CONSTRAINT SATISFACTION – EXERCISES 2002-03

[Answer 3 out of 4 questions]

Marking principle:
Solutions are presented in this document in outlines only. Students are expected to elaborate certain points appropriately. Marks will be awarded to any correct answers and valid points that may be different from the answers presented here. Marks will be deducted if the student shows confusion in the answers (e.g. if the answer asserts both P and not P for some proposition P).

Question 1 (Problem Modeling, Constraint Satisfaction):

The Warehouse Location Problem:

A supermarket chain has to decide which of the available locations to build its warehouses in order to serve its *m* stores. The decision maker is given the following information:

i. **Locations L_1, L_2, …, L_n where warehouses can be built;**
ii. **No more than *k* warehouses can be built, where *k* is a constant and *k*<n;**
iii. **Each location *i* has a capacity constraint CP_i , indicating the number of stores that it can serve (all stores have the same capacity requirement);**
iv. **Each store must be served by one warehouse;**
v. **The cost of supplying store S_i from location L_j is $SC_{i,j}$;**
vi. **The total cost must be less than a constant TC.**

(a) **Briefly explain why problem formulation, or *modeling*, is important. [10%]**

(b) **In the Warehouse Location Problem, suppose we have decided to use Boolean variables W_1, W_2, …, W_n to represent whether each of the *n* locations should be used; if W_j = *True* then a warehouse is to be built on location j. Complete the problem formulation by explaining how you would define the constraints. You may need to introduce extra variables. If you do, please justify your decisions clearly. You may express your constraints either mathematically or in words, but they must be expressed in terms of relationships between the variables in the problem. You should explain how each of the points above is captured by your formulation (please answer in point form). [90%]**

Answer 1:

This question tests the students' ability in problem solving using constraint techniques:
Learning Outcome 1: ***Judge whether a given problem can be formulated as a constraint satisfaction problem***
Learning Outcome 2: ***Formulate constraint satisfaction problems***
Learning Outcome 6: ***Appreciate applications of constraint technology***

(*This is a simplified version of the problem described in Pascal Van Hentenryck's book "The OPL Optimization Programming Language", MIT Press, 1999*)

The question is difficult, especially in exam situations. Students will be forgiven for minor misinterpretation of the problem. Marks will be given generously.

(a) [10%] Modeling is important because without a model, one cannot apply any constraint techniques. Besides, some models make the problem easier to solve than others.

(b)

Variables: ***n*** **variables for** ***n*** **locations,** Loc_1, Loc_2, …, Loc_n
Domains: **All the variables have the domain {*True*, *False*}**
Constraints:

{Following are marking guides; total marks in this section not to exceed [90%]}

(i) Location is captured by the variable-domain definition, so no constraint is needed [10%]
(ii) At most *k* out of the *n* variables can take the value *True* [10%]
(iii) This cannot be expressed by the above variables and domains. One needs variables to express which warehouse serves which store:
{correctly pointing out this fact} [30%]
{correctly defining the new variables below} [30%]

New variables: $Supplier_1$, …, $Supplier_m$, **where** $Supplier_j$ **represents the warehouse that supplies to Store j**
Domains: **For all j, domain of** $Supplier_j$ **[1..n]**

With these variables, one can state that
j Constraint: **(j=1..n, one for each location) at most** CP_j **of the variables** $Supplier_1$, …, $Supplier_m$ **may take value j**
Besides, if $Supplier_i = j$ **for any i, then** Loc_j **= True**

(iv) The variables $Supplier_1$, …, $Supplier_m$ **and their domains defined above capture this already, so no extra constraints are needed [10%]**
(v) This is a definition, not a constraint [10%]
(vi) This is an m-ary "cost constraint" that involves all the constraints $Supplier_1$, …, $Supplier_m$. **[10%]**
The "cost constraint" can be expressed in a function: [20%]

```
Cost = 0;
For s = 1 to m,
    {
        /* let w be the value of Supplier_s   */
        Cost = Cost + SC_w,s
    }
Return Cost
```

Question 2 (Fundamental issues of Constraint Satisfaction):

(a) Finite constraint satisfaction problems are problems with a finite number of variables and finite domains. Explain the implication of these properties in the search space. Do not bring constraints into the discussion in this part of the question. [20%]

(b) How do the above characteristics of constraint satisfaction problems enable the following techniques?

- **i. Problem reduction [20%]**
- **ii. Lookahead algorithms [20%]**
- **iii. Algorithms that learn no-good sets when backtracking takes place [20%]**

(c) Which of the algorithms above cannot be applied if the domain size of all the variables are infinite (e.g. variables can take any integers to be their values) [20%]

Answer 2:

This question tests the students' understanding of the fundamental concepts of constraint satisfaction.
Learning Outcome 2: Formulate constraint satisfaction problems
Learning Outcome 3: Understand basic techniques for solving constraint satisfaction problems
Learning Outcome 6: Appreciate applications of constraint technology

These points were mentioned in lectures, explained in my book, but not written down in the handouts. This question will test students' understanding of these fundamental issues.

(a) Characteristics of CSPs
Finite variables means fixed depth in the search tree. [10%]
Finite domains means fixed number of branches at each level. [10%]
[Note: answers that address he size of the search space only should get no more than 10%]

(b) Techniques that exploit the characteristics of CSPs:
- **i. Fixed domains allows enumeration of values, hence problem reduction can be applied. [20%]**
- **ii. Fixed domains also allows lookahead algorithms to be applied. [20%]**
- **iii. Sibling subtrees are similar, hence learning no-good sets can be useful. [20%]**

[Note: answers that explain what these techniques do but <u>do not</u> explain how they exploit the search space's characteristics should get no more than 8% on each point.]

(c) Infinite domains:
[Note: Students confused "infinite domain size" with "infinite number of variables"]
- **i. Problem reduction is not possible because one cannot enumerate all the values. [5%]**
- **ii. Lookahead is not applicable as problem reduction is not applicable when domains are infinite. [5%]**
- **iii. In reality, it is impossible to enumerate all the values of a variable, which means one cannot establish all the causes for the wiping out of the domain, except in special cases (e.g. if we have $x<y$, and $y=4$, then we can eliminate all the values of $x \geq 4$.) Theoretically one can still learn no-goods, though one has to modify the control strategy because depth first search will never backtrack due to the availability of infinite branches! Iterative broadening might be relevant (not in syllabus) [10%]**

Question 3 (Hill Climbing for Constraint Satisfaction):

The Graph Colouring Problem:

Given a graph (V, E), where V is a set of nodes and E is a set of edges (x, y), and a fixed set of colours S, one has to assign a colour in S to each node in V such that no two connected nodes take the same colour. The task is to find a set of assignments that satisfies all the constraints.

Here is a possible formulation of this problem:

Variables:	**$Z = \{x_1, x_2, \ldots, x_n\}$ where x_i represents the colour taken by node *i* in V**
Domains:	**For all x_i in Z, the domain of x_i is S, the set of colours available**
Constraints:	**For all x_i and x_j in Z, if (i, j) is an edge in the graph, i.e. (i, j) is in E, then $x_i \neq x_j$.**
Objective 1:	**to minimize the number of constraints being violated**

(a) Explain the principles of hill climbing. [20%]

(b) Propose a hill climbing approach to solving the Graph Colouring Problem as formulated above. In particular, explain your neighbourhood function clearly. [30%]

(c) Suppose the task is to find a set of assignments that, having satisfied Objective 1, minimizes the number of colours used:

Objective 2: to minimize the number of colours being assigned to the nodes

Do you have to change your neighbourhood function defined in (b) in response to this change of objective? Why? [20%]

(d) Going back to the original problem (i.e. ignore Objective 2), suppose one is required to use every colour the same number of times (assuming that n is divisible by s, where s is the number of colours available). Is your neighbourhood function defined in (b) above appropriate for this problem? Justify your answer carefully. [30%]

Answer 3:

Learning Outcome 3: ***Understand basic techniques for solving constraint satisfaction problems***
Learning Outcome 4: ***Apply constraint satisfaction techniques to solve given constraint satisfaction problems***
Learning Outcome 5: ***Implement basic constraint satisfaction and optimisation algorithms***
Learning Outcome 6: ***Appreciate applications of constraint technology***

(a) [bookwork] Hill climbing involves a representation of candidate solutions, an objective function, a neighbourhood function and optionally heuristics on which neighbour to move to next. Start with a random position, iteratively move to better candidate solutions according to the given objective function. Stop when no improvement is available from the current solution.

(b) Here is one possible approach. All valid approaches will be accepted.

Representation:	**to search in the space of complete compound labels: ($<x_1, v_1> \ldots <x_n, v_n>$)**
Objective function:	**to minimize the number of constraints being violated**
Neighbourhood Function:	**to change one value in the current compound label**

(c) To minimize the number of colours, all one needs to do is to change the objective function. There is no need to change the neighbourhood function.
[This is a short answer, but students must have a clear idea of what hill climbing is in order to produce his answer.]

(d) The objective function must be changed to reflect this new constraint. This will create many local minimum in the above neighhourhood function. This is because if ($<x_1, v_1> \ldots <x_n, v_n>$) satisfies the new equal-colour-number constraint, changing any single value to another colour will violate this constraint. So changes are less likely under this new objective function.
One could change the neighbourhood function to swapping two different values in each iteration; i.e. moving from ($<x_1, v_1> \ldots <x_i, a> \ldots <x_j, b> \ldots <x_n, v_n>$) to ($<x_1, v_1> \ldots <x_i, b> \ldots <x_j, a> \ldots <x_n, v_n>$)

Question 4 (Application of Algorithms and Heuristics):

Explain what algorithms or heuristics are relevant under the following situations. Justify your answers carefully. There is no need to explain the details of the algorithms or heuristics that you propose unless they are relevant to your justifications.

(i) **The goal is to reducing the chance of backtracking in a problem where some variables are involved in more constraints than others, but overall the constraint graph is sparse [25%]**

(ii) **The goal is to reduce the distance of backtracking in a problem where each variable is constrained by a very small number of other variables [25%]**

(iii) **The goal is to reducing the number of constraint checks in a problem where each constraint check is very computationally expensive [25%]**

(iv) **The goal is to solve the problem quickly in a problem which constraint graph forms a tree. [25%]**

Answer 4:

This question will test the students' understanding of the techniques. It also tests the students' breadth of knowledge in constraint satisfaction.

Learning Outcome 3: ***Understand basic techniques for solving constraint satisfaction problems***

Learning Outcome 4: ***Apply constraint satisfaction techniques to solve given constraint satisfaction problems***

The answers do not have to be long.

(i) **The *minimal width ordering heuristic* minimizes the maximum dependency of a variable to the committed labels. Therefore, by searching under the minimal width ordering, one may reduce the chance of backtracking.**

(ii) **The *minimal bandwidth ordering heuristic* minimizes the maximum distance between mutually constraining variables. Therefore, by searching under the minimal bandwidth ordering, one may reduce the chance of having to backtrack beyond the maximum bandwidth.**

(iii) **The basic algorithm to use is *BackMarking*. Obviously if explores fewer nodes in the search space, one reduces the number of constraint checks. Therefore, any hybrid with BackMarking could be considered.**

(iv) **The algorithm to use is Dechter's *TreeSearch Algorithm*:**

(1) order the variables so that it has a width of one, i.e. all variables are to be labeled after their parent variables (according to the constraint tree);

(2) achieve Directional Arc-consistency according to the order defined above; and

(3) perform a backtrack-free search.

The chance is, this will solve the problem faster than brute-force search, though there is no guarantee for this.