Cost-Based Search as IP
Motivation

- Many problems can be solved by search (e.g., backtracking, branch and bound, etc.) but we haven’t seen anything on the other direction
- IP is a very expressive representation
Formulating Search as IP
Formulating Search as IP

Variables:
Formulating Search as IP

**Variables:** binary variable for each edge in the graph, representing whether the edge is in the final path or not (0 means edge is not in the final path, 1 means edge is in the final path)
Formulating Search as IP

Variables: binary variable for each edge in the graph, representing whether the edge is in the final path or not (0 means edge is not in the final path, 1 means edge is in the final path)

Ex: \( x_{X \rightarrow Y} \) is a binary variable representing whether the edge \( X \rightarrow Y \) is in the final path
Formulating Search as IP

How to represent the path \( S \rightarrow A \rightarrow C \rightarrow G \)?
Formulating Search as IP

How to represent the path $S \rightarrow A \rightarrow C \rightarrow G$?

3 edges: \{S→A, A→C, C→G\}

$x_{S\rightarrow A} =$ indicator for whether $S \rightarrow A$ is in the path, etc (same for every path in our graph)
Formulating Search as IP

How to represent the path S→A→C→G?
3 edges: {S→A, A→C, C→G}
x_{S→A} = indicator for whether S→A is in the path, etc (same for every path in our graph)

\[(x_{S→A} = 1 \quad x_{S→B} = 0 \quad x_{A→B} = 0 \quad x_{A→C} = 1 \quad x_{B→C} = 0 \quad x_{B→G} = 0 \quad x_{C→S} = 0 \quad x_{C→G} = 1 \quad x_{G→C} = 0)\]
Formulating Search as IP

How to represent the path $S \rightarrow A \rightarrow C \rightarrow G$?
3 edges: $\{S \rightarrow A, A \rightarrow C, C \rightarrow G\}$

$x_{S \rightarrow A}$ = indicator for whether $S \rightarrow A$ is in the path, etc (same for every path in our graph)

$$(x_{S \rightarrow A} = 1 \quad x_{S \rightarrow B} = 0 \quad x_{A \rightarrow B} = 0 \quad x_{A \rightarrow C} = 1 \quad x_{B \rightarrow C} = 0 \quad x_{B \rightarrow G} = 0 \quad x_{C \rightarrow S} = 0 \quad x_{C \rightarrow G} = 1 \quad x_{G \rightarrow C} = 0)$$

9-tuple: $(1, 0, 0, 1, 0, 0, 0, 1, 0)$
Order: $x_{S \rightarrow A}$, $x_{S \rightarrow B}$, $x_{A \rightarrow B}$, $x_{A \rightarrow C}$, $x_{B \rightarrow C}$, $x_{B \rightarrow G}$, $x_{C \rightarrow S}$, $x_{C \rightarrow G}$, $x_{G \rightarrow C}$
a) i) 9-tuple representation for $S \rightarrow A \rightarrow B \rightarrow C \rightarrow G$
a) i) 9-tuple representation for $S \rightarrow A \rightarrow B \rightarrow C \rightarrow G$

Order: $x_{S \rightarrow A}, x_{S \rightarrow B}, x_{A \rightarrow B}, x_{A \rightarrow C}, x_{B \rightarrow C}, x_{B \rightarrow G}, x_{C \rightarrow S}, x_{C \rightarrow G}, x_{G \rightarrow C}$
a) i) 9-tuple representation for $S \rightarrow A \rightarrow B \rightarrow C \rightarrow G$

$(1, 0, 1, 0, 1, 0, 0, 1, 0)$
a) i) 9-tuple representation for $S \rightarrow A \rightarrow B \rightarrow C \rightarrow G$

$(1, 0, 1, 0, 1, 0, 0, 1, 0)$

ii) 9-tuple representation for $A \rightarrow C \rightarrow S \rightarrow B$
a) i) 9-tuple representation for $S \rightarrow A \rightarrow B \rightarrow C \rightarrow G$
   $(1, 0, 1, 0, 1, 0, 0, 1, 0)$
ii) 9-tuple representation for $A \rightarrow C \rightarrow S \rightarrow B$
   $(0, 1, 0, 1, 0, 0, 1, 0, 0)$
a) i) 9-tuple representation for $S\rightarrow A\rightarrow B\rightarrow C\rightarrow G$
   $(1, 0, 1, 0, 1, 0, 0, 1, 0)$
   
   ii) 9-tuple representation for $A\rightarrow C\rightarrow S\rightarrow B$
   $(0, 1, 0, 1, 0, 0, 1, 0, 0)$
   
   iii) Path that corresponds to $(0, 0, 1, 0, 1, 0, 0, 0, 0)$
a) i) 9-tuple representation for $S \rightarrow A \rightarrow B \rightarrow C \rightarrow G$
   \[(1, 0, 1, 0, 1, 0, 0, 1, 0)\]
ii) 9-tuple representation for $A \rightarrow C \rightarrow S \rightarrow B$
    \[(0, 1, 0, 1, 0, 0, 1, 0, 0)\]
iii) Path that corresponds to \((0, 0, 1, 0, 1, 0, 0, 0, 0)\)
    $A \rightarrow B \rightarrow C$
Constraints:
Constraints: need to make sure paths are valid
Constraints: need to make sure paths are valid
1) Ensure path starts at S
**Constraints:** need to make sure paths are valid

1) Ensure path starts at S
2) Ensure path ends at G
Constraint 1: path starts at S
Constraint 1: path starts at S
Two nodes going out of S: A and B
Constraint 1: path starts at S
Two nodes going out of S: A and B → either \( x_{S \rightarrow A} \) or \( x_{S \rightarrow B} \) must be 1
Constraint 1: path starts at S
Two nodes going out of S: A and B → either $x_{S\rightarrow A}$ or $x_{S\rightarrow B}$ must be 1
$x_{S\rightarrow A} + x_{S\rightarrow B} = 1$
Constraint 1: path starts at S
Two nodes going out of S: A and B → either $x_{S \rightarrow A}$ or $x_{S \rightarrow B}$ must be 1

$x_{S \rightarrow A} + x_{S \rightarrow B} = 1$

Inequality form: $x_{S \rightarrow A} + x_{S \rightarrow B} \leq 1$ and $-x_{S \rightarrow A} - x_{S \rightarrow B} \leq -1$
Constraint 1: path starts at S
Two nodes going out of S: A and B → either $x_{S \rightarrow A}$ or $x_{S \rightarrow B}$ must be 1

$x_{S \rightarrow A} + x_{S \rightarrow B} = 1$

Inequality form: $x_{S \rightarrow A} + x_{S \rightarrow B} \leq 1$ and $-x_{S \rightarrow A} - x_{S \rightarrow B} \leq -1$

One node going into S: C
**Constraint 1:** path starts at S
Two nodes going out of S: A and B → either $x_{S \rightarrow A}$ or $x_{S \rightarrow B}$ must be 1
\[ x_{S \rightarrow A} + x_{S \rightarrow B} = 1 \]

Inequality form: $x_{S \rightarrow A} + x_{S \rightarrow B} \leq 1$ and $-x_{S \rightarrow A} - x_{S \rightarrow B} \leq -1$

One node going into S: C
\[ x_{C \rightarrow S} = 0 \]
Constraint 1: path starts at S
Two nodes going out of S: A and B → either $x_{S\rightarrow A}$ or $x_{S\rightarrow B}$ must be 1

$$x_{S\rightarrow A} + x_{S\rightarrow B} = 1$$

Inequality form: $x_{S\rightarrow A} + x_{S\rightarrow B} \leq 1$ and $-x_{S\rightarrow A} - x_{S\rightarrow B} \leq -1$

One node going into S: C

$$x_{C\rightarrow S} \leq 0$$ and $$-x_{C\rightarrow S} \leq 0$$
Constraint 1: path starts at S
Two nodes going out of S: A and B → either $x_{S \rightarrow A}$ or $x_{S \rightarrow B}$ must be 1

$x_{S \rightarrow A} + x_{S \rightarrow B} = 1$

Inequality form: $x_{S \rightarrow A} + x_{S \rightarrow B} \leq 1$ and $-x_{S \rightarrow A} - x_{S \rightarrow B} \leq -1$

One node going into S: C

$x_{C \rightarrow S} \leq 0$ and $-x_{C \rightarrow S} \leq 0$
Constraint 2: path ends at G
Constraint 2: path ends at G

Two nodes going into G: C and B → either $x_{C\rightarrow G}$ or $x_{B\rightarrow G}$ must be 1

$x_{C\rightarrow G} + x_{B\rightarrow G} \leq 1$ and $-x_{C\rightarrow G} - x_{B\rightarrow G} \leq -1$
Constraint 2: path ends at G
Two nodes going into G: C and B → either $x_{C\rightarrow G}$ or $x_{B\rightarrow G}$ must be 1

$$x_{C\rightarrow G} + x_{S\rightarrow B} \leq 1 \quad \text{and} \quad -x_{C\rightarrow G} - x_{B\rightarrow G} \leq -1$$

One node coming out of G: C → $x_{G\rightarrow C}$ must be 0

$$x_{G\rightarrow C} \leq 0 \quad \text{and} \quad -x_{G\rightarrow C} \leq 0$$
Constraint 2: path ends at G
Two nodes going into G: C and B → either $x_{C\rightarrow G}$ or $x_{B\rightarrow G}$ must be 1

$x_{C\rightarrow G} + x_{B\rightarrow G} \leq 1$ and $-x_{C\rightarrow G} - x_{B\rightarrow G} \leq -1$

One node coming out of G: C → $x_{G\rightarrow C}$ must be 0

$x_{G\rightarrow C} \leq 0$ and $-x_{G\rightarrow C} \leq 0$
Constraints: need to make sure paths are valid
1) Ensure path starts at S - done
2) Ensure path ends at G - done
Constraints: need to make sure paths are valid
1) Ensure path starts at S - done
2) Ensure path ends at G - done

These two constraints are not enough :(
Constraints: need to make sure paths are valid
1) Ensure path starts at S - done
2) Ensure path ends at G - done

Question: 9-tuple that satisfies these constraints but does not represent a valid path from S to G
**Constraints:** need to make sure paths are valid
1) Ensure path starts at S - **done**
2) Ensure path ends at G - **done**

**Question:** 9-tuple that satisfies these constraints but does **not** represent a valid path from S to G
{S→A, C→G}: (1, 0, 0, 0, 0, 0, 0, 1, 0)
More constraints: ensure all other nodes are non-terminal (not start or goal)
More constraints: ensure all other nodes are non-terminal (not start or goal)
- Path can only pass through each non-terminal node at most once
More constraints: ensure all other nodes are non-terminal (not start or goal)
- Path can only pass through each non-terminal node at most once

Constraint that node B can only appear on the path at most once:
More constraints: ensure all other nodes are non-terminal (not start or goal)

- Path can only pass through each non-terminal node at most once

Constraint that node B can only appear on the path at most once:
Two nodes going into B: S, A
More constraints: ensure all other nodes are non-terminal (not start or goal)
- Path can only pass through each non-terminal node at most once

Constraint that node B can only appear on the path at most once:
Two nodes going into B: S, A → either $x_{S\rightarrow B}$ or $x_{A\rightarrow B}$ must be 1, but both cannot be 1
More constraints: ensure all other nodes are non-terminal (not start or goal)
- Path can only pass through each non-terminal node at most once

Constraint that node B can only appear on the path at most once:
Two nodes going into B: S, A → either $x_{S \rightarrow B}$ or $x_{A \rightarrow B}$ must be 1, but both cannot be 1

$$x_{S \rightarrow B} + x_{A \rightarrow B} \leq 1$$
More constraints: ensure all other nodes are non-terminal (not start or goal)

- Path can only pass through each non-terminal node at most once

Constraint that node B can only appear on the path at most once:
Two nodes going into B: S, A → either $x_{S \rightarrow B}$ or $x_{A \rightarrow B}$ must be 1, but both cannot be 1
$$x_{S \rightarrow B} + x_{A \rightarrow B} \leq 1$$

Two nodes coming out of B: C, G → either $x_{B \rightarrow C}$ or $x_{B \rightarrow G}$ must be 1, but both cannot be 1
More constraints: ensure all other nodes are non-terminal (not start or goal)

- Path can only pass through each non-terminal node at most once

Constraint that node B can only appear on the path at most once:
Two nodes going into B: S, A → either \( x_{S\to B} \) or \( x_{A\to B} \) must be 1, but both cannot be 1
\[ x_{S\to B} + x_{A\to B} \leq 1 \]

Two nodes coming out of B: C, G → either \( x_{B\to C} \) or \( x_{B\to G} \) must be 1, but both cannot be 1
\[ x_{B\to C} + x_{B\to G} \leq 1 \]
More constraints: If there is an edge to B, then there must be an edge out of B (otherwise, B is either a dead end or a start)
More constraints: If there is an edge to B, then there must be an edge out of B (otherwise, B is either a dead end or a start)

Idea: number of edges into B = number of edges out of B (we already constrained that you can only have one of those edges)
More constraints: If there is an edge to B, then there must be an edge out of B (otherwise, B is either a dead end or a start)

Idea: number of edges into B = number of edges out of B (we already constrained that you can only have one of those edges)

\[ x_{S \rightarrow B} + x_{A \rightarrow B} = x_{B \rightarrow C} + x_{B \rightarrow G} \]
More constraints: If there is an edge to $B$, then there must be an edge out of $B$ (otherwise, $B$ is either a dead end or a start)

Idea: number of edges into $B = $ number of edges out of $B$ (we already constrained that you can only have one of those edges)

$x_{S \rightarrow B} + x_{A \rightarrow B} \leq x_{B \rightarrow C} + x_{B \rightarrow G}$

$x_{S \rightarrow B} + x_{A \rightarrow B} \geq x_{B \rightarrow C} + x_{B \rightarrow G}$
More constraints: If there is an edge to B, then there must be an edge out of B (otherwise, B is either a dead end or a start)

Idea: number of edges into B = number of edges out of B (we already constrained that you can only have one of those edges)

\[ x_{S \rightarrow B} + x_{A \rightarrow B} - x_{B \rightarrow C} - x_{B \rightarrow G} \leq 0 \]
\[-x_{S \rightarrow B} - x_{A \rightarrow B} + x_{B \rightarrow C} + x_{B \rightarrow G} \leq 0 \]
Objective function:
Objective function:
Idea: coefficient for each edge is the cost of that edge
Objective function:
Idea: coefficient for each edge is the cost of that edge

$$3x_{S\rightarrow A} + 7x_{S\rightarrow B} + 5x_{A\rightarrow B} + 4x_{A\rightarrow C} + 2x_{B\rightarrow C} + 1x_{B\rightarrow G} + 1x_{C\rightarrow S} + 6x_{C\rightarrow G} + 6x_{G\rightarrow C}$$
Still not enough to ensure a valid path :(
Still not enough to ensure a valid path :( 

Counterexample:
Still not enough to ensure a valid path :(

Counterexample:

Idea: anything with a loop outside the path is still allowed by our constraints
How can we fix this?
How can we fix this?
Answer: we don’t have to :)

How can we fix this?
Answer: we don’t have to :)
How can we fix this?

**Answer:** we don’t have to :)

**Idea:** If we have an extra cycle, that would just increase the total path cost. Because we are trying to minimize cost, this would only hurt us, so we wouldn’t return such a solution anyway.
Cost-Based Search as IP

- Now let’s put everything together, and define the following search algorithm
  - First convert the search problem into the IP representation
  - Then run an IP-solver (which is complete and optimal) on the representation
  - Reconstruct the path from start to goal by getting all the ones in the variables

- Is this is complete?
- Is this is optimal?
Cost-Based Search as IP

- Now let’s put everything together, and define the following search algorithm
  - First convert the search problem into the IP representation
  - Then run an IP-solver (which is complete and optimal) on the representation
  - Reconstruct the path from start to goal by getting all the ones in the variables

- Is this is complete? Yes
- Is this is optimal?
Cost-Based Search as IP

- Now let’s put everything together, and define the following search algorithm
  - First convert the search problem into the IP representation
  - Then run an IP-solver (which is complete and optimal) on the representation
  - Reconstruct the path from start to goal by getting all the ones in the variables

- Is this is complete? Yes
- Is this is optimal? Yes
Take Home Messages

- Cost-based search can be expressed, and solved with IP
- IP is very expressive, we can do many interesting things with it

- Want some more?
  
  Minimax as IP!!!  (Bonus question on the course website)