Topics in Reinforcement Learning: Lessons from AlphaZero for (Sub)Optimal Control and Discrete Optimization

Arizona State University
Course CSE 691, Spring 2022

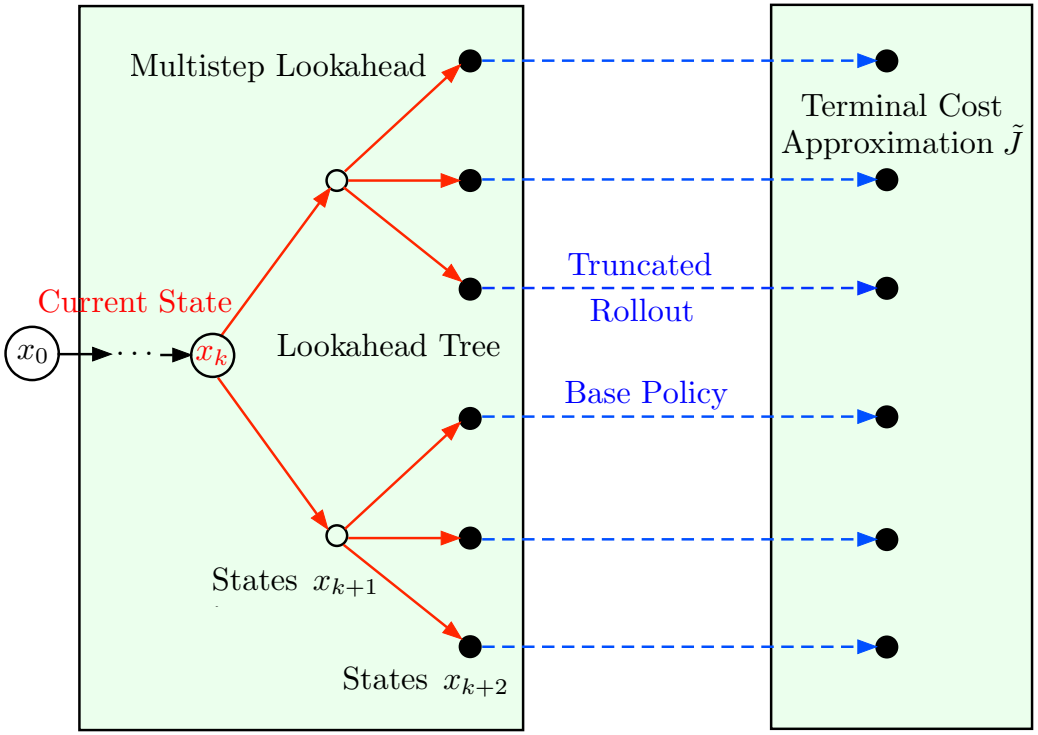
Links to Class Notes, Videolectures, and Slides at
http://web.mit.edu/dimitrib/www/RLbook.html

Dimitri P. Bertsekas
dbertsek@asu.edu

Lecture 6
Deterministic Problems: Multistep Approximation in Value Space, Constrained Rollout, Rollout for Discrete Optimization

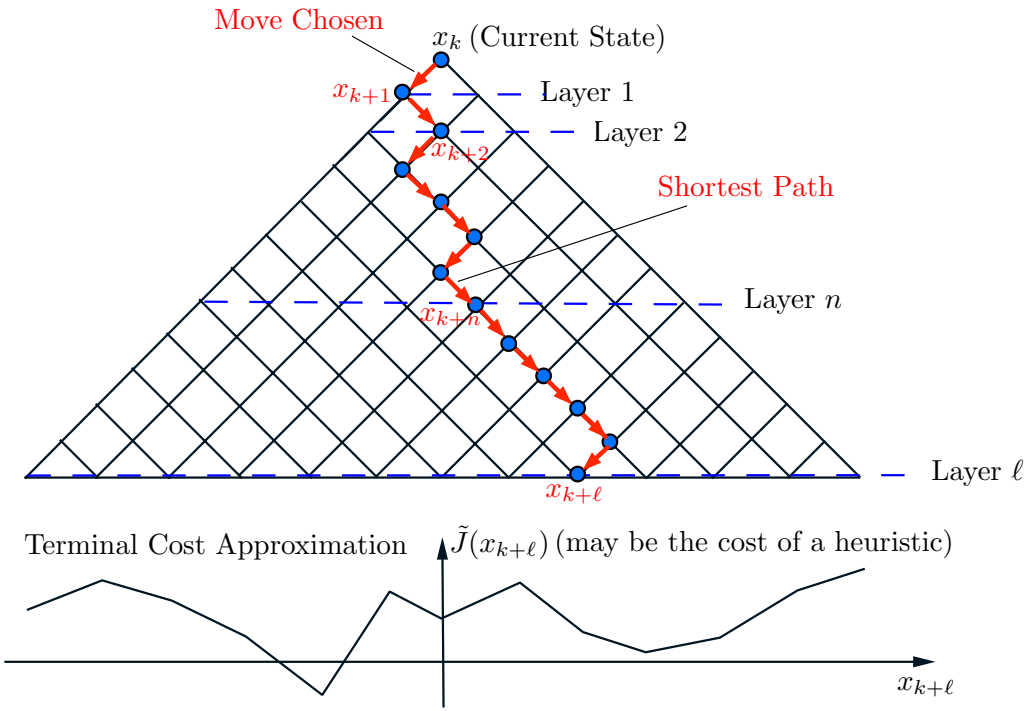
Outline

Multistep Approximation in Value Space - The General Case



- Special case: No rollout. The general multistep approximation in value space scheme.
- Special case: Pure multistep rollout. No terminal cost and no truncation.
- WE TAKE IT AS FACT: Longer lookahead improves performance (but is costly).
- OUR STRATEGY: Extend the lookahead as much as the comp. budget allows.
- One idea: Truncated rollout (a cheap extension of the lookahead length).
- Another computation-saving idea: Selectively prune the lookahead tree.

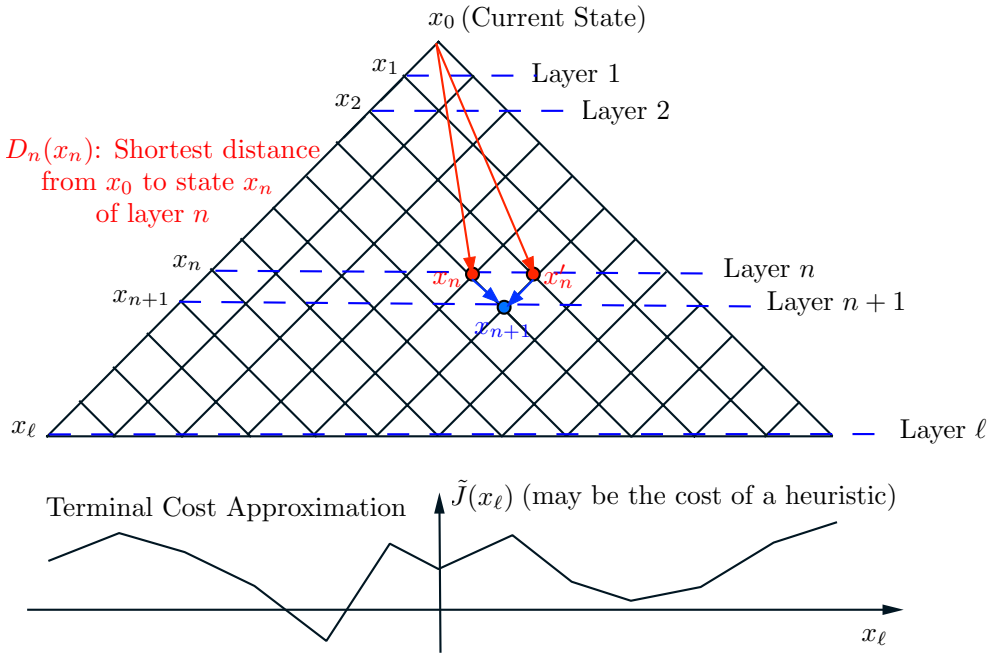
Multistep Lookahead in Deteministic Problems



We obtain a trajectory $\{x_k, x_{k+1}, \ldots, x_{k+\ell}\}$ that minimizes the shortest distance from x_k to $x_{k+\ell}$ PLUS $\tilde{J}(x_{k+\ell})$. We then use the first move $x_k \to x_{k+1}$.

- All the shortest path problems from x_k to $x_{k+\ell}$ can be solved simultaneously by backward DP (start from layer ℓ go towards x_k).
- An important alternative is the forward DP algorithm.
- It is the same as the backwards DP algorithm with the direction of the arcs reversed (start from x_k go towards layer ℓ - see the next slide).

Forward DP Algorithm and Iterative Deepening [$\tilde{J}(x)$ is given for all x]

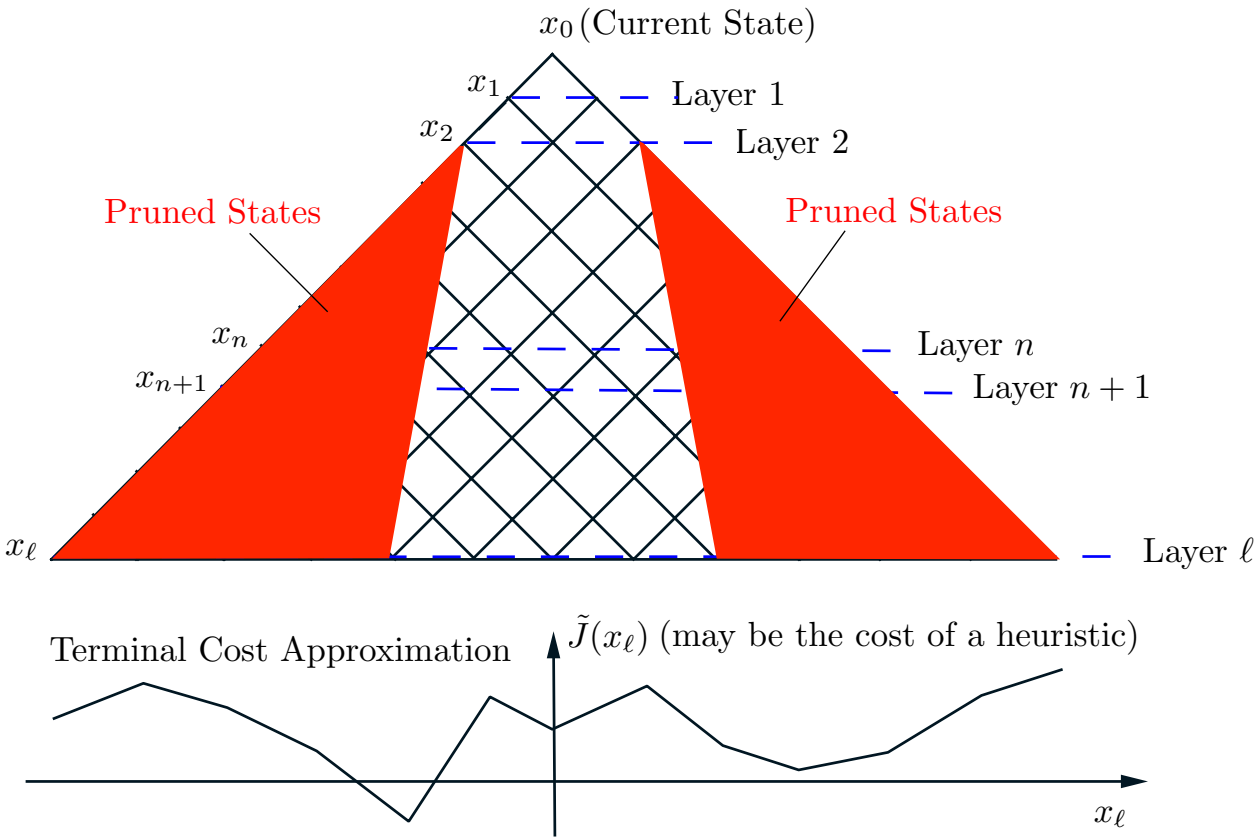


- The "forward" DP algorithm: The shortest distances $D_{n+1}(x_{n+1})$ to layer $n+1$ states are obtained from the shortest distances $D_n(x_n)$ to layer n states as follows:

$$D_{n+1}(x_{n+1}) = \min_{x_n} \Big[(\text{Cost } x_n \to x_{n+1}) + D_n(x_n) \Big]$$

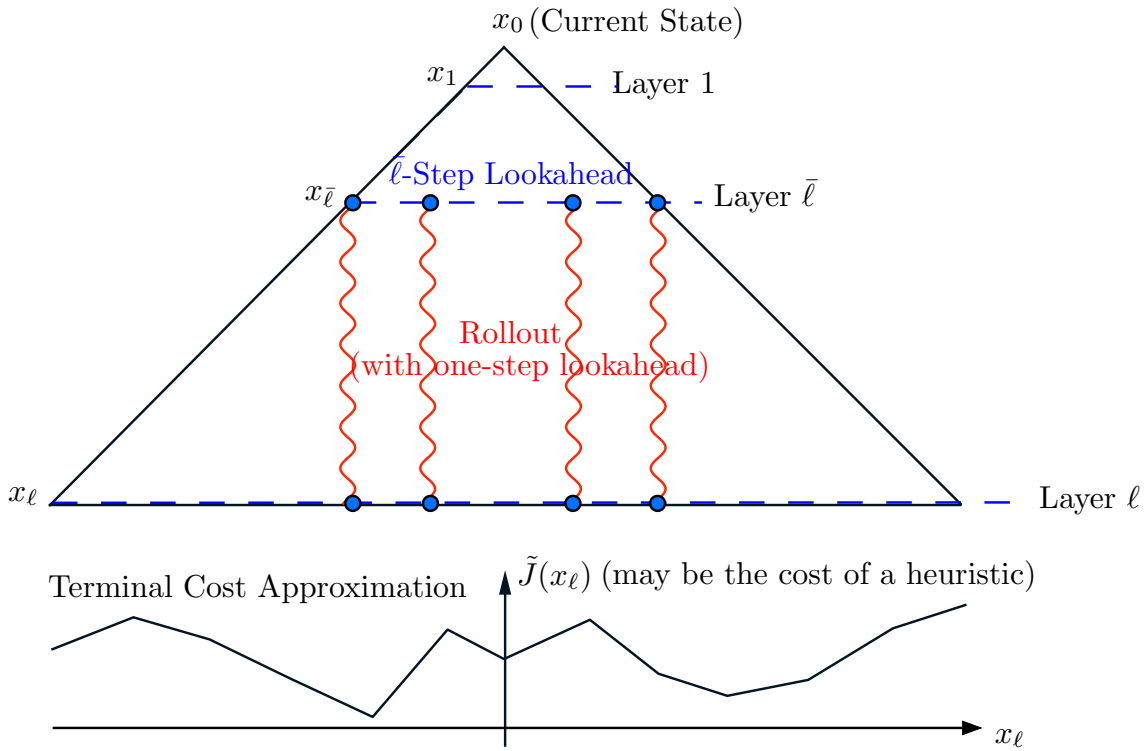
- Solution of the ℓ-step lookahead problem: The shortest path to the state x_ℓ^* of layer ℓ that minimizes $D_\ell(x_\ell) + \tilde{J}(x_\ell)$.
- Iterative deepening: Solve the n-step lookahead problem before solving the $(n+1)$-step lookahead problem.
- This is an "anytime" algorithm (returns a feasible solution even if it is interrupted).

Iterative Deepening with Tree Pruning



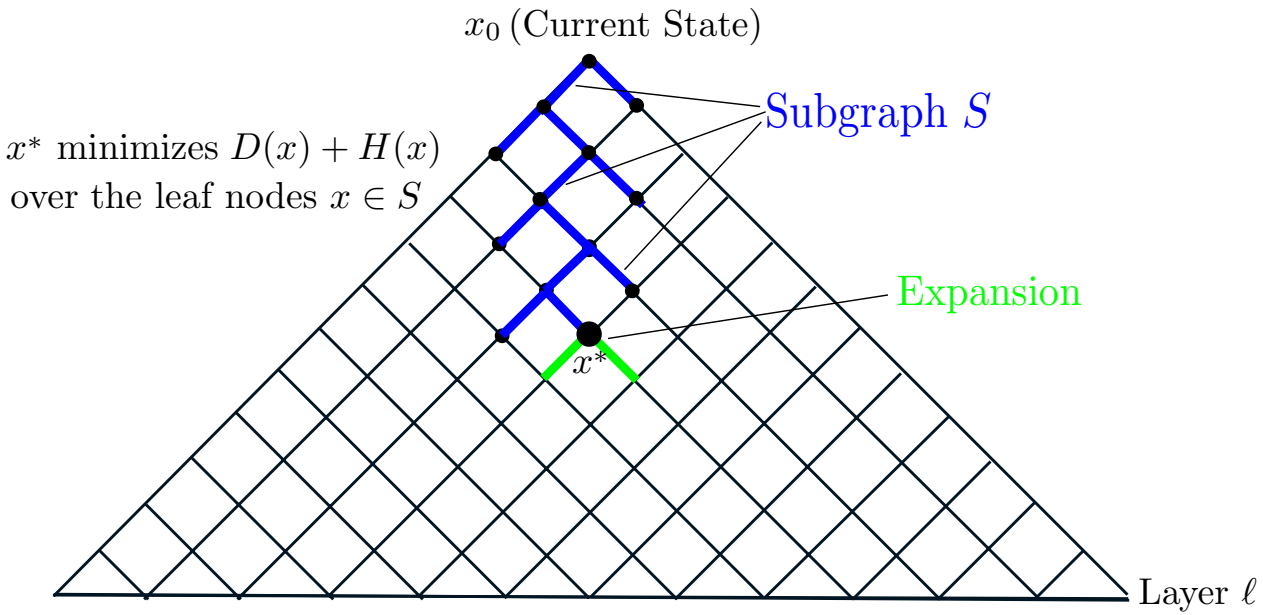
- Iterative deepening can be “enhanced" by pruning states $\hat{x}_n$ such that the n-step lookahead cost $D_n(\hat{x}_n) + \tilde{J}(\hat{x}_n)$ is “far from the minimum" over x_n.
- We prune as we go: Prune states in layer n before pruning states in layer $n+1$.
- Runs the risk of overpruning: Some pruned states may be “good" in hindsight.
- Should we go back and check for overpruning? How?

An Alternative Form of Pruning - Double Rollout



- Instead of solving the ℓ-step lookahead shortest path problem by iterative deepening and pruning, solve it approximately using rollout as in the figure.
- The base heuristic used from layer $\bar{\ell}$ to layer ℓ need not be related to the terminal cost approximation $\tilde{J}(x_\ell)$.
- For $\bar{\ell} = 0$ we obtain one-step lookahead truncated rollout. For $\bar{\ell} = \ell - 1$ we obtain ℓ-step lookahead approximation in value space (in effect no rollout).
- Variants of double rollout: Simplified, fortified, parallel, iterative deepening, etc.

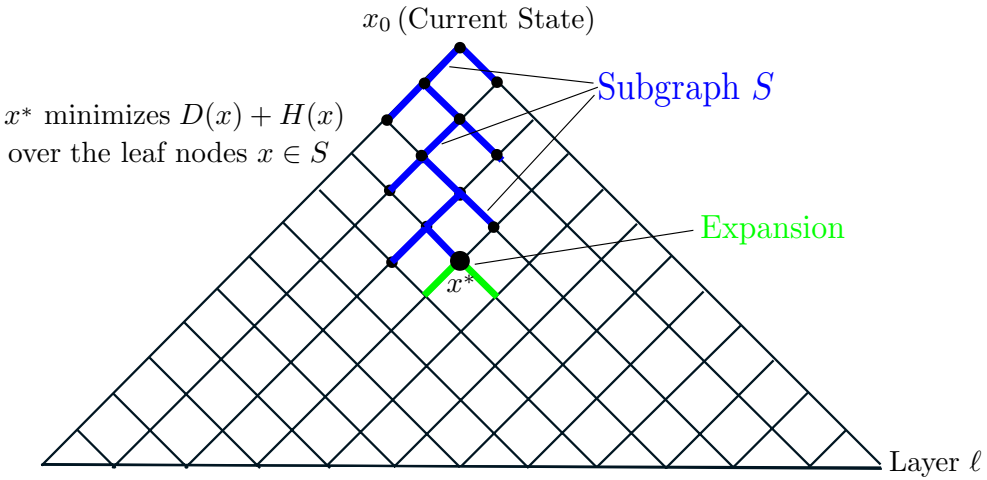
Incremental Multistep Rollout - Flexible Pruning/Iterative Deepening



Difference from double rollout: In place of the graph of the first $\bar{\ell}$ layers, we use a less regular graph, which is expanded at each iteration based on a shortest path computation.

- At the start of an iteration, we have an acyclic connected subgraph S rooted at x_0.
- We compute the shortest distance $D(x)$ from x_0 to all $x \in S$, going through S.
- We find a leaf node $x^* \in S$ that minimizes $D(x) + H(x)$, where $H(x)$ is a "heuristic distance" from x to layer ℓ.
- Expand x^* to enlarge S and start the next iteration (or stop if x^* is in layer ℓ).

Incremental Multistep Rollout - Some Details



- At the start of an iteration, we have an acyclic connected subgraph S rooted at x_0.
- We minimize $D(x) + H(x)$ over all leaf nodes $x \in S$.
- We expand the minimizing node x^* to form the new subgraph.
- Example of $H(x)$: The cost of a base heuristic that starts from x and ends at some node x_ℓ of layer ℓ, plus $\tilde{J}(x_\ell)$, plus an extra term that favors paths with few hops; e.g., $\delta \cdot$ (number of hops from x_0 to x), where $\delta > 0$.
- The computation of the shortest distances $D(x)$ is done progressively with the forward DP algorithm as the subgraph S expands.
- Note: The δ term allows the algorithm to "backtrack."
- For $\delta = 0$, we get max pruning: S ends up being "long and skinny". For $\delta \approx \infty$, we get min pruning: S ends up being as "fat" as possible.

Break Time

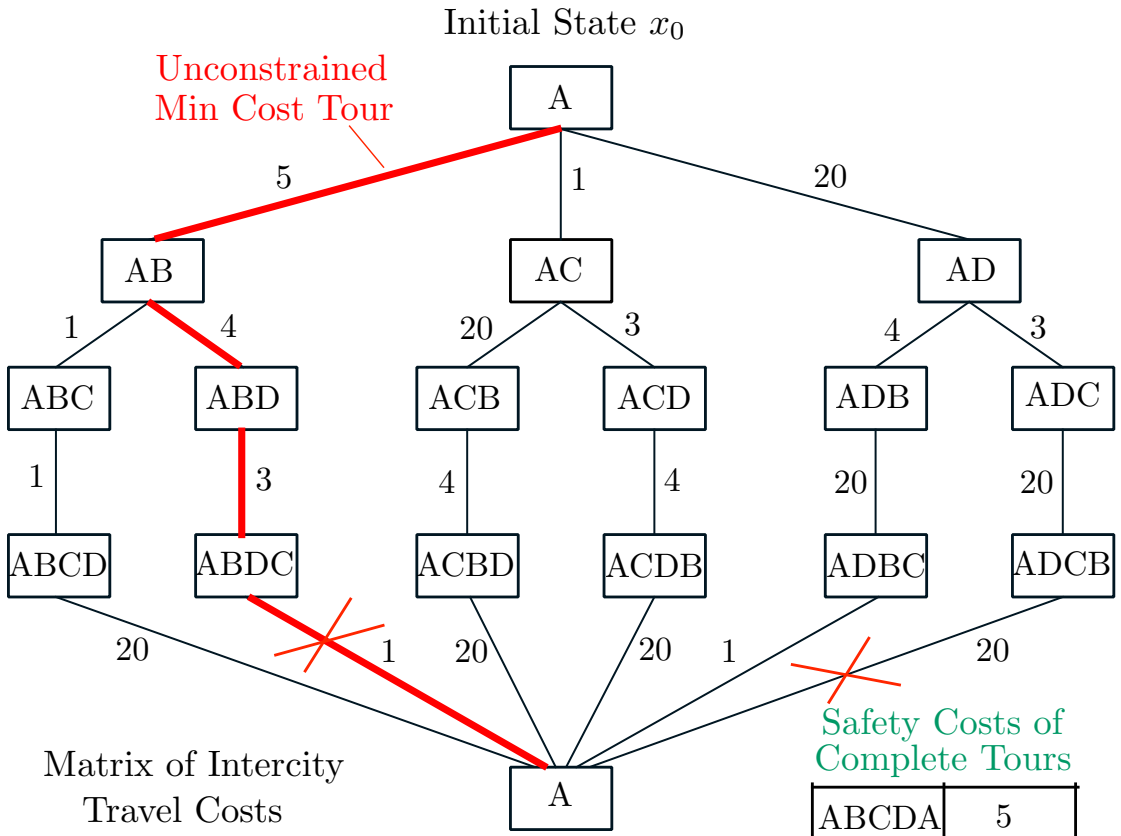
A Ten-Minute Break

Constrained Rollout - Main Ideas

Applies to problems with additional constraints on the entire optimal trajectory

- Greatly expands the range of applications of rollout
- For example it applies to intractable discrete optimization problems (e.g., shortest path problems with a limit on the number of hops).
- It is similar to unconstrained rollout: As we expand the rollout path, we exclude from consideration the Q-factors that correspond to constraint violation.
- Guarantees cost improvement over the base heuristic under appropriate conditions (modified versions of sequential consistency, sequential improvement, or use of a fortified version).

Traveling Salesman: Example of a Trajectory Constraint



Matrix of Intercity Travel Costs

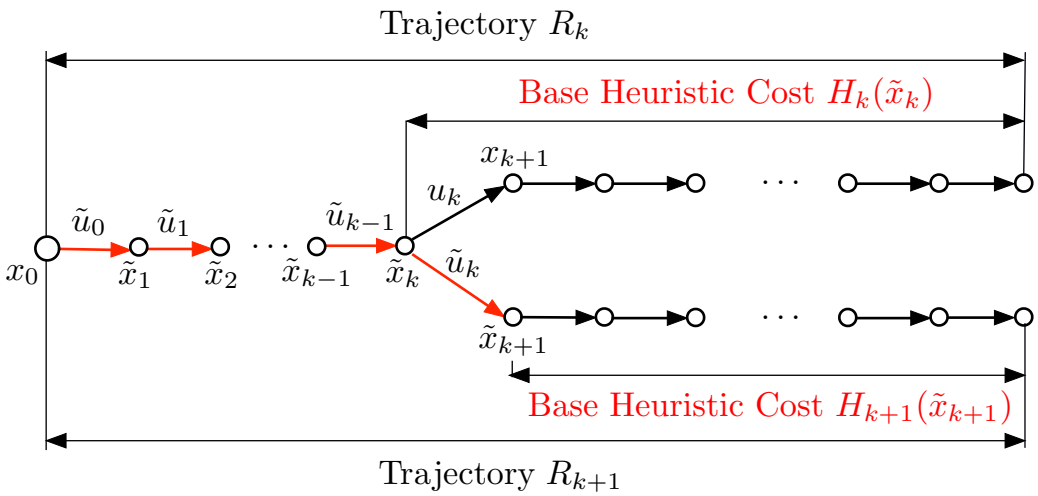
	5	1	20
20		1	4
1	20		1
20	4	3	

Safety Costs of Complete Tours

ABCDA	5
ABDCA	20
ACBDA	4
ACDBA	3
ADBCA	1
ADCBA	15

Find a minimum cost tour subject to a safety constraint

Deterministic Rollout with Trajectory Constraint: Basic Idea



Review of the unconstrained rollout algorithm:

- Construct sequence of trajectories $\{T_0, T_1, \ldots, T_N\}$ with monotonically nonincreasing cost (assuming a sequential improvement condition).
- For each k, the trajectories $T_k, T_{k+1}, \ldots, T_N$ share the same initial portion $(x_0, \tilde{u}_0, \ldots, \tilde{u}_{k-1}, \tilde{x}_k)$.
- The base heuristic is used to generate candidate trajectories that correspond to the controls $u_k \in U_k(x_k)$.
- The next trajectory T_{k+1} is the candidate trajectory that has min cost.

To deal with a trajectory constraint $T \in C$, we discard all the candidate trajectories that violate the constraint, and we choose T_{k+1} to be the best of the remaining trajectories.

Deterministic Problems with Constraints: Definition

- Consider a deterministic optimal control problem with system $x_{k+1} = f_k(x_k, u_k)$.
- A complete trajectory is a sequence

$$T = (x_0, u_0, x_1, u_1, \ldots, u_{N-1}, x_N)$$

- Problem:

$$\min_{T \in C} G(T)$$

where G is a given cost function and C is a given constraint set of trajectories.

State augmentation idea for rollout

- Redefine the state to be the partial trajectory

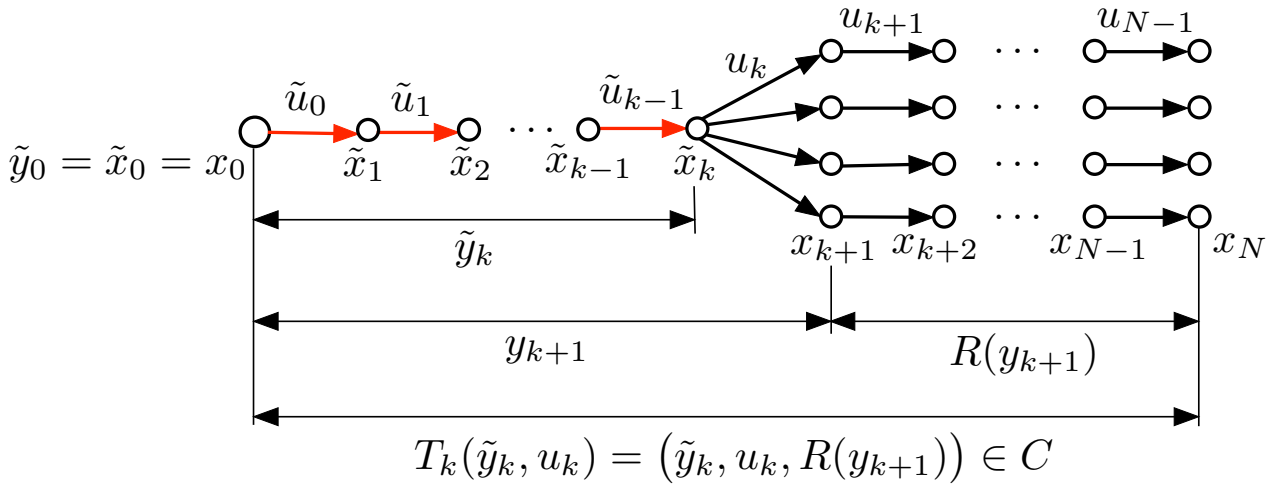
$$y_k = (x_0, u_0, x_1, \ldots, u_{k-1}, x_k)$$

- Partial trajectory evolves according to a redefined system equation:

$$y_{k+1} = \big(y_k, u_k, f_k(x_k, u_k)\big)$$

- The problem becomes to minimize $G(y_N)$ subject to the constraint $y_N \in C$.

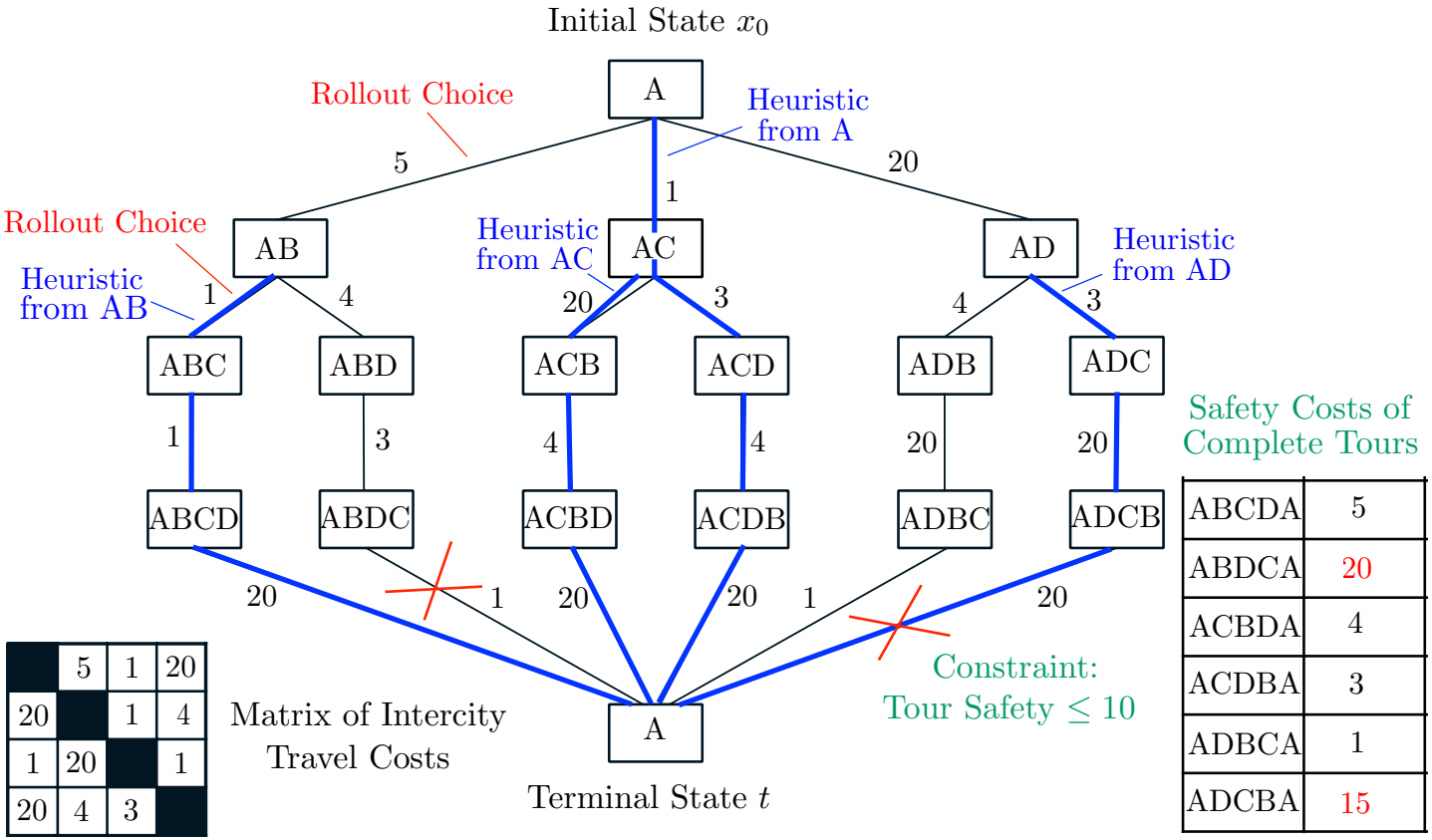
Rollout Algorithm - Partial Trajectory-Dependent Base Heuristic



- Given $\tilde{y}_k = \{\tilde{x}_0, \tilde{u}_0, \tilde{x}_1, \tilde{u}_1, \ldots, \tilde{u}_{k-1}, \tilde{x}_k\}$ consider all controls u_k and corresponding next states x_{k+1}.
- Extend $\tilde{y}_k$ to obtain the partial trajectories $y_{k+1} = \big(\tilde{y}_k, u_k, x_{k+1}\big)$, for $u_k \in U_k(x_k)$.
- Run the base heuristic from each y_{k+1} to obtain the partial trajectory $R(y_{k+1})$.
- Join the partial trajectories y_{k+1} and $R(y_{k+1})$ to obtain complete trajectories denoted by $T_k(\tilde{y}_k, u_k) = \big(\tilde{y}_k, u_k, R(y_{k+1})\big)$
- Find the set of controls $\tilde{U}_k(\tilde{y}_k)$ for which $T_k(\tilde{y}_k, u_k)$ is feasible, i.e., $T_k(\tilde{y}_k, u_k) \in C$
- Choose the control $\tilde{u}_k \in \tilde{U}_k(\tilde{y}_k)$ according to the minimization

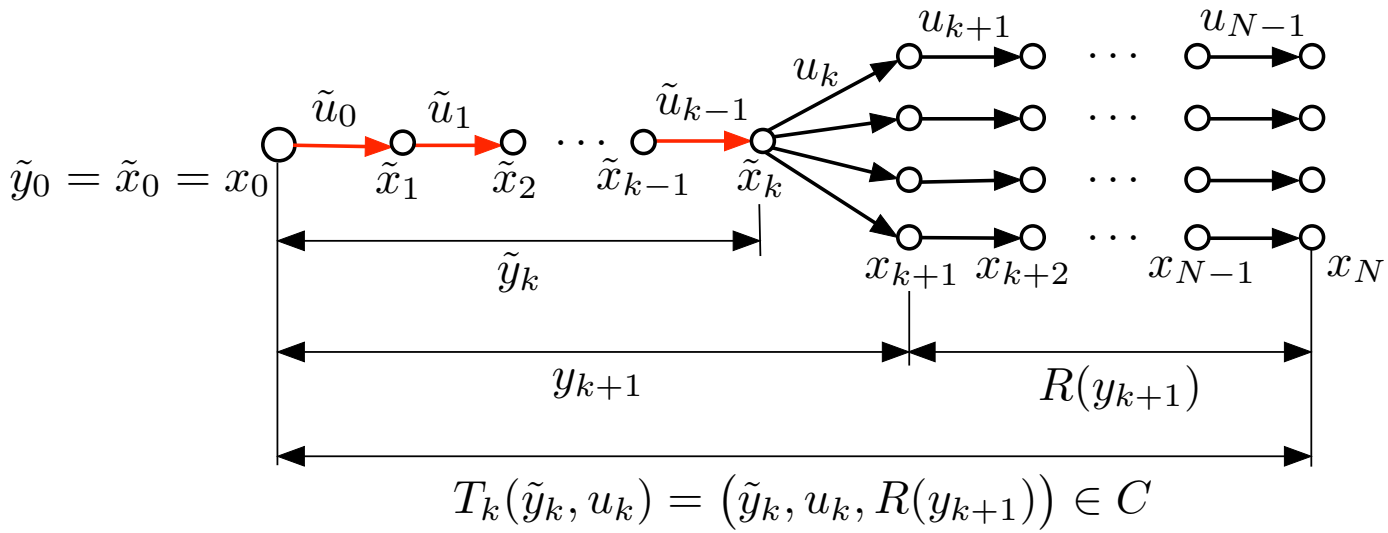
$$\tilde{u}_k \in \arg \min_{u_k \in \tilde{U}_k(\tilde{y}_k)} G\big(T_k(\tilde{y}_k, u_k)\big)$$

Constrained Traveling Salesman Example



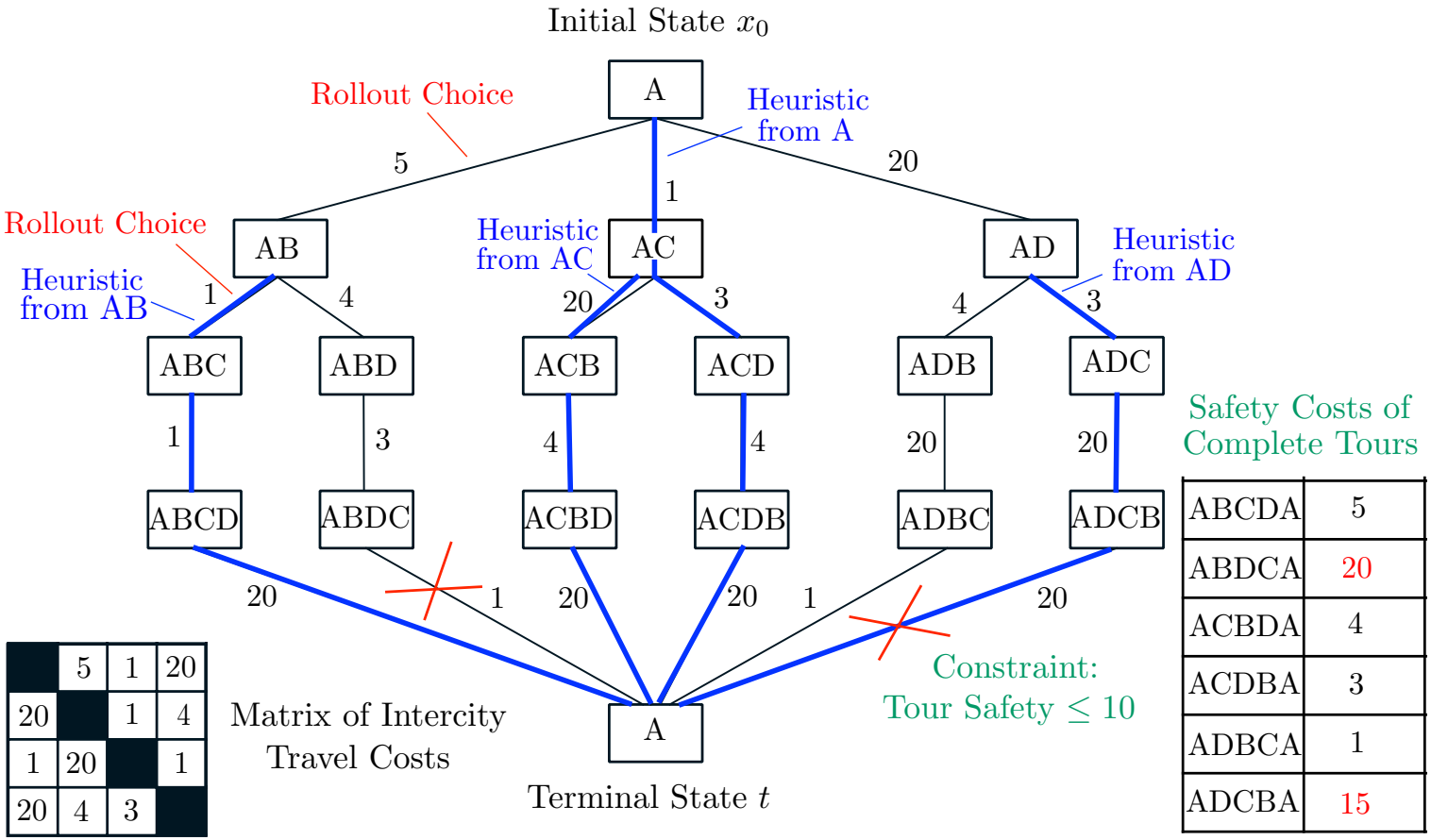
- Rollout at A: Considers partial tours AB, AC, and AD; Obtains the complete tours ABCDA, ACBDA, and ADCBA; Discards ADCBA as being infeasible; Compares ABCDA and ACBDA, finds ABCDA to have smaller cost, and selects AB.
- Rollout at AB: Considers the partial tours ABC and ABD; Obtains the complete tours ABCDA and ABDCA; Discards ABDCA as being infeasible; Selects the complete tour ABCDA.

Constrained Rollout Algorithm Properties



- The notions of sequential consistency and sequential improvement apply. Their definition includes that the set of "feasible" controls $\tilde{U}_k(\tilde{y}_k)$ is nonempty for all k.
- Sequential improvement condition: The min heuristic Q-factor over $\tilde{U}_k(\tilde{y}_k)$ is no larger than the heuristic cost at $\tilde{y}_k$ (see the notes).
- Fortified version (if sequential improvement does not hold; see the notes):
  - Maintains the "tentative best" trajectory, and follows it up to generating a better trajectory through rollout.
  - Has the cost improvement property, assuming the base heuristic generates a feasible trajectory starting from the initial condition $\tilde{y}_0 = x_0$.
- Multiagent version: Selects one-control-component-at-a-time (apply constrained rollout to the equivalent reformulation, i.e., the one with control space "unfolded").

Example of Sequential Consistency and Sequential Improvement



	5	1	20
20		1	4
1	20		1
20	4	3	

ABCDA	5
ABDCA	20
ACBDA	4
ACDBA	3
ADBCA	1
ADCBA	15

- The heuristic is not sequentially consistent at A, but it is sequentially improving.
- If we change the D→A cost to 25, the heuristic is not sequentially improving at A, and the cost improvement property is lost.
- If we change the D→A cost to 25 and we add fortification, the rollout algorithm at A sticks with the initial tentative best trajectory ACDBA, and rejects ABCDA.

Break Time

A Five-Minute Break

A Retrospective Summary on Deterministic Constrained Rollout

Structural components

(1) Trajectories T consisting of a sequence of decisions defined by a layered/optimal control graph

(2) A cost function $G(T)$ to rank trajectories

(3) A constraint $T \in C$ to determine feasibility of trajectories

(4) A base heuristic that starts from a partial trajectory and generates a complete trajectory

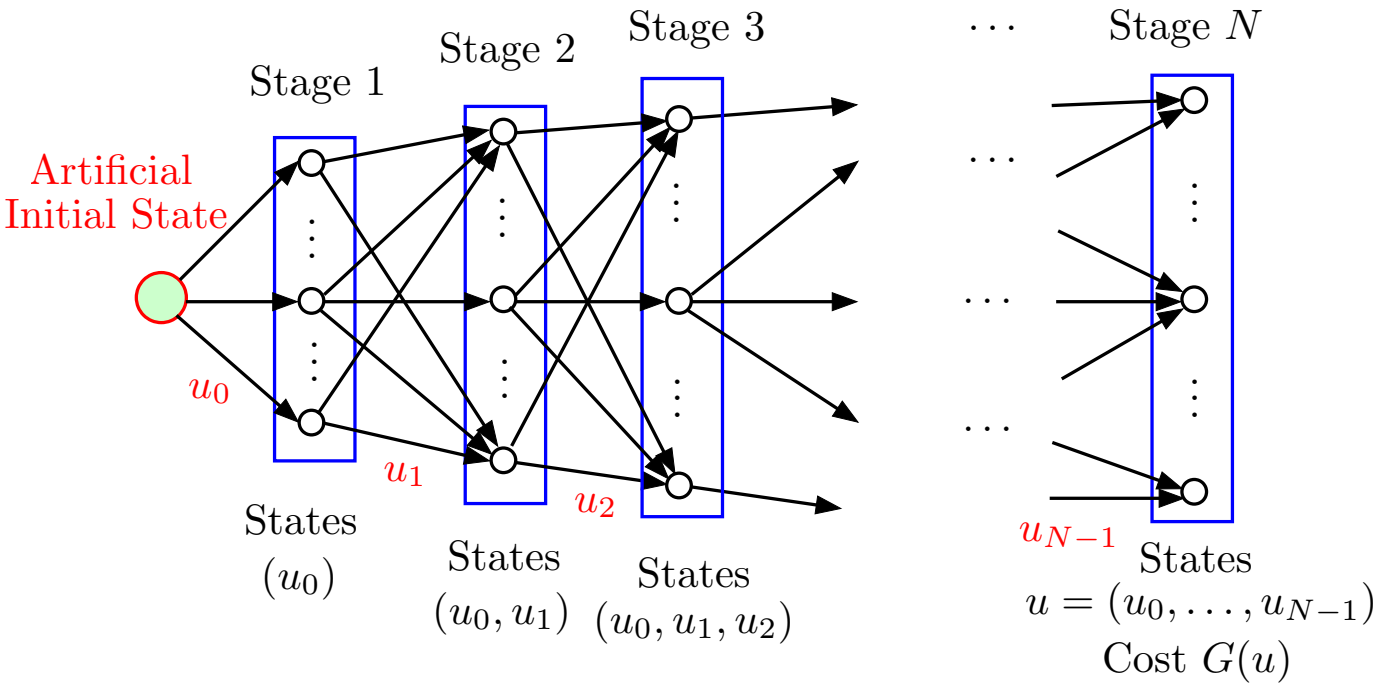
Given (1)

The choices of (2), (3), and (4) are independent of each other

In particular, given (1)-(3):

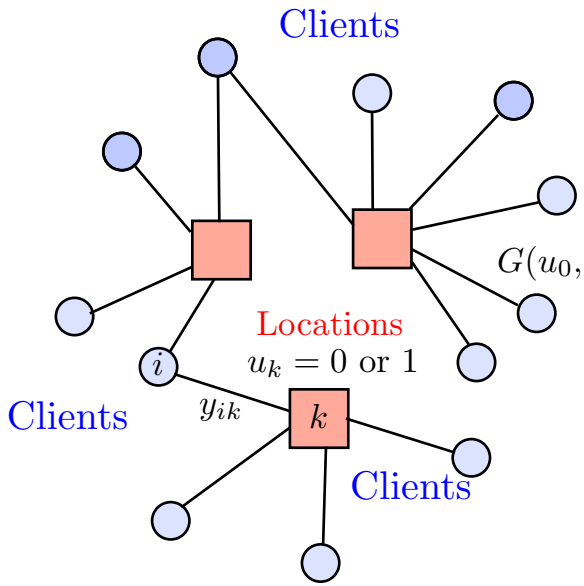
We can try several different base heuristics or a superheuristic

General Discrete Optimization Problem: Minimize $G(u)$ Subject to $u \in C$, where $u = (u_0, \ldots, u_{N-1})$



- This is a special case of the constrained deterministic optimal control problem where each state x_k can only take a single value, i.e., $x_k \equiv$ "artificial" x_0.
- A very broad range of problems, e.g., combinatorial, integer programming, etc.
- Solution by constrained rollout applies. Provides entry point to the use of RL ideas in discrete optimization through DP and approximation in value space.
- Competing methods: local/random search, genetic algorithms, integer programming/branch and bound, etc. Rollout is different.

Facility Location: A Prototype Integer Programming Problem



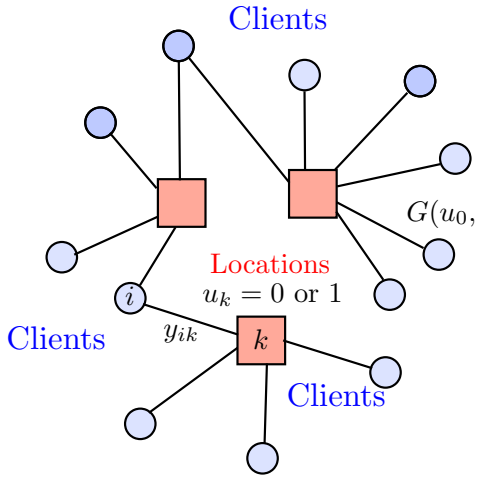
C : Set of $(u_0, \ldots, u_{N-1})$ such that $u_k \in \{0, 1\}$ and can satisfy the demand and other constraints (e.g., public policy constraints)

$$G(u_0, \ldots, u_{N-1}) = \min_{(y_{ik}, i, k) \in H(u_0, \ldots, u_{N-1})} \sum_{i=1}^{M} \sum_{k=0}^{N-1} a_{ik} y_{ik} + \sum_{k=0}^{N-1} b_k u_k$$

$H(u_0, \ldots, u_{N-1})$: Set of feasible demand allocations, i.e. Set of $y_{ik} \geq 0$ such that $\sum_k y_{ik} = d_i$ for all i, $\sum_i y_{ik} \leq u_k c_k$ for all k

- Place facilities at some of the given candidate locations to serve M “clients."
- Client $i = 1, \ldots, M$ has a demand d_i for services that may be satisfied at a location $k = 0, \ldots, N-1$ at a cost a_{ik} per unit.
- A facility placed at location k has capacity c_k and cost b_k. Here $u_k \in \{0, 1\}$, with $u_k = 1$ if a facility is placed at k.
- Problem: Minimize $\sum_{i=1}^{M} \sum_{k=0}^{N-1} a_{ik} y_{ik} + \sum_{k=0}^{N-1} b_k u_k$ subject to total demand satisfaction constraints ($y_{ik} \geq 0$, $\sum_k y_{ik} = d_i$ for all i, and $\sum_i y_{ik} \leq u_k c_k$ for all k).
- There may be additional constraints on u, but we will ignore for the moment.
- Note: If the placement variables u_k are known, the remaining problem is easily solvable (it is a linear “transportation" problem).

Facility Location Problem: Formulation for Constrained Rollout



C : Set of $(u_0, \ldots, u_{N-1})$ such that $u_k \in \{0, 1\}$ and can satisfy the demand and other constraints (e.g., public policy constraints)

$$G(u_0, \ldots, u_{N-1}) = \min_{(y_{ik}, i, k) \in H(u_0, \ldots, u_{N-1})} \sum_{i=1}^{M} \sum_{k=0}^{N-1} a_{ik} y_{ik} + \sum_{k=0}^{N-1} b_k u_k$$

$H(u_0, \ldots, u_{N-1})$: Set of feasible demand allocations, i.e.
Set of $y_{ik} \geq 0$ such that
$\sum_k y_{ik} = d_i$ for all i,
$\sum_i y_{ik} \leq u_k c_k$ for all k

- Consider placements one location at a time.
- Stage k = Placement decision $u_k \in \{0, 1\}$ at location k (N stages).
- Base heuristic: Having fixed $u_0, \ldots, u_k$, place a facility in all remaining locations.
- Rollout: Having fixed $u_0, \ldots, u_k$, compare two possibilities:
  - Set $u_{k+1} = 1$ (place facility at location $k + 1$), set $u_{k+2} = \cdots = u_{N-1} = 1$ (as per the base heuristic), and solve the remaining problem.
  - Set $u_{k+1} = 0$ (don't place facility at location $k + 1$), set $u_{k+2} = \cdots = u_{N-1} = 1$ (as per the base heuristic), and solve the remaining problem.
- Select $u_{k+1} = 1$ or $u_{k+1} = 0$ depending on which yields feasibility and min cost.
- Sequential improvement is satisfied in the absence of additional constraints.
- Transportation problems are similar; solved efficiently with the auction algorithm (see literature on network optimization).

Final Notes

The material of today's lecture is covered in the "Lessons from AlphaZero ..." text

In the next lecture we will cover:

- Stochastic Rollout.
- Monte Carlo Tree Search.
- Rollout for infinite spaces problems.

About your project:

- Send me email (dbertsek@asu.edu)
- Make appointment to talk by zoom (there are no fixed office hours in this course)
- Please send me by the end of the spring break a one-page-or-less proposal about your term paper, be it a read-and-report type or a mini-research project