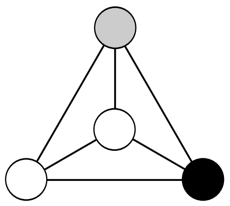
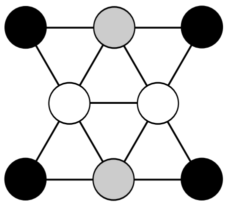
CS 188
Spring 2023

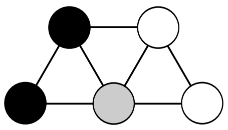
Midterm Review CSPs Solutions

Q1. CSPs: Graph Coloring

In this question we are considering CSPs for map coloring. Each region on the map is a variable, and their values are chosen from {black, gray, white}. Adjacent regions cannot have the same color. The figures below show the constraint graphs for three CSPs and an assignment for each one. None of the assignments are solutions as each has a pair of adjacent variables that are white. For both parts of this question, let the score of an assignment be the number of satisfied constraints (so a higher score is better).


(1)


(2)


(3)

(a) Consider applying Local Search starting from each of the assignments in the figure above. For each successor function, indicate whether each configuration is a local optimum and whether it is a global optimum (note that the CSPs may not have satisfying assignments).

Successor Function	CSP	Local optimum?	Global Optimum?
Change a single variable	(1)	[✓ **Yes**] No	[✓ **Yes**] No
	(2)	[✓ **Yes**] No	Yes [✓ **No**]
	(3)	Yes [✓ **No**]	Yes [✓ **No**]
Change a single variable, or a pair of variables	(1)	[✓ **Yes**] No	[✓ **Yes**] No
	(2)	Yes [✓ **No**]	Yes [✓ **No**]
	(3)	Yes [✓ **No**]	Yes [✓ **No**]

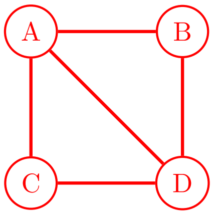
Q2. CSPs: Apartments

Four people, A, B, C, and D, are all looking to rent space in an apartment building. There are three floors in the building, 1, 2, and 3 (where 1 is the lowest floor and 3 is the highest). Each person must be assigned to some floor, but it's ok if more than one person is living on a floor. We have the following constraints on assignments:

- A and B must not live together on the same floor.
- If A and C live on *the same* floor, they must both be living on floor 2.
- If A and C live on *different* floors, one of them must be living on floor 3.
- D must not live on the same floor as anyone else.
- D must live on a higher floor than C.

We will formulate this as a CSP, where each person has a variable and the variable values are floors.

(a) Draw the edges for the constraint graph representing this problem. Use binary constraints only. You do not need to label the edges.



(b) Suppose we have assigned C = 2. Apply forward checking to the CSP, filling in the boxes next to the values for each variable that are eliminated:

A	🟥 1	☐ 2	☐ 3
B	☐ 1	☐ 2	☐ 3
C		☐ 2	
D	🟥 1	🟥 2	☐ 3

(c) Starting from the original CSP with full domains (i.e. without assigning any variables or doing the forward checking in the previous part), enforce arc consistency for the entire CSP graph, filling in the boxes next to the values that are eliminated for each variable:

A	🟥 1	☐ 2	☐ 3
B	☐ 1	☐ 2	☐ 3
C	☐ 1	☐ 2	🟥 3
D	🟥 1	☐ 2	☐ 3

(d) Suppose that we were running local search with the min-conflicts algorithm for this CSP, and currently have the following variable assignments.

A	3
B	1
C	2
D	3

Which variable would be reassigned, and which value would it be reassigned to? Assume that any ties are broken alphabetically for variables and in numerical order for values.

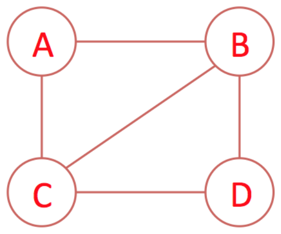
A is reassigned value 2.

Q3. CSPs with Preferences

Let us formulate a CSP with variables A, B, C, D, and domains of $\{1, 2, 3\}$ for each of these variables. A **valid assignment** in this CSP is defined as a complete assignment of values to variables which satisfies the following constraints:

1. B will not ride in car 2.
2. A and B refuse to ride in the same car.
3. The sum of the car numbers for B and C is less than 4.
4. A's car number must be greater than C's car number.
5. B and D refuse to ride in the same car.
6. C's car number must be lesser than D's car number.

(a) Draw the corresponding constraint graph for this CSP.



Although there are several valid assignments which exist for this problem, A, B, C and D have additional "soft" preferences on which value they prefer to be assigned. To encode these preferences, we define utility functions $U_{Var}(Val)$ which represent how preferable an assignment of the value(Val) to the variable(Var) is.

For a complete assignment $P = \{A : V_A, B : V_B, ...D : V_D\}$, the utility of P is defined as the sum of the utility values: $U_A(V_A) + U_B(V_B) + U_C(V_C) + U_D(V_D)$. A higher utility for P indicates a higher preference for that complete assignment. This scheme can be extended to an arbitrary CSP, with several variables and values.

We can now define a modified CSP problem, whose goal is to find the valid assignment which has the maximum utility amongst all valid assignments.

(b) Suppose the utilities for the assignment of values to variables is given by the table below

U	U_A	U_B	U_C	U_D
1	7	10	200	2000
2	6	20	300	1000
3	5	30	100	3000

Under these preferences, given a choice between the following complete assignments which are valid solutions to the CSP, which would be the preferred solution.

- ○ A:3 B:1 C:1 D:2
- [✓ A:3 B:1 C:2 D:3]
- ○ A:3 B:1 C:1 D:3
- ○ A:2 B:1 C:1 D:2

Solution 2 has value $U_A(3) + U_B(1) + U_C(2) + U_D(3) = 5 + 10 + 300 + 3000 = 3315$, which is the highest amongst the choices

To decouple from the previous questions, for the rest of the question, the preference utilities are not necessarily the table shown above but can be arbitrary positive values.

This problem can be formulated as a modified search problem, where we use the modified tree search shown below to find the valid assignment with the highest utility, instead of just finding an arbitrary valid assignment.

The search formulation is:

- State space: The space of partial assignments of values to variables
- Start state: The empty assignment
- Goal Test: State X is a valid assignment
- Successor function: The successors of a node X are states which have partial assignments which are the assignment in X extended by one more assignment of a value to an unassigned variable, as long as this assignment does not violate any constraints
- Edge weights: Utilities of the assignment made through that edge

In the algorithm below $f(node)$ is an **estimator of distance** from *node* to *goal*, ACCUMULATED-UTILITY-FROM-START(*node*) is the sum of utilities of assignments made from the *start-node* to the current *node*.

```
function ModifiedTreeSearch(problem, start-node)
    fringe ← Insert(key : start-node, value : $f(start\text{-}node)$)
    do
        if IsEmpty(fringe) then
            return failure
        end if
        node, cost ← remove entry with maximum value from fringe
        if Goal-Test(node) then
            return node
        end if
        for child in Successors(node) do
            fringe ← Insert(key : child, value : $f(child)$ + Accumulated-Utility-From-Start(child))
        end for
    while True
end function
```

(c) Under this search formulation, for a node X with assigned variables $\{v_1, v_2 .... v_n\}$ and unassigned variables $\{u_1, u_2, u_3 ... u_m\}$

(i) Which of these expressions for $f(X)$ in the algorithm above, is guaranteed to give an optimal assignment according to the preference utilities. (Select **all** that apply)

☐ $f_1 = \min_{Val_1, Val_2, ... Val_m} U_{u_1}(Val_1) + U_{u_2}(Val_2) + ... + U_{u_m}(Val_m)$

■ $f_2 = \max_{Val_1, Val_2, ... Val_m} U_{u_1}(Val_1) + U_{u_2}(Val_2) + ... + U_{u_m}(Val_m)$

☐ $f_3 = \min_{Val_1, Val_2, ... Val_m} U_{u_1}(Val_1) + U_{u_2}(Val_2) + ... + U_{u_m}(Val_m)$ such that the complete assignment satisfies constraints.

■ $f_4 = \max_{Val_1, Val_2, ... Val_m} U_{u_1}(Val_1) + U_{u_2}(Val_2) + ... + U_{u_m}(Val_m)$ such that the complete assignment satisfies constraints.

■ $f_5 = Q$, a fixed extremely high value ($\gg$ sum of all utilities) which is the same across all states

☐ $f_6 = 0$

Because we have a maximum search we need an overestimator of cost instead of an underestimator for the function f, like standard A^* search. ModifiedTreeSearch is A^* search picking the maximum node from the fringe instead of the minimum. This requires an overestimator instead of an understimator to ensure optimality of the tree search.

(ii) For the expressions for $f(X)$ which guaranteed to give an optimal solution in part(i) among $f_1, f_2, f_3, f_4, f_5, f_6$, order them in ascending order of number of nodes expanded by ModifiedTreeSearch. Based on the dominance of heuristics, but modified to be an overestimate instead of an underestimate in standard A* search. Hence the closer the estimate is to the actual cost, the better it does in terms of number of nodes expanded. So the ordering is option 4 $<$ option 2 $<$ option 5.