1 Discussion Questions

(a) What is the difference between Forward Checking and AC-3?

(b) Why would one use the following heuristics for CSP?

(i) Minimum Remaining Values (MRV)

(ii) Least Constraining Value (LCV)

2 CSP: Air Traffic Control

We have five planes: A, B, C, D, and E and two runways: international and domestic. We would like to schedule a time slot and runway for each aircraft to either land or take off. We have four time slots: 1, 2, 3, 4 for each runway, during which we can schedule a landing or take off of a plane. We must find an assignment that meets the following constraints:

- Plane B has lost an engine and must land in time slot 1.
- Plane D can only arrive at the airport to land during or after time slot 3.
- Plane A is running low on fuel but can last until at most time slot 2.
- Plane D must land before plane C takes off, because some passengers must transfer from D to C.
- No two aircrafts can reserve the same time slot for the same runway.

(a) Complete the formulation of this problem as a CSP in terms of variables, domains, and constraints (both unary and binary). Constraints should be expressed implicitly using mathematical or logical notation rather than with words. Make sure to specify variables, domains, and constraints.

For the following parts, we add the following two constraints:

- Planes A, B, and C cater to international flights and can only use the international runway.
- Planes D and E cater to domestic flights and can only use the domestic runway.

(b) The addition of the two constraints above alters the CSP. Specifically, the domain does not need to include the runway type since this information is carried by the variable, and the binary constraints have changed. Determine the new domain and complete the constraint graph for this problem given the original constraints and the two added ones.

(c) What are the domains of the variables after enforcing arc consistency? Begin by enforcing unary constraints. (Cross out values that are no longer in the domain.)

(d) Arc-consistency can be rather expensive to enforce, and we believe that we can obtain faster solutions using only forward-checking on our variable assignments. Using the Minimum Remaining Values heuristic, perform backtracking search on the graph, breaking ties by picking lower values and characters first. List the (variable, assignment) pairs in the order they occur (including the assignments that are reverted upon reaching a dead end). Enforce unary constraints before starting the search.

List of (variable, assignment) pairs:

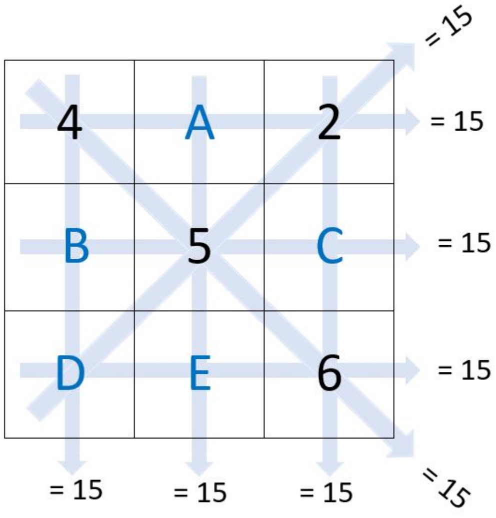
(You don't have to use this table)

A	1	2	3	4
B	1	2	3	4
C	1	2	3	4
D	1	2	3	4
E	1	2	3	4

3 CSP: Magic Square

A magic square is an $n \times n$ grid where each entry is unique and contains one of $\{1, ..., n^2\}$. It has that every row, column, and diagonal sum to the same number.

In this problem, we'll solve a 3×3 magic square by formulating it as a CSP. Each row, column, and diagonal in the 3×3 magic square must sum to 15. We have already filled out some of the numbers for you, but the letters in blue still need to be filled in.



(a) Complete the formulation of this problem as a CSP in terms of variables, domains, and constraints (both unary and binary). Constraints should be expressed implicitly using mathematical or logical notation rather than with words. Make sure to specify variables, domains, and constraints.

Hint: You do not need to create variables for the squares already provided.

(b) Draw the binary constraint graph for this problem. For simplicity, you may ignore the "alldiff" constraint that all variables are unique when creating this graph.

(c) Use the binary constraint graph to run the AC-3 algorithm and find a solution to the magic square.

(d) How would the representation of this problem change if we had three variables in a singular row?

4 MRV and LCV in Action

Raashi, Simrit, and Ihita want to paint "15-281" on the fence tomorrow in honor of their favorite class. They have the following paint collections at home:

- Raashi = Red, Yellow, Green
- Simrit = Red, Yellow, Pink
- Ihita = Red, Pink

Each of them will contribute exactly one bucket of paint from their collections such that no two TAs bring the same paint color.

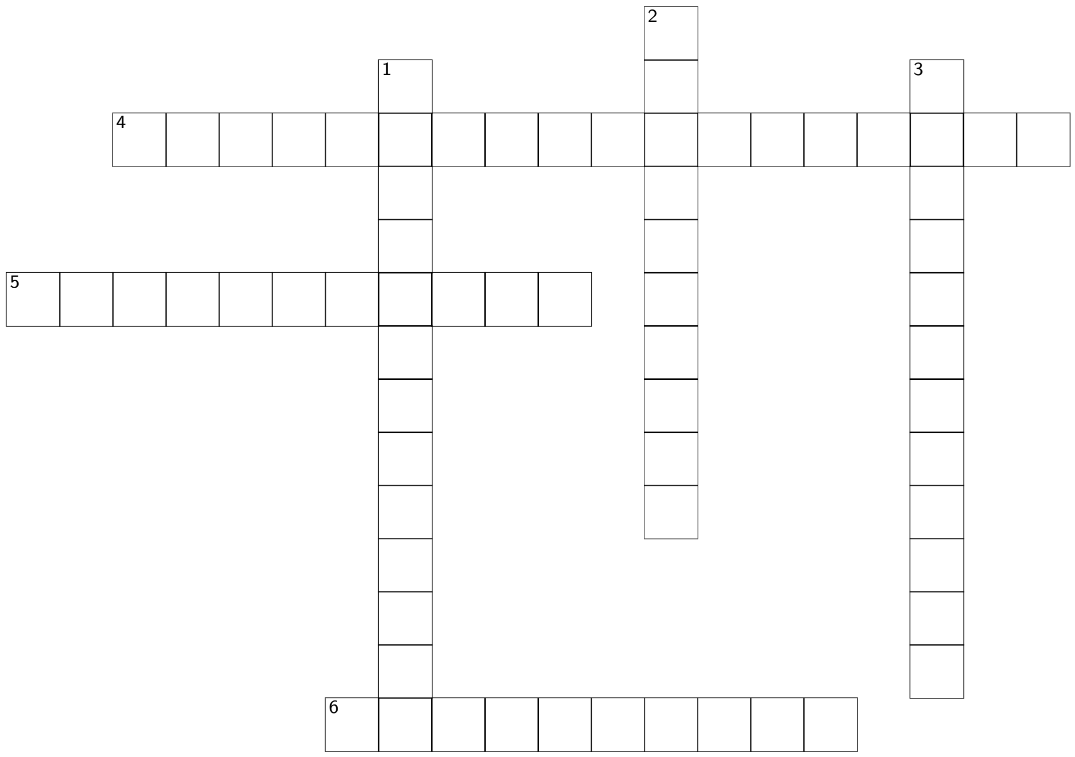
Raashi suddenly remembers going over CSPs in lecture, and suggests formulating this problem as a CSP to determine an assignment of each TA to a paint color so that all three chosen paint colors are different. Simrit isn't fully convinced yet that LCV and MRV will help speed up a constraint satisfaction problem, so Raashi asks for your help to convince Simrit.

(a) Let's use the minimum remaining values (MRV) and least constraining value (LCV) heuristics to assign TAs to paint colors. Recall that the MRV heuristic determines which *variable* to assign, while the LCV heuristic determines which *value* to assign to that variable to. How many times will we backtrack to a previous assignment? Assume we break ties in rainbow order.

(b) Suppose we use a new set of heuristics to assign TAs to paint colors: maximum remaining values (instead of MRV) and most constraining value (instead of LCV). How many times will we backtrack to a previous assignment?

5 Missing in the Mountains

The 15-281 Course Staff decided to climb the Rocky Mountains together over Winter Break. On their way back down from the mountains, they realize they left Simrit at the top of the tallest mountain! They have no idea where on the mountain they are or which mountain they are on, and are worried about how they will find Simrit before lecture. Help the 281 Staff remember all of the local search algorithms they have learned so they can save Simrit!



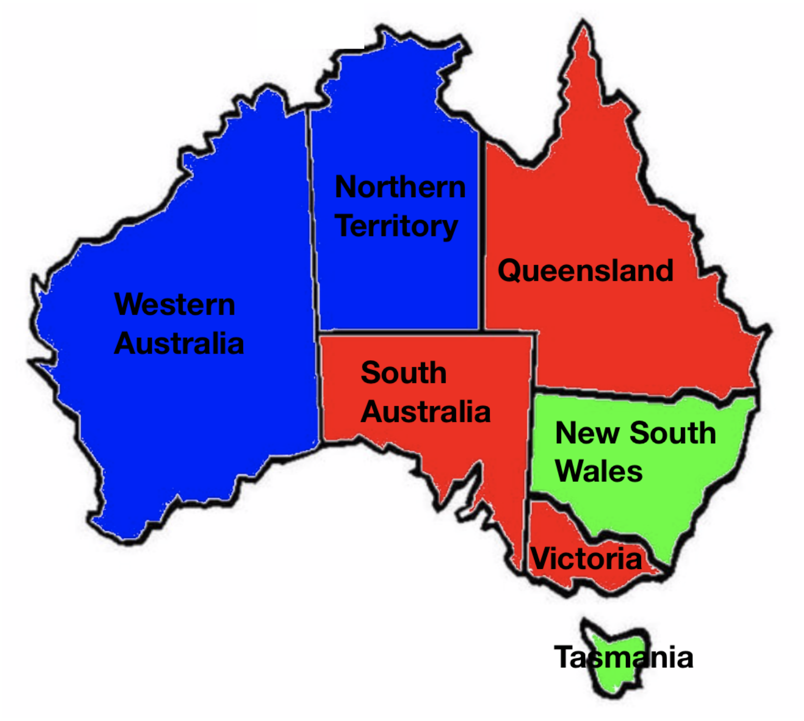
Down

1. A variant of hill-climbing where you conduct a series of searches from randomly generated starting states until the goal is found.
2. A local search technique where you uniformly randomly choose a neighbor to move to.
3. A type of greedy local search where you move uphill to local maxima.

Across

4. A local search technique where you allow for downhill moves but make them rarer as time goes on.
5. A variant of hill-climbing where you generate successors randomly (one by one) until a better one is found.
6. A variant of hill climbing in which you choose a move randomly from the uphill moves, with the probability of a move being chosen dependent on the "steepness" (amount of improvement from making that move).

6 Map Coloring with Local Search



Recall the various local search algorithms presented in lecture. Local search differs from previously discussed search methods in that it begins with a complete, potentially conflicting state and iteratively improves it by reassigning values. We will consider a simple map coloring problem, and will attempt to solve it with hill climbing.

(a) How is the map coloring problem defined (In other words, what are variables, domain and constraints of the problem)? How do you define states in this coloring problem?

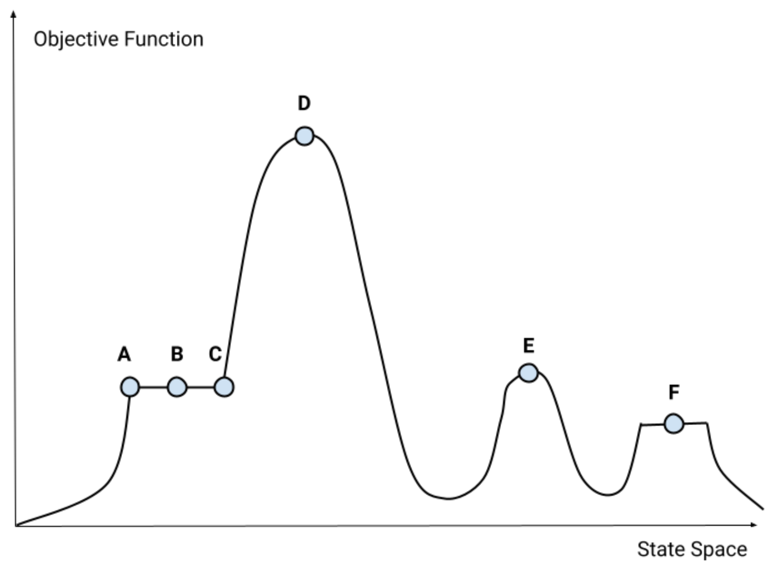
(b) Given a complete state (coloring), how could we define a neighboring state?

(c) What could be a good heuristic be in this problem for local search? What is the initial value of this heuristic?

(d) Use hill climbing to find a solution based on the coloring provided in the graph.

(e) How is local search different from tree search?

7 Local Search Discussion Questions



Consider the state space above in the context of local search. Recall that our goal is to find the state that maximizes the objective.

(a) Consider the points A, B, C, D, E, and F on the graph.

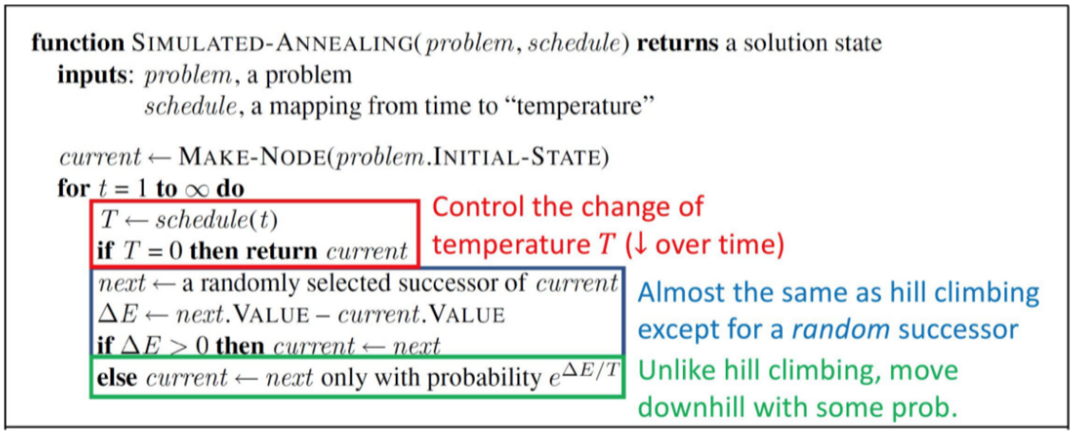
(i) Which of the points on the graph are on a shoulder? Which of those points are local maximums?

(ii) Which of the points on the graph are a "flat" local maximum?

(iii) What is the difference between a shoulder and a "flat" local maximum?

(b) Let's take a look at simulated annealing. Simulated Annealing is quite similar to hill climbing.

- Instead of picking the best move, it picks a random move.
- If the move improves the situation, the move is always accepted.
- Otherwise, it accepts the move with some probability less than 1



```
function SIMULATED-ANNEALING(problem, schedule) returns a solution state
  inputs: problem, a problem
          schedule, a mapping from time to "temperature"

  current ← MAKE-NODE(problem.INITIAL-STATE)
  for t = 1 to ∞ do
      T ← schedule(t)
      if T = 0 then return current
      next ← a randomly selected successor of current
      ΔE ← next.VALUE – current.VALUE
      if ΔE > 0 then current ← next
      else current ← next only with probability e^{ΔE/T}
```

Control the change of temperature T (↓ over time)

Almost the same as hill climbing except for a *random* successor

Unlike hill climbing, move downhill with some prob.

(i) How does the sign of ΔE reflect the "badness" of a move?

(ii) In simulated annealing, we control the temperature T. How does the value of T impact the probability with which we choose a "bad" move?

(c) Mark True or False for each of the following statements.

(i) Regular hill climbing is optimal (i.e., will always find the global maximum)

(ii) Random restart hill climbing is optimal when given an infinite amount of time.

(iii) Simulated annealing allows for downward moves according to some fixed constant temperature T.

(iv) Simulated annealing is generally less time efficient than random walk.

(v) A random walk algorithm is more likely to choose a better neighbor than a worse one.