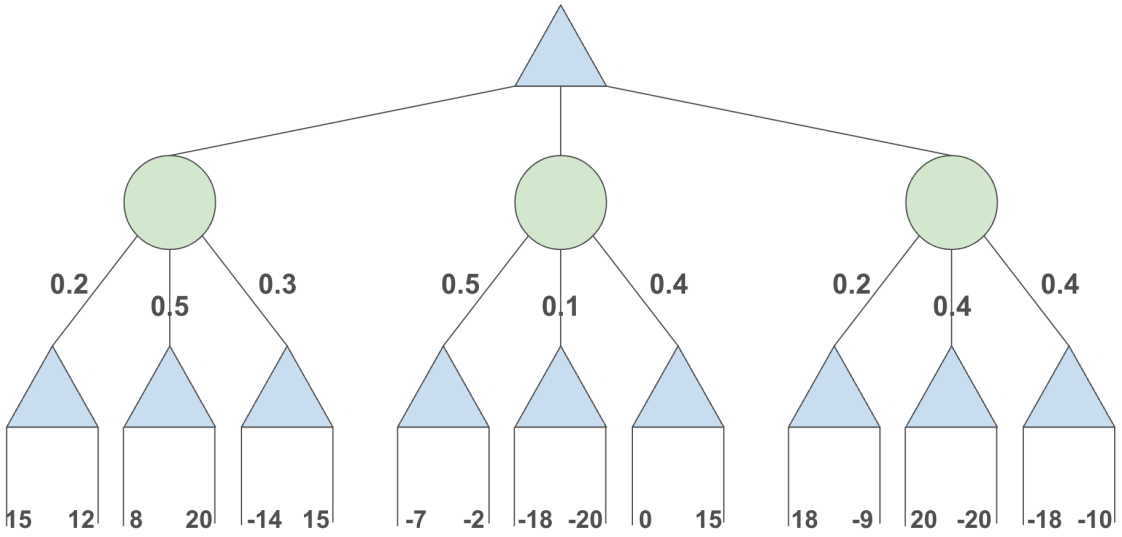
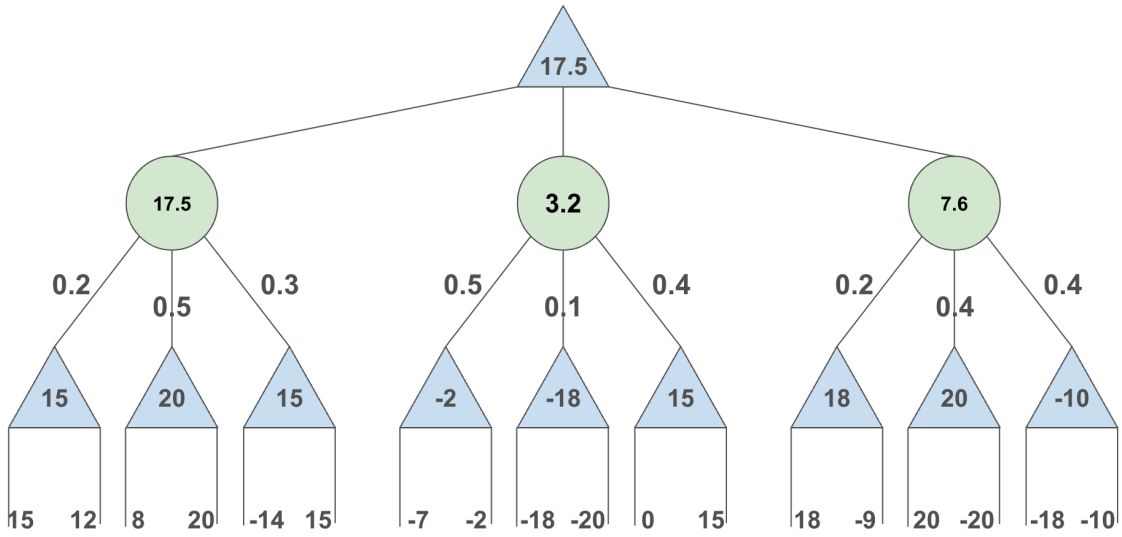
1 Adversarial Search

(a) Consider the following game tree, where the root node is a maximizer. Visit successors from left to right and write the value being returned at each node inside the triangle or circle. Note that each blue triangle is a maximizer and each green circle is a chance node.



The following picture shows the final values at each node.

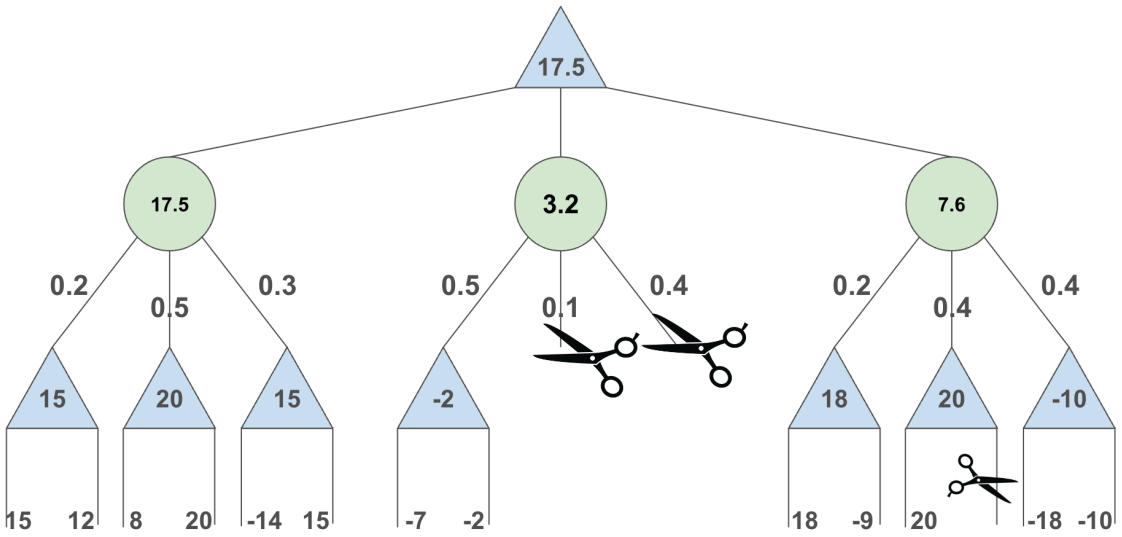


(b) Which of the nodes in the previous tree could have been pruned?

None of them! Any leaf node could have had an arbitrarily large value that resulted in the value of its parent node(s) being larger than the value of previous nodes, thus it was necessary to explore all nodes.

(c) Now you are given that each leaf node is bounded by [-20, 20]. With this information, resolve the previous game tree, this time pruning any unnecessary paths. As before, visit successors from left to right and write the value being returned at each node.

The following picture shows the pruned nodes and final values at each node.



2 Discussion Questions

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

Forward checking and AC-3 both enforce arc consistency, but forward checking is more limited. Whenever a variable X is assigned, forward checking enforces arc consistency only for arcs that are pointing to X, which will reduce the domains of the neighboring variables in the constraint graph. Forward checking stops at this point, but AC-3 will continue to enforce arc consistency on neighboring arcs until there are no more variables whose domain can be reduced. As a result, FC ensures arc consistency of the assigned variable and its neighbors only, while AC-3 ensures arc consistency for the whole graph.

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

(i) Minimum Remaining Values (MRV)

MRV: "Which variable should we assign next?"

- Fail fast
- We have to assign all variables at some point, so we might as well do hard stuff first (allowing us to prune the search tree faster/realize we need to backtrack)

(ii) Least Constraining Value (LCV)

LCV: "Which value should we try next?"

- We just want one solution.
- We don't try all combinations of value, so we should try ones that are likely to lead to a solution.

3 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.

Variables: A, B, C, D, E for each plane.
Domains: a tuple (*runway type, time slot*) for runway type $\in$ {international, domestic} and time slot $\in \{1, 2, 3, 4\}$.
Constraints:

$$B[1] = 1$$

$$D[1] \geq 3$$

$$A[1] \leq 2$$

$$D[1] < C[1]$$

$$A \neq B \ \neq C \neq D \neq E$$

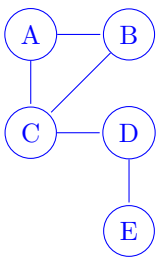
Note here we use B[1] to denote the second value of the tuple assigned to variable B, the time slot value, which is a number in $\{1, 2, 3, 4\}$.

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.

Variables: A, B, C, D, E for each plane.
Domain: $\{1, 2, 3, 4\}$
Constraint Graph:



Explanation of Constraints Graph: We can now encode the runway information into the identity of the variable, since each runway has more than enough time slots for the planes it serves. We represent the non-colliding time slot constraint as a binary constraint between the planes that use the same runways.

(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.)

Enforcing arc consistency with AC-3, we have the following domain as a result:

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

(explanation of process below)
Enforcing unary constraints (in an arbitrary order) first,

1. We cross out 2, 3, 4 from B's domain, adding arcs A → B and C → B to the queue.
2. We cross out 3, 4 from A's domain, adding arcs B → A and C → A to the queue.
3. We cross out 1, 2 from D's domain, adding arcs C → D and E → D to the queue.

Enforcing A → B, we cross out 1 from A's domain; Arcs B → A and C → A are already on the queue.
Enforcing C → B, we cross out 1 from C's domain; add arcs A → C, B → C, and D → C to the queue.
Enforcing B → A, no domain changes are necessary (all values remaining in B's domain have a consistent corresponding value in A's domain); no arcs are added.
Enforcing C → A, we cross out 2 from C's domain; Arcs A → C, B → C, and D → C are already on the queue.
Enforcing C → D, we cross out 3 from C's domain; Arcs A → C, B → C, and D → C are already on the queue.
Enforcing E → D, no domain changes are necessary.
Enforcing A → C, no domain changes are necessary.
Enforcing B → C, no domain changes are necessary.
Enforcing D → C, we cross out 4 from D's domain (there is no c in C's domain such that $c > 4$); add arcs C → D and E → D to the queue.
Enforcing C → D, no domain changes are necessary.
Enforcing E → D, we cross out 3 from E's domain; add arc D → E to the queue.
Enforcing D → E, no domain changes are necessary.
(phew!)

Note: For a general binary CSP, to enforce arc consistency before assigning any variables, you should add all arcs to the initial queue. For this problem, it can be easily seen that if there are no unary constraints, all the arcs will be consistent before any variable is assigned a value. As a result, we can start with the unary constraints and add arcs only for the related variables after enforcing the unary constraints.

(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)

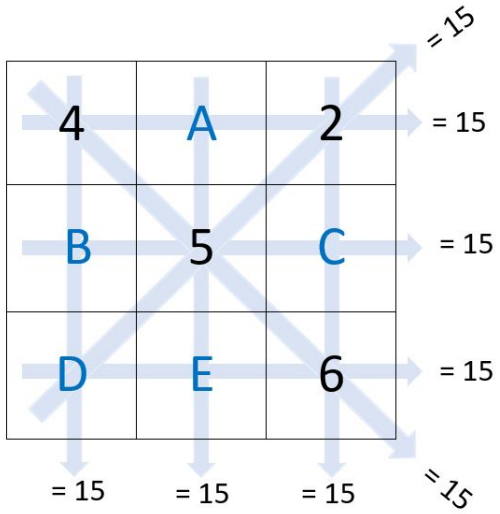
A	1	2	3	4
B	1	2	3	4
C	1	2	3	4
D	1	2	3	4
E	1	2	3	4

Answer: (B, 1), (A, 2), (C, 3), (C, 4), (D, 3), (E, 1)

4 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.

Variables: A, B, C, D, E for each empty square

Domain: $\{1, 3, 7, 8, 9\}$

Constraints:

Horizontal:

$$4 + A + 2 = 15 \implies A = 9$$
$$B + 5 + C = 15 \implies B = 10 - C$$
$$D + E + 6 = 15 \implies D = 9 - E$$

Vertical:

$$4 + B + D = 15 \implies B = 11 - D$$
$$A + 5 + E = 15 \implies A = 10 - E$$
$$2 + C + 6 = 15 \implies C = 7$$

Diagonal:

$$D + 5 + 2 = 15 \implies D = 8$$

Other:

$$A \neq B \neq C \neq D \neq E$$

(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.

Explanation of Constraint Graph: We can encode the binary constraints as edges between pairs of nodes. Note that in this problem, all rows, columns and diagonals have at most 2 variables, since at least one number is already given. Therefore, every pair of variables in the same row, column, or diagonal has the binary constraint that the sum of those two variables along with the third given number must be 15.

Here are the four binary constraints (represented by edges in the graph):

- A is in the same column as E, so A and E have a binary constraint.
- E is also in the same row as D, so E and D have a binary constraint.
- D is also in the same column as B, so D and B have a binary constraint.
- B is also in the same row as C, so B and C have a binary constraint.

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

Enforcing unary constraints (in an arbitrary order) first,

(a) We cross out 1, 3, 7, 8 from A's domain as it is already fully constrained.

(b) We cross out 1, 3, 8, 9 from C's domain as it is already fully constrained.

(c) We cross out 1, 3, 7, 9 from D's domain as it is already fully constrained.

AC3 Solution:

Enforcing arc consistency with AC-3, we have the following domain as a result:

A	~~1~~	~~3~~	~~7~~	~~8~~	9
B	~~1~~	3	~~7~~	~~8~~	~~9~~
C	~~1~~	~~3~~	7	~~8~~	~~9~~
D	~~1~~	~~3~~	~~7~~	8	~~9~~
E	1	~~3~~	~~7~~	~~8~~	~~9~~

Our starting queue will contain, in an arbitrary but convenient order, arcs E $\rightarrow$ A, B $\rightarrow$ C, B $\rightarrow$ D, and E $\rightarrow$ D.

Enforcing E $\rightarrow$ A, we cross 3, 7, 8, 9 from E's domain. Arcs A $\rightarrow$ E and D $\rightarrow$ E will be added to the queue.
Enforcing B $\rightarrow$ C, we cross 1, 7, 8, 9 from B's domain, Arcs D $\rightarrow$ B and C $\rightarrow$ B will be added to the queue.
Enforcing B $\rightarrow$ D, no domain changes are necessary.
Enforcing E $\rightarrow$ D, no domain changes are necessary.
Enforcing A $\rightarrow$ E, no domain changes are necessary.
Enforcing D $\rightarrow$ E, no domain changes are necessary.
Enforcing D $\rightarrow$ B, no domain changes are necessary.
Enforcing C $\rightarrow$ B, no domain changes are necessary.

5 MRV and LCV in Action

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

- Theo = Red, Yellow, Green
- Shruti = Red, Yellow, Pink
- Roy = 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.

Theo 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. Shruti isn't fully convinced yet that LCV and MRV will help speed up a constraint satisfaction problem, so Theo asks for your help to convince Shruti.

(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.

We will backtrack zero times.

We first use the MRV (minimum remaining values) heuristic to choose which TA gets assigned next. Roy has 2 possible paint colors, while Theo and Shruti have 3. Since Roy has the fewest possible paint colors left, we will assign her next.

We will assign Roy a paint color using the LCV heuristic.

If we assign Roy to "Pink," the sum of the number of paint colors remaining over the other TAs is 5.

- Theo = Red, Yellow, Green
- Shruti = Red, Yellow, ~~Pink~~
- Roy = ~~Red~~, **Pink**

If we assign Roy to "Red," the sum of the number of paint colors remaining over the other TAs is 4.

- Theo = ~~Red~~, Yellow, Green
- Shruti = ~~Red~~, Yellow, Pink
- Roy = **Red**, ~~Pink~~

Since 5 ¿ 4, assigning Roy to "Pink" is the least constraining value (or paint color).

Now, we will use MRV again to choose which TA will get assigned next. Shruti has 2 possible paint colors, while Theo has 3. Since Shruti has the fewest possible paint colors left, we will assign her next.

If we assign Shruti "Red", there are 2 paint colors left for Theo.

- Theo = ~~Red~~, Yellow, Green

- Shruti = **Red**, ~~Yellow~~, ~~Pink~~
- Roy = ~~Red~~, **Pink**

If we assign Shruti "Yellow", there are 2 paint colors left for Theo.

- Theo = Red, ~~Yellow~~, Green
- Shruti = ~~Red~~, **Yellow**, ~~Pink~~
- Roy = ~~Red~~, **Pink**

Since we have a tie (and we break ties in rainbow order), we will assign Shruti to "Red."

Theo is the only TA left without a paint color assignment. We can assign her either "Yellow" or "Green," but choose "Yellow" because we break ties in rainbow order.

We have satisfied the constraints (that no two TAs bring the same paint color) without backtracking to previous assignments.

(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?

We will backtrack one time.

We first use the maximum remaining values heuristic to choose which TA gets assigned next. Roy has 2 possible paint colors, while Theo and Shruti have 3. Since Theo and Shruti have the most possible paint colors left, we will assign Theo next (tiebreaking by order).

We will assign Theo a paint color using the most constraining value heuristic.

If we assign Theo to "Red," the sum of the number of paint colors remaining over the other TAs is 3.

- Theo = **Red**, ~~Yellow~~, ~~Green~~
- Shruti = ~~Red~~, Yellow, Pink
- Roy = ~~Red~~, Pink

If we assign Theo to "Yellow," the sum of the number of paint colors remaining over the other TAs is 4.

- Theo = ~~Red~~, **Yellow**, ~~Green~~
- Shruti = Red, ~~Yellow~~, Pink
- Roy = Red, Pink

If we assign Theo to "Green," the sum of the number of paint colors remaining over the other TAs is 5.

- Theo = ~~Red~~, ~~Yellow~~, **Green**
- Shruti = Red, Yellow, Pink
- Roy = Red, Pink

Since $3 < 4$ and $3 < 5$, we will assign Theo to "Red" using the most constraining value heuristic.

Now, we will use the *maximum* remaining values heuristic again to choose which TA gets assigned next. Shruti has 2 possible paint colors, while Roy has 1. Since Shruti has more possible paint colors left, we will assign her next.

If we assign Shruti to "Yellow," the sum of the number of paint colors remaining over the other TAs is 1.

- Theo = **Red**, ~~Yellow~~, ~~Green~~
- Shruti = ~~Red~~, **Yellow**, ~~Pink~~
- Roy = ~~Red~~, Pink

If we assign Shruti to "Pink," the sum of the number of paint colors remaining over the other TAs is 0.

- Theo = **Red**, ~~Yellow~~, ~~Green~~
- Shruti = ~~Red~~, ~~Yellow~~, **Pink**
- Roy = ~~Red~~, ~~Pink~~

Since $0 < 1$, we will assign Shruti to "Pink" using the *most* constraining value heuristic.

Now, Roy has no paint colors to choose from, so we have not found a solution to our CSP.

Thus, we must backtrack and assign Shruti to "Yellow" instead of "Pink."

- Theo = **Red**, ~~Yellow~~, ~~Green~~
- Shruti = ~~Red~~, **Yellow**, ~~Pink~~
- Roy = ~~Red~~, Pink

Roy is the only TA left without a paint color assignment, and there is only one paint color left in her domain. Thus, we can assign Roy to "Pink."

We have satisfied the constraints (that no two TAs bring the same paint color), but we had to backtrack to a previous assignment one time.