**Sistemas Inteligentes**

**2017-II**

**Luis Fernando Niño V.**

**Ejercicios. Problemas de búsqueda y de satisfacción de restricciones (*Constraint satisfaction problems, CSP*)**

**Problem 1**. Joy Taxi has four taxis (1,2,3 and 4), and there are four customers (P, Q, R and S) requiring taxis. The distance between the taxis and the customers are given in the table below, in kilometers. The Taxi company wishes to assign the taxis to customers so that the distance traveled is minimum.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  |  | **Customers** |  |  |
|  |  | **P** | **Q** | **R** | **S** |
|  | **1** | 10 | 8 | 4 | 6 |
| **Taxis** | **2** | 6 | 4 | 12 | 8 |
|  | **3** | 14 | 10 | 8 | 2 |
|  | **4** | 4 | 14 | 10 | 9 |

1. Formulate this problem as a search problem.
2. Define an admissible heuristic for this search problem.
3. Compute the optimum value using the A\* algorithm.
4. Formulate this problem as a Constraint Satisfaction Problem (CSP).
5. Solve this CSP using backtracking search.

**Problem 2 (Exrercise 6.7 AIMA).** Consider the following logic puzzle:

In five houses, each with a different color, live 5 persons of different nationalities, each of whom prefer a different brand of cigarette, a different drink, and a different pet.

Given the following facts, the question to answer is “Where does the zebra live, and in which house do they drink water?”

The Englishman lives in the red house.

The Spaniard owns the dog.

The Norwegian lives in the first house on the left.

Kools are smoked in the yellow house.

The man who smokes Chesterfields lives in the house next to the man with the fox.

The Norwegian lives next to the blue house.

The Winston smoker owns snails.

The Lucky Strike smoker drinks orange juice.

The Ukrainian drinks tea.

The Japanese smokes Parliaments.

Kools are smoked in the house next to the house where the horse is kept.

Coffee is drunk in the green house.

The Green house is immediately to the right (your right) of the ivory house.

Milk is drunk in the middle house.

Discuss different representations of this problem as a CSP. Why would one prefer one representation over the other?

**Problem 3.** Consider a scheduling problem, where there are five activities to be scheduled in four time slots. Suppose we represent the activities by the variables A, B, C, D, and E, where the domain of each variable is {1,2,3,4} and the constraints are

A>D, D>E, C ≠A, C>E, C ≠D, B ≥ A, B≠C, and C≠D+1.

[Before you start this, try to find the legal schedule(s) using your own intutions.]

1. Show how backtracking can be used to solve this problem. To do this, you should draw the search tree generated to find all answers. Indicate clearly the valid schedule(s). Make sure you choose a reasonable variable ordering.

To indicate the search tree, write it in text form with each branch on one line. For example, suppose we had variables X, Y, and Z with domains t, f and constraints X ≠Y and Y≠Z. The corresponding search tree can be written as:

X=t Y=t failure

Y=f Z=t solution

Z=f failure

X=f Y=t Z=t failure

Z=f solution

Y=f failure

[Hint: It may be easier to write a program to generate such a tree for a particular problem than to do it by hand.]

1. Show how arc consistency can be used to solve this problem. To do this you must draw the constraint graph; show which elements of a domain are deleted at each step, and which arc is responsible for removing the element;
2. Show explicitly the constraint graph after arc consistency has stopped; and
3. Show how splitting a domain can be used to sove this problem.

**Problem 4:** Consider the following scheduling problem. We need to execute n tasks (or processes) using m processors under certain constraints. For simplicity, we will use the following notation:

* **Tasks**: The n tasks will be denoted by t1, t2, t3, ..., tn.
  + The "length" of each task is given. The length corresponds to the number of seconds that takes a processor to run the task. The length of task *i* will be denoted by *li*.
* **Processors**: The *m* processors will be denoted by p1, p2, p3, ..., pm. Each of the processors runs the tasks assigned to it in a sequencial manner. Processors are independent from each other.
* **Constraints**: The constraints are of the following sort:
  + **Deadline**: A deadline *d*, given in seconds, is specified by the user. All tasks must be executed before the deadline. That is, each of the processors must be able to sequentially execute all tasks assigned to it before the deadline. All processors start executing the processes assigned to them at time t=0.
  + **Unary constraints** (involving one variable):
    - A task can only be assigned to one of a subgroup of processors. For instance, the user may specify that task t10 can be run only on one of the following processors: p2, p7, p10.
    - A task cannot be assigned to any in a given group of procesors. Example: task t9 cannot be assigned to neither p3 nor p2
  + **Binary constraints** (involving two variables):
    - Two given tasks need to be run on the same processor
    - Two given tasks cannot be assigned to the same processor
    - Two given tasks cannot be simultaneously assigned to a given pair of processors. For instance, the user can specify that task t1 cannot be assigned to processor p7 if task t4 is assigned to processor t3.

Design and implement a constraint satisfaction system that produces an assignment of processors to tasks which satisfies all constraints (including the deadline constraint), if such an assignment exists. Your system must satisfy the following requirements:

1. The input to the system consists of:
   * n: the number of tasks,
   * m: the number of processors,
   * li: the length (in seconds) of task i, for each task.
   * d: the deadline (in seconds).
   * the unary and binary constraints.
2. The internal representation of the problem should use a constraint graph.   
   Note that the variables in this problem are the tasks t1, ..., tn and the domain of each of the variables is the set of all processors {p1,...,pm}.
3. Binary constraints should be represented using constraint matrices. Each constraint matrix is an m-by-m matrix.
4. The problem solving strategy should be an implementation of the ones discussed in class:
   * Arc consistency checking,
   * Depth-first search with backtracking, and
   * a sound heuristic to select the first variable and the order in which other variables are expanded.
5. The output of the system should be either:
   * An assignment of processors to tasks such that all constraints are satisfied (including the deadline constraint). OR
   * A message stating that NO such assignment is possible, IF that's the case.

You must write your own code in any high level programming language.