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).

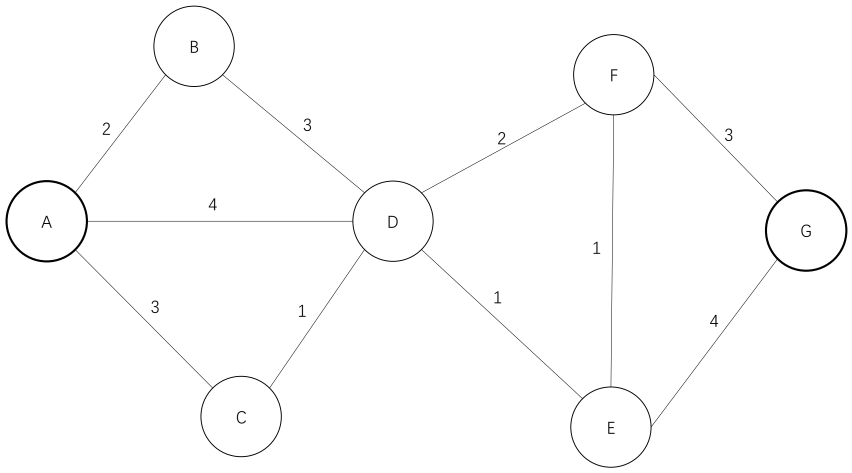


Figure 1: Problem 1.

Solution:
The search begin at the state A, so we put the vertex A into the frontier and start to run $\mathbf{A}^*$ algorithm.

(a) step 1:Frontier={A:7},Explored=∅. Choose A
(b) step 2:Frontier={B:7,C:9,D:8},Explored={A}. Choose B
(c) step 3:Frontier={C:9,D:8},Explored={A,B}. Choose D

(d) step 4:Frontier={C:9,E:8,F:9},Explored={A,B,D}. Choose E

(e) step 5:Frontier={C:9,F:9,G:9},Explored={A,B,D,E}. Choose C

(f) step 6:Frontier={F:9,G:9},Explored={A,B,D,E,C}. Choose F

(g) step 7:Frontier={G:9},Explored={A,B,D,E,C}. Choose G

So the sequence of nodes expanded is A,B,D,E,C,F,G.

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:
   To distinguish planning with identification problem, we are supposed to consider the significance of the path of searching for the goal. If the path has to do with the answers, it is a planning problem, otherwise it is a identification problem.

   (a) It's an identification problem. We only need to determine whether we are able to place these rectangles,but how to place them does not matters.

      i. **X(Variables):**$X_i = (x_i, y_i), i = 1, 2 \dots n$ represents the i^{th} smaller rectangke, and x_i, y_i represents the $x - axis$ and $y - axis$ of it.

      ii. **D(Domains):**We assume a_i, b_i to be the length and width of the i^{th} smaller rectangle and A, B to be those of large rectangle. We define domain of $X_i \in [\frac{a_i}{2}, A - \frac{a_i}{2}] \times [\frac{b_i}{2}, B - \frac{b_i}{2}]$, where $\times$ means Cartesian product.

      iii. **C(Constraints:)**Randomly choose two different subscripts $i \neq j$, we have the constraint:

$$|x_i - x_j| \geq \frac{a_i + a_j}{2} \quad and \quad |y_i - y_j| \geq \frac{b_i + b_j}{2} \tag{1}$$

(b) It's an identification problem. We only need to determine whether we are able to assign professors,classrooms and time for each class rather than concern how and when to assign them.

   i. **X(Variables):**$X_i = (p_i, r_i, s_i), i = 1, 2 \dots n$ represents assigning professor p_i, classrooms r_i and time slots t_i for the i^{th} class.

   ii. **D(Domains):** We only need to guarantee the professor, classrooms and time slots are legal. So the Domains of $X_i \in$ $Domain(professors) \times Domain(classrooms) \times Domain(time\ slots)$.

   iii. **C(Constraints:)**The constraints are that there is no conflict between two class, which means for random $i \neq j$:

$$t_i \neq t_j \quad or \quad (p_i \neq p_j \quad and \quad r_i \neq r_j) \tag{2}$$

(c) It's an identification problem. Though there may be many orders to visit all cities, they all have the same depth, which is the number of the cities. So the path to the goal does not matters.

   i. **X(Variables):**An order $X = (x_1, x_2 \dots x_n)$ represents the trip. x_i means the city we visit at i^{th} times.

   ii. **D(Domains):** For each city we visit, it must be in the Domain of the cities. So $X \in Domain(city)^n$.

   iii. **C(Constraints:)**The constraints are that we cannot visit one city more than twice and the path exists in the graph. So we can conclude the constraint that(G means the graph of the cities):

$$\begin{gathered} \forall 0 \leq i \leq n-1, x_i x_{i+1} \in G \\ \forall 0 \leq i < j \leq n, x_i \neq x_j \end{gathered} \tag{3}$$

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 $\{0, 1, \dots, 9\}$, and the domain for F is $\{1, 2, \dots, 9\}$).

   Solution:
   We have the constraints that:

$$\begin{gathered} 2 * O = R + 10 * X_1 \\ 2 * W + X_1 = U + 10 * X_2 \\ x * T + X_2 = O + 10 * X_3 \\ F_1 = X_3 \\ X_1, X_2, X_3 \in 0, 1 \\ F, T, U, W, R, O \quad are \quad all \quad different \end{gathered} \tag{4}$$

Step	X_3	F	X_2	X_1	O	T	R	U	W	assignment
1	0, 1	1~9	0,1	0, 1	0~9	0~9	0~9	0~9	0~9	start
2	0	$\emptyset$	0,1	0, 1	0~9	0~9	0~9	0~9	0~9	$X_3 = 0$
3	1	1	0,1	0, 1	0~9	0~9	0~9	0~9	0~9	$X_3 = 1$
4	1	1	0,1	0, 1	0,2~9	5~9	0,2~9	0,2~9	0,2~9	F=1
5	1	1	0	0, 1	0,2,4,6,8	5~9	0,2~9	0,4~9	0,2,3,4	$X_2 = 0$
6	1	1	0	0	0,2,4	5~9	0,4,8	0,4,6,8	0,2,3,4	$X_1 = 0$
7	1	1	0	0	0	5~9	$\emptyset$	0,4,6,8	0,2,3,4	O=0
8	1	1	0	0	2	6	4	0,4,6,8	0,3,4	O=2
9	1	1	0	0	2	6	4	0,4,8	0,3,4	T=6
10	1	1	0	0	2	6	4	0,8	0,3	R=4
11	1	1	0	0	2	6	4	0	$\emptyset$	U=0
12	1	1	0	0	2	6	4	8	$\emptyset$	U=8
13	1	1	0	0	4	7	8	0,6,8	0,2,3	O=4
14	1	1	0	0	4	7	8	0,6,8	0,2,3	T=7
15	1	1	0	0	4	7	8	0,6	0,2,3	R=8
16	1	1	0	0	4	7	8	0	$\emptyset$	U=0
17	1	1	0	0	4	7	8	6	3	U=6
18	1	1	0	0	4	7	8	6	3	W=3

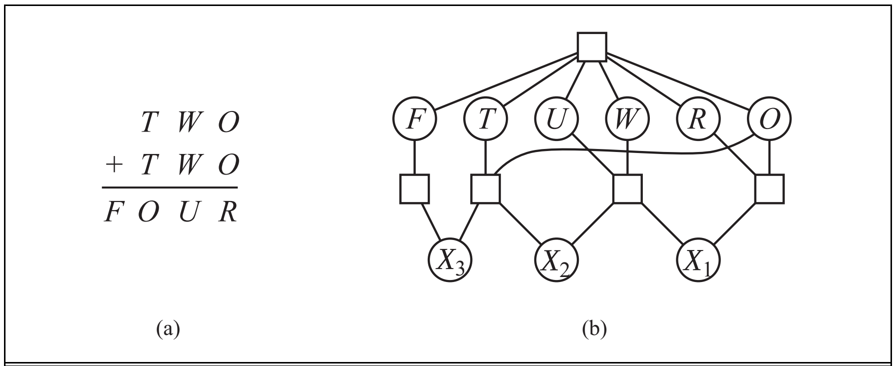
So the answer is $\{X_3 = 1, X_2 = 0, X_1 = 0, F = 1, O = 4, T = 7, R = 8, U = 6, W = 3\}$. So the formation is $734 + 734 = 1468$.

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:
   According to the table below.



(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.

step	A	B	C	D	Q	action
1	1,2,3	1,2,3	1,2,3	1,2,3	pppqqq	start
2	3	1,2,3	1,2,3	1,2,3	B→A	assign A=3
3	3	1,2	1,2,3	1,2,3	A→B,C→B	pop B→A
4	3	1,2	1,2,3	1,2,3	C→B	pop A→B
5	3	1,2	1	1,2,3	B→C,D→C	pop C→B
6	3	2	1	2,3	A→B,C→B,C →D	pop B→C
7	3	2	1	2,3	pppqqq	pop A→B,C→B,C →D
8	3	2	1	2,3	A→B,C→B	assign B=2
9	3	2	1	2,3	pppqqq	pop A→B,C→B
10	3	2	1	2,3	B→C,D→C	assign C=2
11	3	2	1	2,3	pppqqq	pop B→C,D→C
12	3	2	1	3	C→D	assign D=3
13	3	2	1	3	pppqqq	pop C→D

So the answer is $A = 3, B = 2, C = 1, D = 3$.

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:

(a) For any complete graph $G = K_n$, we assign k_{n-1} colors to every node. The graph G satisfies $n-1$-consistency while it does not satisfies n-consistency.

   i. It satisfies n-1-consistency. For any n-1 nodes, if we have assigned n-2 nodes, it at most use n-2 colors, so there is at least 1 color that has no been used, then we assign it to the $n-1^{th}$ node.

   ii. It does not satisfy n-consistency. We are not able to find any consistent assignment to n nodes. That's because there are n-1 colors and it is a complete graph.

   For example, n=3, refer to figure 5a,n=4,refer to figure 5a.

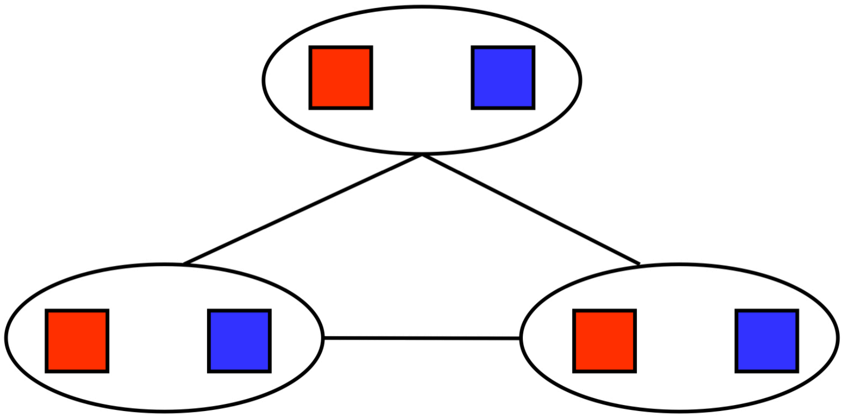


Figure 3: n=3

(b) The graph 5b satisfies 3-consistency but does not satisfy 2-consistency.

   i. It satisfies 3-consistency. If we choose A and B(A and C is the same), we must have assigned A to be blue and B to be red, then

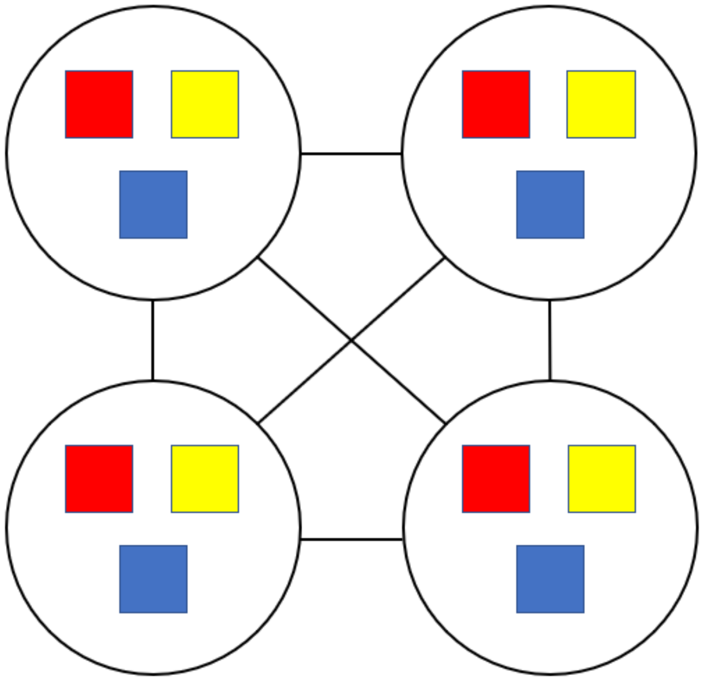


Figure 4: n=4

we assign C to be red. If we choose B and C, we only need to assign A to be blue.

ii. It does not satisfy 2-consistency. If we choose A and assign it red. Then we choose A and B, we cannot assign B any color.

(c) Refer to the 5c. If the yellow edge is not in the graph, we can use the algorithm, but when we add this edge, the algorithm will fail. We fail at the step assignment forward, refer to 5c.

The node D will have 2 parents in the graph. When it is the tree, it only has one parent, after assigning its parent, we only need to choose one of colors it left we assign it. We only need to guarantee the color is different from its parent. And it at least left one color that different from its parent, otherwise it will only has one color, then we have removed this color from its parent's domain when we remove backward.

However, when it has more than 1 parents, the assignment of its parents may not be consistent with any of the colors in its domains. Such as we assign A red and C green, and all the colors in D's domain

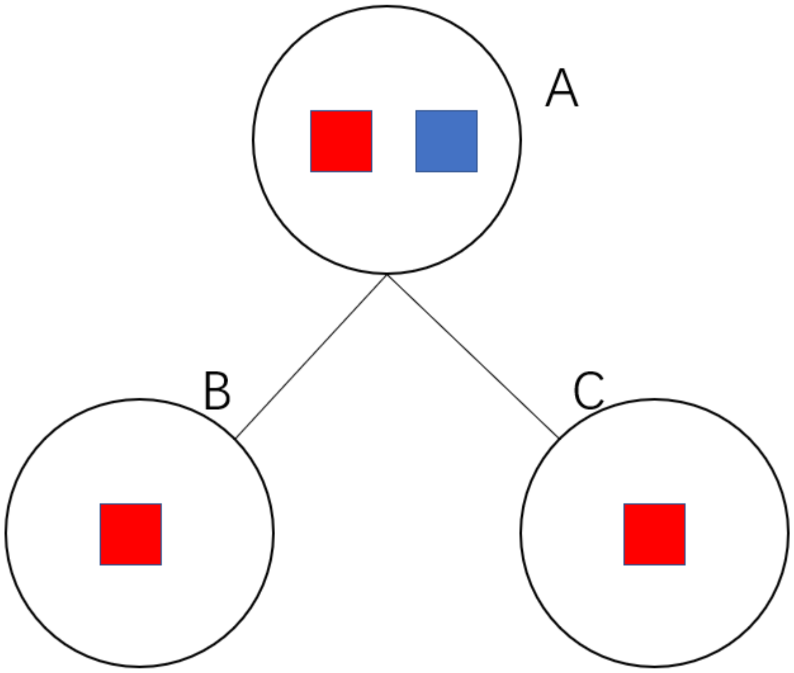


Figure 5: counter-example

.

will conflict to either A or C.

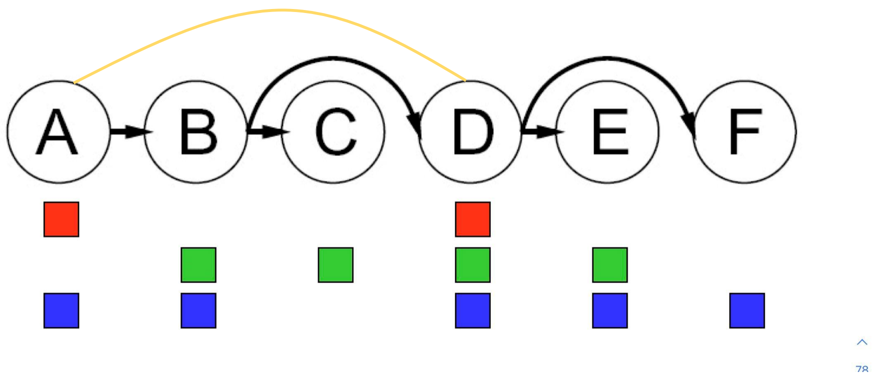


Figure 6: Tree Algorithm Failure

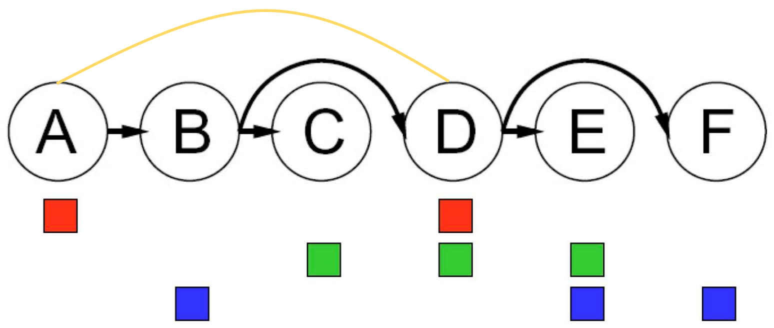


Figure 7: Assignment forward