CS410: Artificial Intelligence 2021 Fall
Homework 2: Constraint Satisfaction Problems
Due date: 23:59:59 (GMT +08:00), October 22 2021

1. **A* graph search**. Consider the following undirected graph shown in Figure 1 where we are searching from start state A to goal state G. The number over each edge is the transition cost. Additionally, we are given a heuristic function h as follows: $\{h(A) = 7, h(B) = 5, h(C) = 6, h(D) = 4, h(E) = 3, h(F) = 3, h(G) = 0\}$. Assume that, in case of ties, the search procedure uses an alphabetical order for tie-breaking.

   Find the sequence of nodes expanded by A* graph search algorithm, with problem-solving steps (i.e., updates for the frontier and explored set).

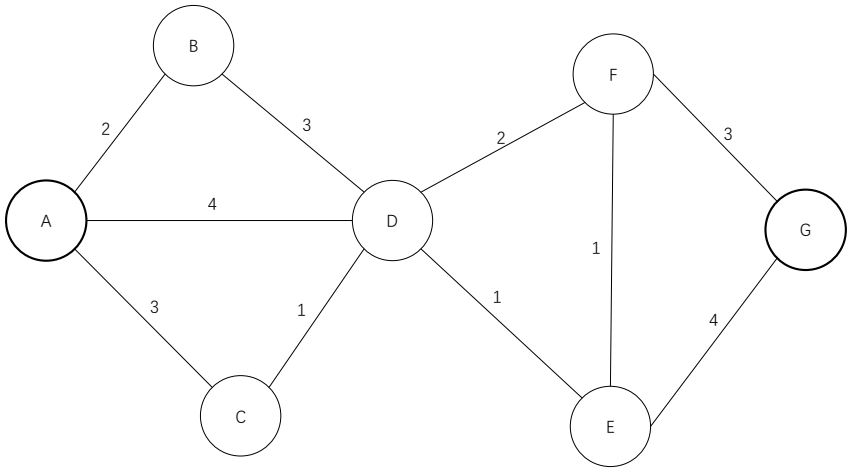


Figure 1: Problem 1.

Solution:

(a) $Frontier = \{A^7\}, Explored = \{\emptyset\} \rightarrow A$
(b) $Frontier = \{B^7, D^8, C^9\}, Explored = \{A\} \rightarrow B$
(c) $Frontier = \{D^8, C^9\}, Explored = \{A, B\} \rightarrow D$
(d) $Frontier = \{E^8, C^9, F^9\}, Explored = \{A, B, D\} \rightarrow E$
(e) $Frontier = \{C^9, F^9, G^9\}, Explored = \{A, B, D, E\} \rightarrow C$

(f) $Frontier = \{F^9, G^9\}, Explored = \{A, B, D, E, C\} \rightarrow F$

(g) $Frontier = \{G^9\}, Explored = \{A, B, D, E, C, F\} \rightarrow G$

(h) $Frontier = \{\}, Explored = \{A, B, D, E, C, F, G\} \rightarrow End$

2. **CSP formulation**. Consider the following three problems:

   (a) Rectilinear floor-planning: find non-overlapping places in a large rectangle for a number of smaller rectangles.

   (b) Class scheduling: There is a fixed number of professors and classrooms, a list of classes to be offered, and a list of possible time slots for classes. Each professor has a set of classes that he or she can teach.

   (c) Hamiltonian tour: given a network of cities connected by roads, choose an order to visit all cities in a country without repeating any.

   Determine each as a planning problem or an identification problem and explain why. Give precise formulations for each problem as constraint satisfaction problems.

   Recall a CSP consists of three components, X, D, and C. Note that there are many valid formulations, but you only need to provide one.

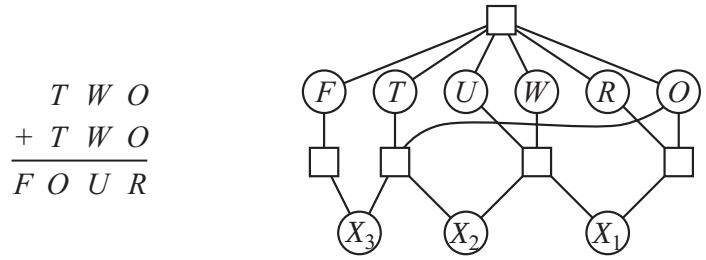
   Solution (not unique):

   (a) Rectilinear floor-planning
   - Identification problem
   - X: position coordinates for each small rectangle
   - D: real number for each variable
   - C: no overlapping between each pair of small rectangles & small rectangles should not be placed outside

   (b) Class scheduling
   - Identification problem
   - X: teachers T_{ij}, subjects S_{ij}, classrooms i, and time slots j
   - D: corresponding choices for each variable
   - C: no teacher is assigned to two classes which take place at the same time (i.e., $T_{ij} \neq T_{kj}$) & teacher can teach (i.e., if teacher t is assigned to T_{ij}, then S_{ij} is assigned a value from the set of subjects that t can teach)

   (c) Hamiltonian tour
   - **Identification problem** (only the goal matters)
   - X: a sequence of cities on the tour
   - D: each variable should be a city
   - C: neighboring cities should be adjacent & all variables have a different value

3. **Forward checking**. Solve the cryptarithmetic problem shown in Figure 3 step by step, using the strategy of backtracking with forward checking. Assume the variable order is $X_3 \to F \to X_2 \to X_1 \to O \to T \to R \to U \to W$, and the value order is increasing. Note that different variables have different domains (e.g., the domain for X_3 is $\{0, 1\}$, the domain for O is $\{0, 1, \ldots, 9\}$, and the domain for F is $\{1, 2, \ldots, 9\}$).



(a) (b)

(a) A cryptarithmetic problem. Each letter stands for a distinct digit; the aim is to find a substitution of digits for letters such that the resulting sum is arithmetically correct, with the added restriction that no leading zeroes are allowed. (b) The constraint hypergraph for the cryptarithmetic problem, showing the *Alldiff* constraint (square box at the top) as well as the column addition constraints (four square boxes in the middle). The variables X_1, X_2, and X_3 represent the carry digits for the three columns.

Figure 2: Problem 3.

Solution:

- Assign 0 to X_3
- No option left for F. Backtrack to X_3
- Assign 1 to X_3
- Assign 1 to F
- Assign 0 to X_2
- Assign 0 to X_1
- Assign 0 to O
- No option left for R. Backtrack to O
- Assign 2 to O (O must be an even number)
- Assign 6 to T
- Assign 4 to R
- Assign 0 to U
- No option left for W. Backtrack to U

- Assign 8 to U (8 is the only option left for U)
- No option left for W. Backtrack to O
- Assign 4 to O (O must be an even number)
- Assign 7 to T
- Assign 8 to R
- Assign 0 to U
- No option left for W. Backtrack to U. It happens repeatedly for $U = 2$ and $U = 4$
- Assign 6 to U
- Assign 3 to W
- Finally, we have $F = 1, O = 4, T = 7, R = 8, U = 6, W = 3$

4. **AC-3**. Consider the following CSP:

   - Variables: A, B, C, D
   - Domain: $\{1, 2, 3\}$
   - Constraints: $A \neq B, B > C, C < D$

   Solve the problem using the strategy of backtracking search with AC-3 algorithm. Give the problem-solving steps, specifying each assignment when backtracking and the consequence of each pop operation in AC-3 (i.e., what values you cross off and which arcs you push to the queue).

   Suppose the value order is descending and the variable order is alphabetical, which both queue initialization and neighbor consideration follow (i.e., the queue is initialized as $A \to B, B \to A, \dots$).

   Solution:

   (a) Assign $A = 3$

   (b) Run AC-3 with initialized queue $\{B \to A\}$

      i. Pop $B \to A$. Now $B \in \{1, 2\}$. Push $A \to B$ and $C \to B$ to the queue

      ii. Pop $A \to B$. Nothing happens

      iii. Pop $C \to B$. Now $C \in \{1\}$. Push $B \to C$ and $D \to C$ to the queue

      iv. Pop $B \to C$. Now $B \in \{2\}$. Push $A \to B$ and $C \to B$ to the queue

      v. Pop $D \to C$. Now $D \in \{2, 3\}$. Push $C \to D$ to the queue

      vi. Pop all the arcs in the queue. Nothing happens

   (c) Assign $B = 2$ and $C = 1$. Running AC-3 makes no value cross-off

   (d) Assign $D = 3$. Running AC-3 makes no value cross-off

5. **K-consistency & tree structure**. Consider the following three questions:

   (a) Give a concrete CSP example to show that k-consistency does not imply $(k+1)$-consistency for some $k \geq 2$.

   (b) Give a concrete CSP example to show that k-consistency does not imply $(k-1)$-consistency for some $k \geq 3$.

   (c) Why graphs with cycles can not be applied with the algorithm for tree-structured CSPs (introduced in Page 77-83, Lecture 4)? What step in the analysis fails? Why this step holds for trees? Give an example to explain the failure.

   Solution (not unique):

   (a) Consider a CSP with

   - Variables: X_1, X_2, and X_3
   - Domains: $\{0, 1\}$
   - Constraints: $X_1 \neq X_2$, $X_2 \neq X_3$, and $X_3 \neq X_1$

   Obviously it is 2-consistent. However, it is not 3-consistent (e.g., X_3 has no value left if $X_1 = 0$ and $X_2 = 1$).

   (b) Consider a CSP with

   - Variables: X_1, X_2, and X_3
   - Domains: $\{0\}$, $\{0\}$, and $\{0, 1\}$ for X_1, X_2, and X_3 respectively
   - Constraints: $X_1 \neq X_3$ and $X_2 \neq X_3$

   It is 3-consistent that can be checked in seconds. However, it is not 2-consistent (e.g., X_1 and X_2 has no value left if $X_3 = 0$).

   (c) The main idea is already discussed in the lecture.

   - Assign forward might fail for graphs with cycles, given that nodes can have multiple parents
   - Assign forward
   - Each node in trees has at most one parent
   - The example presented in the lecture suffices