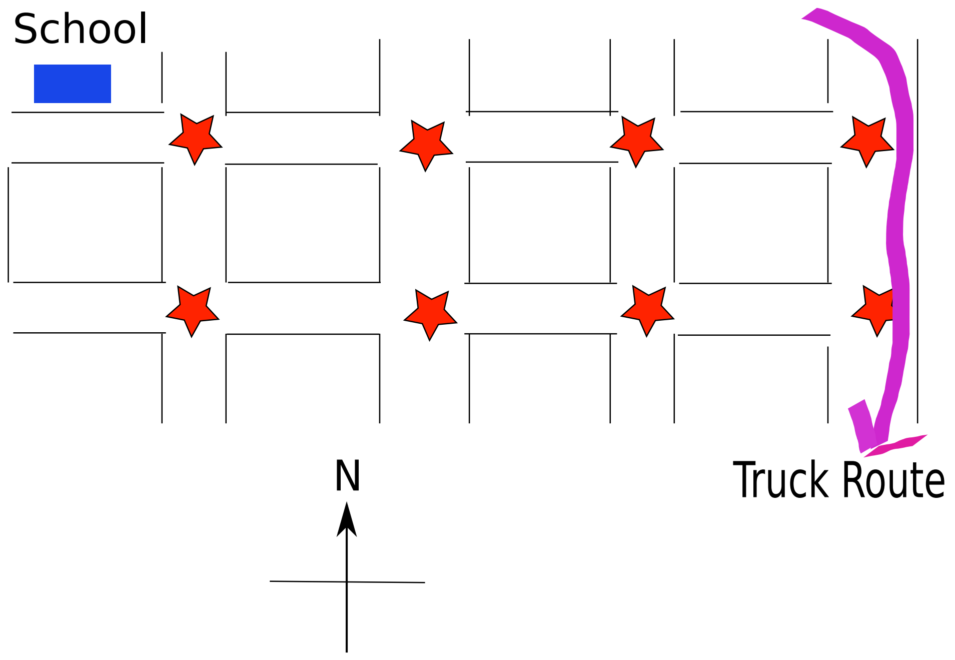

Figure 1: Neighbourhood Traffic Control

The red stars indicate the 8 intersections under consideration. The intersection nearest the school in the NW corner of the neighbourhood must contain a 4-way stop for safety reasons.

Along the east side of the neighbourhood runs a truck route, and no roundabouts can be placed on this street because they pose a problem for large trucks. Also, it is not allowed to have 4-way stops at consecutive intersections or to have two consecutive uncontrolled intersections. Finally, due to the cost of installing roundabouts compared with the other options, each block can have *at most* one of its four corners with a roundabout.

2.1 CSP Representation

How would you represent the above problem as a CSP? Identify the variables, their domains, and the constraints involved.

Once you are done a sketch on paper, navigate to the AISpace Local Search applet at `http://www.aispace.org/hill/`, start the applet, and load the file `roundabouts.xml` (available from the course website) by clicking File → Load from file. This shows one possible representation, but there might be more than one correct representation.

2.2 Comparing Local Search Algorithms

Using the AISpace Local Search applet, we will experiment with several local search algorithms for solving this problem.

- Greedy Descent

  From the menu, choose Hill Options → Algorithm Options and then select Greedy Descent from the dropdown menu. Click Ok. Click Initialize. This will assign a value to each variable. Note: you can choose Hill Options → AutoSolve Speed → Very Fast to speed up the solver. Click AutoSolve. What happens? Does it find a solution within 100 steps? Hypothesize why or why not. Now click Batch Run, which will calculate the runtime distribution and plot the percentage of successes against the number of steps. What does the runtime distribution tell you about this solver?

- Greedy Descent with Random Restarts

  Go back to Algorithm Options and now select Greedy Descent with Random Restarts. Click Batch Run again and compare the runtime distributions. Do the random restarts improve Greedy Descent? Why or why not?

- Random Walk

  Go back to Algorithm Options and now select Random Walk. Click Batch Run again. How does this compare with plain Greedy Descent? How would the two algorithms compare if you gave them 10000 steps? (a logarithmic scale might help the visualization)

3 Local Search Learning Goals

- Implement local search for a CSP
- Implement different ways to generate neighbours
- Implement scoring functions to solve a CSP by local search through either greedy descent or hill-climbing

- Implement SLS with random steps and random restarts
- Compare SLS algorithms with runtime distributions