Efficient Tree Generation for Globally Optimal Decisions under Probabilistic Outcomes

Dr. Les Servi, FS
lds112@gmail.com

Dr. She'ifa Punla-Green
The MITRE Corporation
spunla-green@mitre.org

Dr. Berk Ozturk
The MITRE Corporation
bozturk@mitre.org

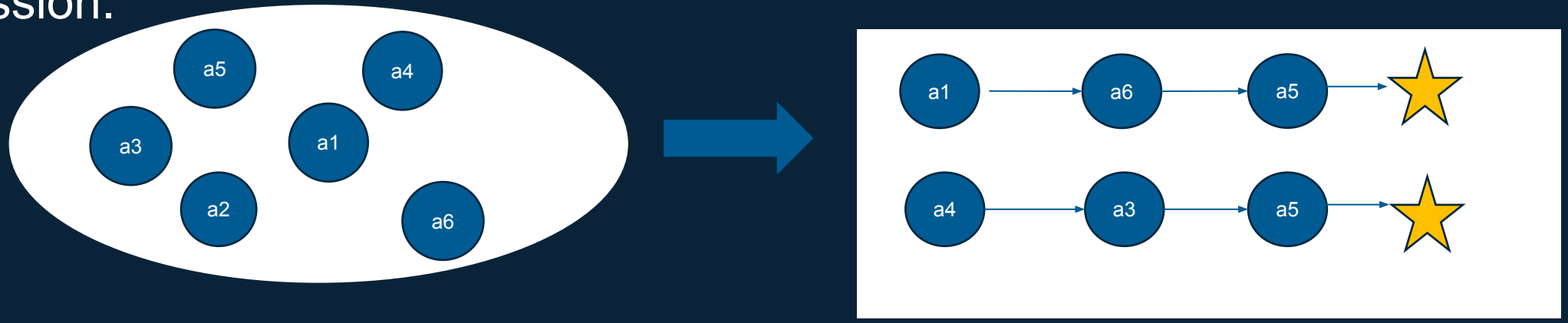
MORS Cyber Community of Practice

August 20, 2025

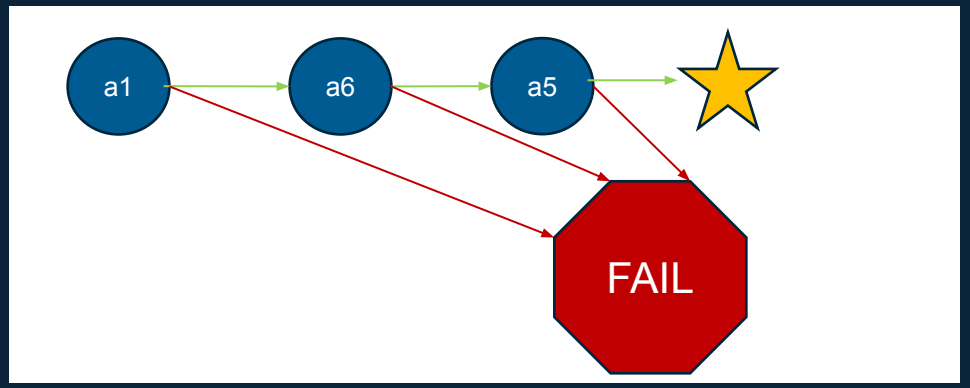
MITRE | SOLVING PROBLEMS FOR A SAFER WORLD

Algorithmic Strategy Generation

- Possible actions can be combined into sequential strategies that accomplish a mission.

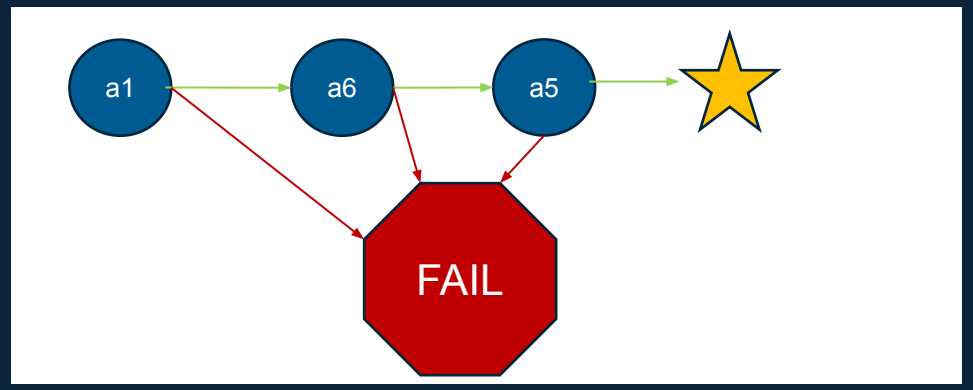


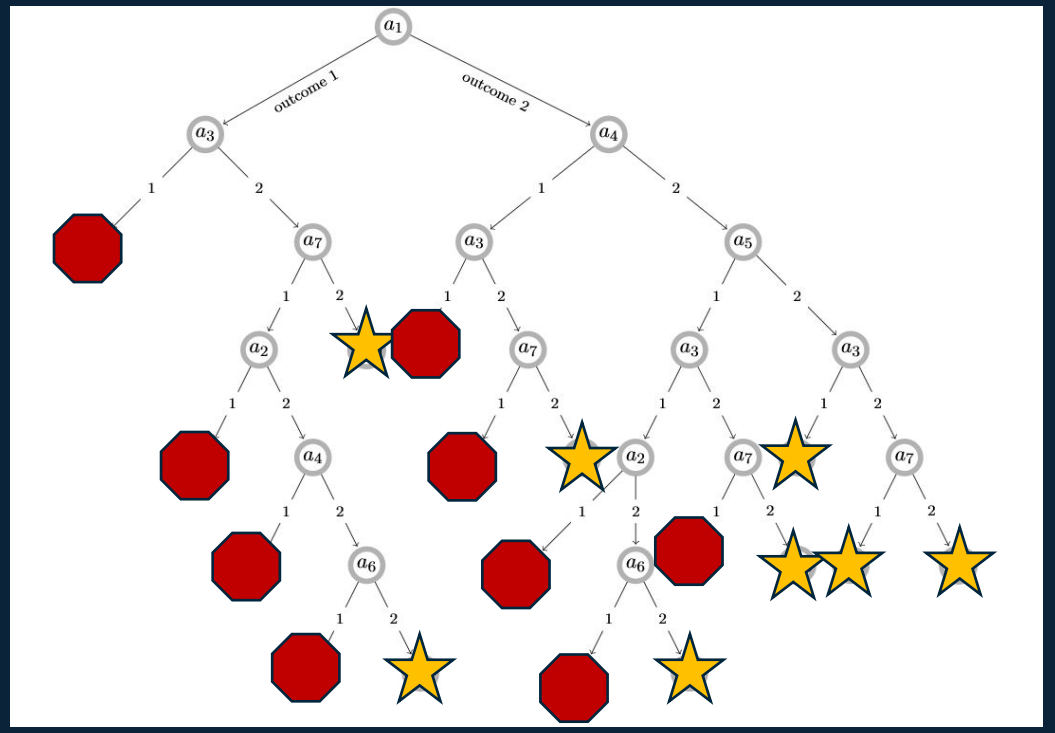
- But given actions with uncertain outcomes, good strategies still have a chance to fail!



Adaptive Strategies via Decision Trees

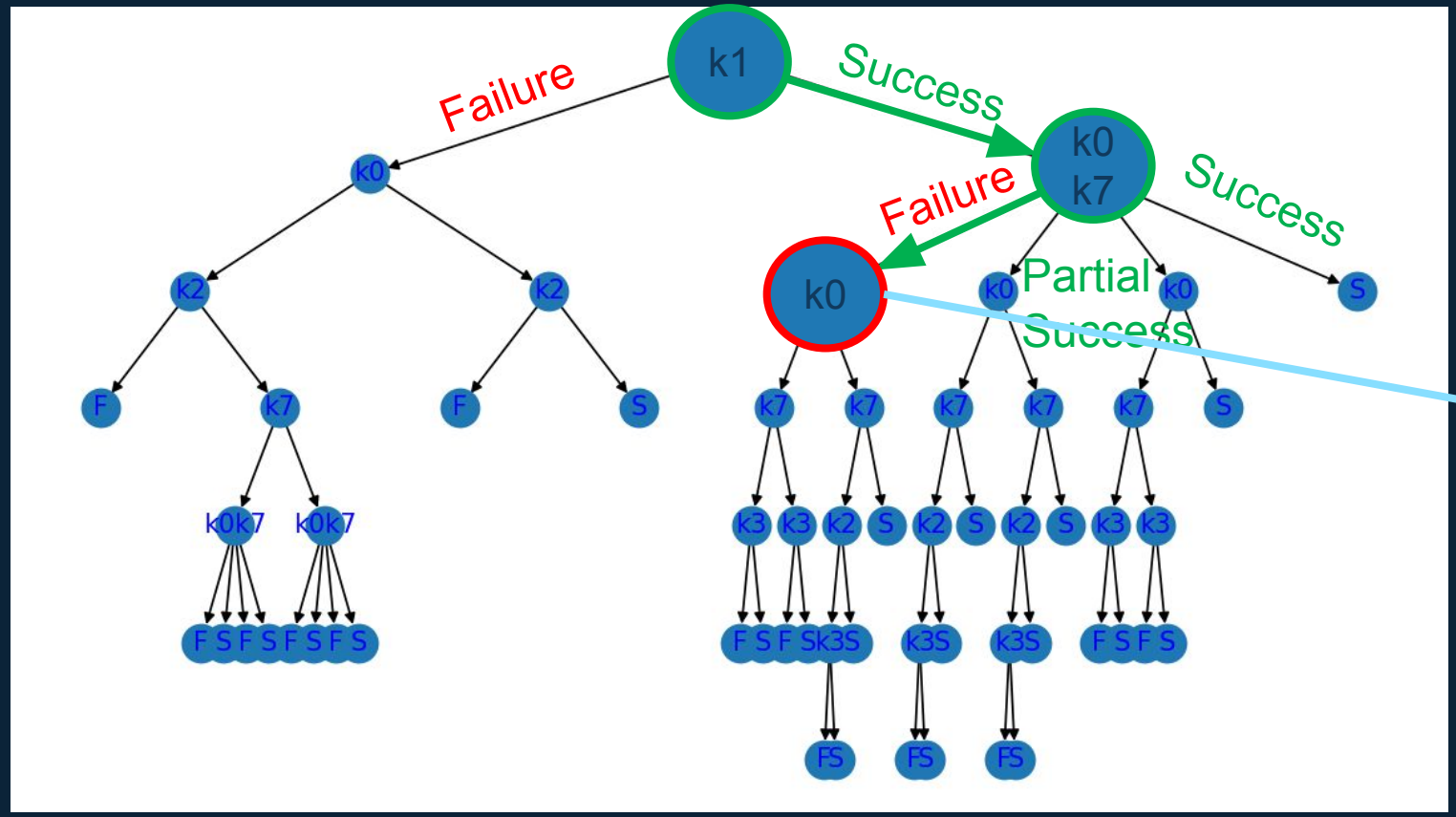
- *Prescriptive* decision trees define the next optimal action in the strategy sequence based on the uncertain outcomes of previous actions, while accounting for the complex interdependencies between actions and outcomes.





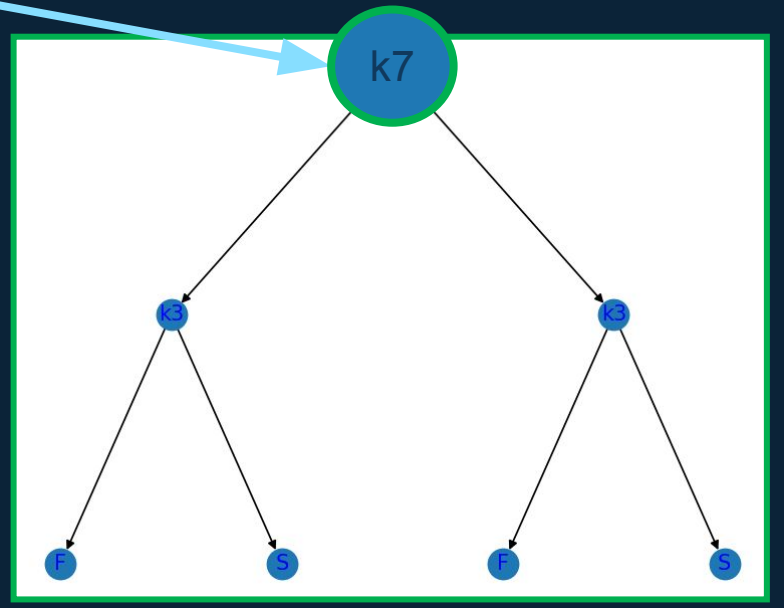
MITRE

Real-Time Human Machine Teaming



User: I implemented the first two kill chains in the tree, but k0 is no longer makes sense because ….

Tool: In that case, the best alternative is….



Applications of Decision Trees: Cybersecurity

	Prerequisite	Outcomes	Probability
Start admin attack		Success	1
Seek admin authorization	(Start admin attack, Success)	Full Success	0.15
		Some Success	0.6
		No Success	0.25
Attempt access crown jewel CPU	(Seek admin authorization, Full Success)	Great Success	0.21
		Some Success	0.7
		No Success	0.09
Create phishing traps	(Attempt access crown jewel CPU, Some Success)	High-level user	0.2
		Low-level user	0.8

```
START:
Do 'Start admin attack' d11,
Do 'Seek admin authorization' d12,
   If 'Full success' d12-1 | Prob: 0.15 |,
      Do 'Attempt access to crown jewel CPU' d14,
         If 'Some success' d14-1 | Prob: 0.70 |,
            Do 'Start phishing attack' d21,
            Do 'Creating phishing traps' d22,
                  If 'Success with a high level user' d22-1 | Prob: 0.20 |,
                     Do 'Corrupt important database' d23,
                        If 'Major damage' d23-1,
                           STOP.
                        If 'Minor damage' d23-2 | Prob: 0.15 |,
                           STOP.
                        If 'No damage' d23-3 | Prob: 0.05 |,
                           STOP.
                  If 'Success with a low level user' d22-2 | Prob: 0.80 |,
                     Do 'Seek other phishing opportunities' d24,
                        If 'Minor success' d24-1 | Prob: 0.70 |,
                           STOP.
                        If 'Major success' d24-2 | Prob: 0.30 |,
                           STOP.
         If 'Great success' d14-2 | Prob: 0.21 |,
            Do 'Start phishing attack' d21,
            Do 'Creating phishing traps' d22,
```

*illustrative

Scope

- **Actions**, with
  - Discrete probabilistic outcomes (not restricted to binary outcome actions)
  - Complex Interdependencies, such as:
    - Prerequisites, i.e. actions that must be attempted and result in a specific outcome before another action may be attempted, and
    - Preclusions, i.e. actions that if attempted and resulting in a specific outcome prevent another action from being attempted.
- **Decision Space**
  - Finite, and terminates either when the agent's goals are achieved, or when no additional actions are available.

Illustrative Example: Input

	Action	Prerequisite	Preclusion	Cost	Outcome	Probability	Reward
Seek admin authorization	a_1	–	–	1	1 2	0.4 0.6	– –
Create phishing traps 1	a_2	–	–	1	1 2	0.4 0.6	– –
Create phishing traps 2	a_3	–	–	1	1 2	0.7 0.3	– –
Corrupt database	a_4	$(a_1,2)$ OR $(a_3,2)$	–	1	1 2	0.7 0.3	– –
Attack asset 1	a_5	$(a_4,2)$	$(a_3,1)$ OR $(a_3,2)$	1	1 2	0.4 0.6	– 50
Attack asset 2	a_6	$(a_4,2)$ AND $(a_2,2)$	–	1	1 2	0.6 0.4	– 10
Attack assets 2 and 3	a_7	$(a_3,2)$	–	1	1 2	0.9 0.1	– 100

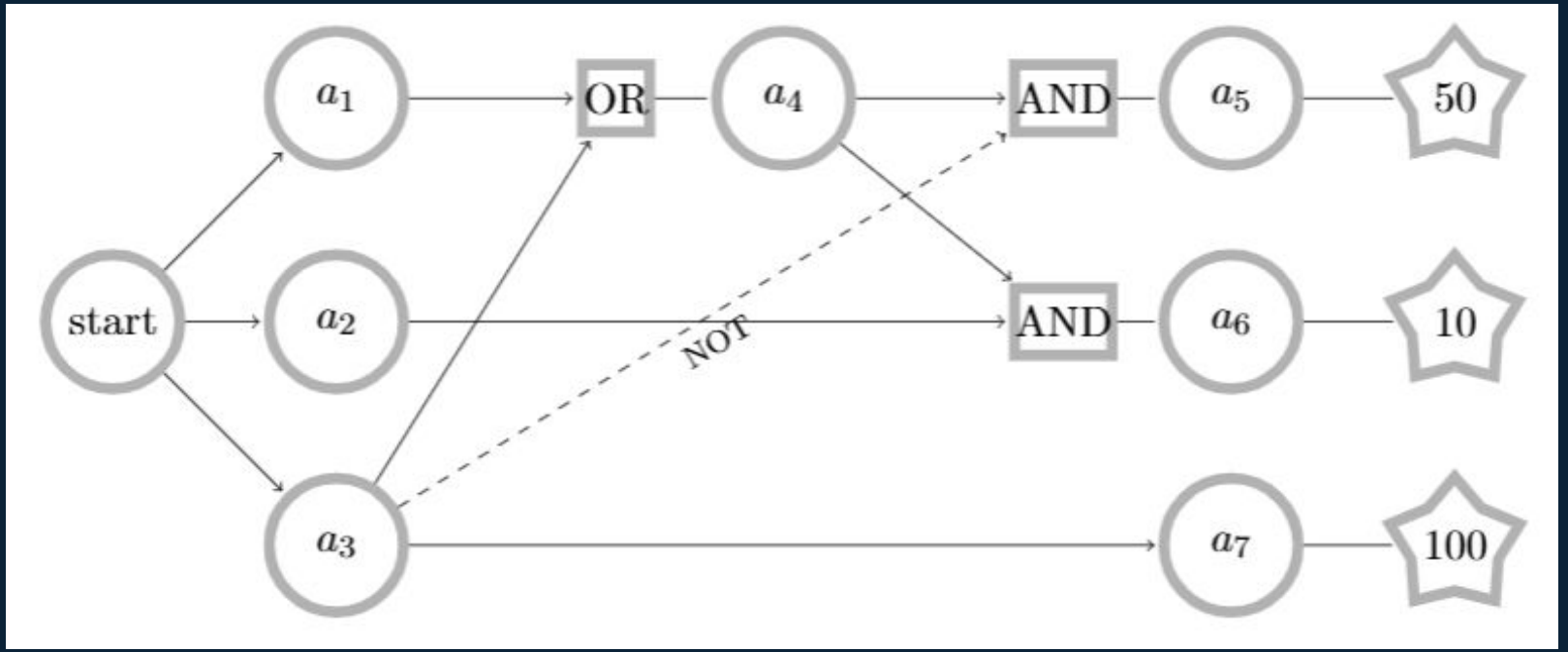
*illustrative

Illustrative Example: Input

Action	Prerequisite	Preclusion	Cost	Outcome	Probability	Reward
a_1	–	–	1	1 2	0.4 0.6	– –
a_2	–	–	1	1 2	0.4 0.6	– –
a_3	–	–	1	1 2	0.7 0.3	– –
a_4	(a_1,2) OR (a_3,2)	–	1	1 2	0.7 0.3	– –
a_5	(a_4,2)	(a_3,1) OR (a_3,2)	1	1 2	0.4 0.6	– 50
a_6	(a_4,2) AND (a_2,2)	–	1	1 2	0.6 0.4	– 10
a_7	(a_3,2)	–	1	1 2	0.9 0.1	– 100

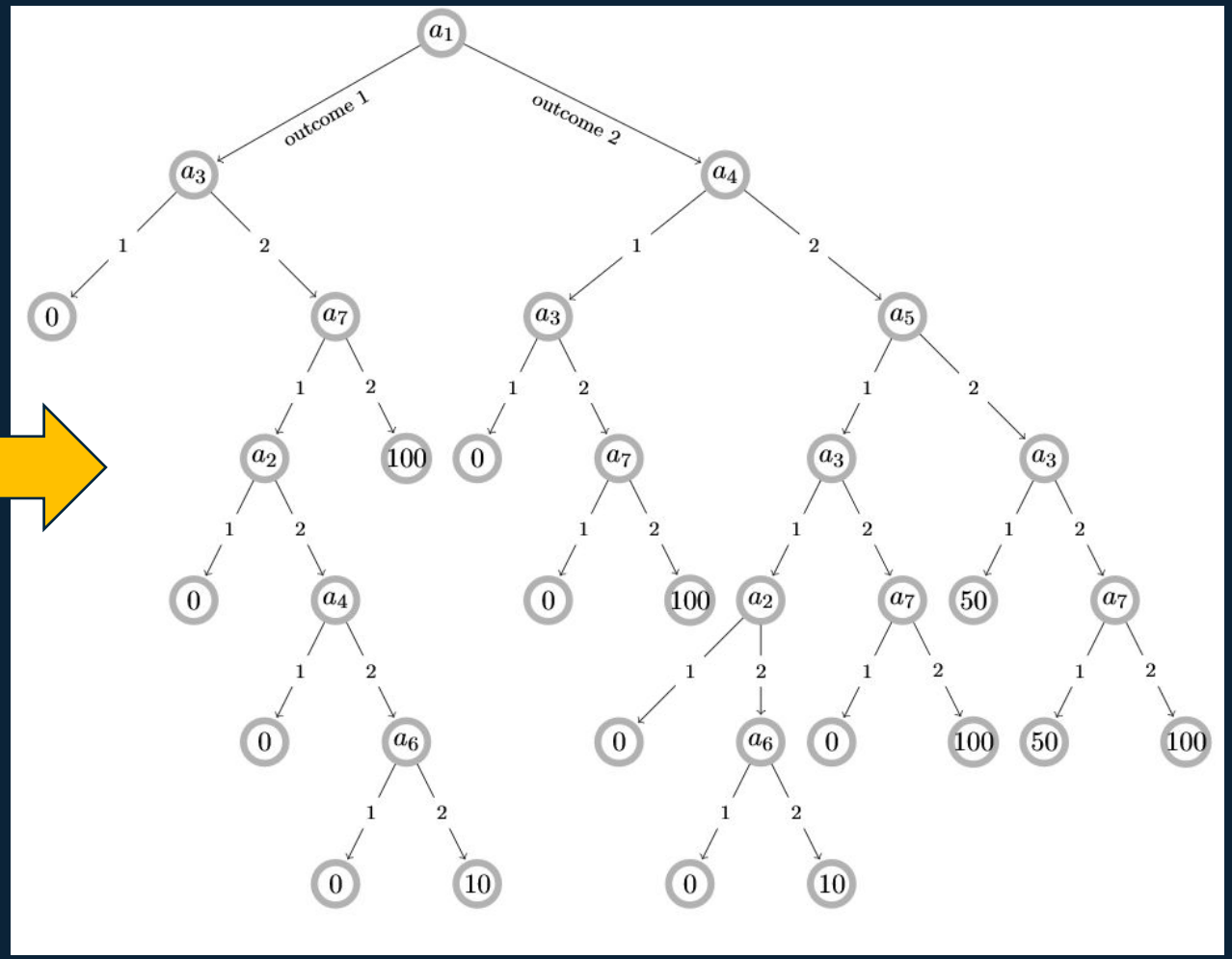
Action Data

Action Dependency Graph



Illustrative Example: Output

Action	Prerequisite	Preclusion	Cost	Outcome	Probability	Reward
a_1	–	–	1	1 2	0.4 0.6	– –
a_2	–	–	1	1 2	0.4 0.6	– –
a_3	–	–	1	1 2	0.7 0.3	– –
a_4	(a_1,2) OR (a_3,2)	–	1	1 2	0.7 0.3	– –
a_5	(a_4,2)	(a_3,1) OR (a_3,2)	1	1 2	0.4 0.6	– 50
a_6	(a_4,2) AND (a_2,2)	–	1	1 2	0.6 0.4	– 10
a_7	(a_3,2)	–	1	1 2	0.9 0.1	– 100



Optimal Decision Trees vs Markov Decision Processes

- Scope of the State
  - DTs: Encodes all past actions of the agent so it is impossible for a state to be revisited.
  - MDPs: Encodes minimal information about past actions so the state space is a closed system where states can be revisited.
- Termination Criteria
  - DTs: Leverages the finite state space that is created by termination criteria where no additional actions can be taken or a goal is reached.
  - MDPs: Can be formulated with absorbing states or action budget constraints, but those add significant computational complexity.

Our approach can efficiently model problems with complex interdependencies

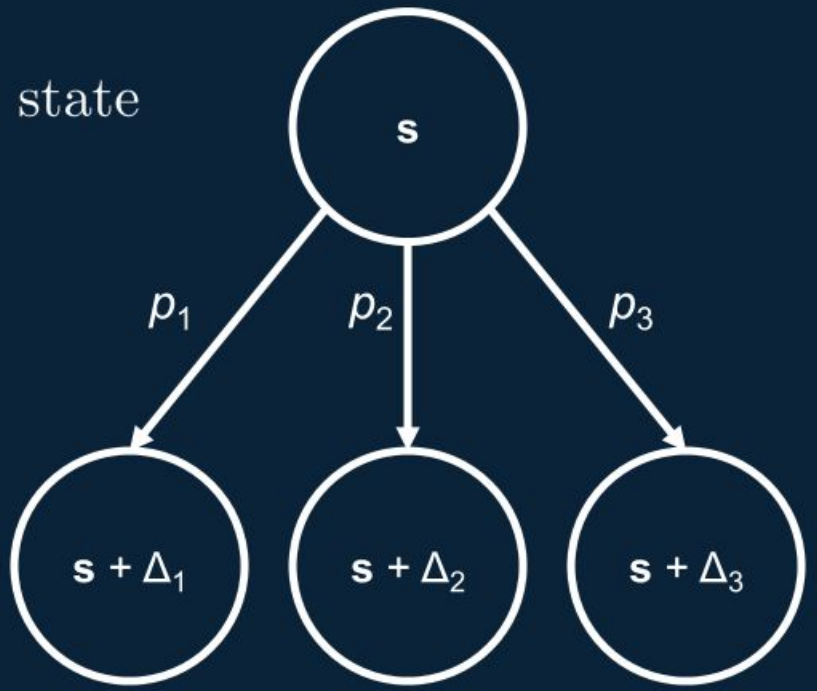
Nomenclature

Moving through the state space

Below are the mathematical primitives describing state transitions.

- State $\mathbf{s} \in \mathbb{Z}^n$ describes the here-and-now,
- Each action $r := \{(p_j, \Delta_j), \forall j \in |r|\}$ is a set that contains state changes Δ_j and associated probabilities p_j,
- Action function $\mathcal{R}(\mathbf{s})$ returns all actions available at $\mathbf{s}$ (abstracts away dependencies).

An action $r \in \mathcal{R}(\mathbf{s})$ taken at state $\mathbf{s}$ gives new state $\mathbf{s} + \Delta_j$ with probability p_j.



Obtaining rewards while meeting budgets

As aforementioned, we solve a finite-horizon decision problem, so we need the following machinery:

- Cost function $c(r)$ maps each action r to its associated cost,
- Budget function $B(\mathbf{s})$ returns the remaining budget at state $\mathbf{s}$,
- Reward function $\rho(\mathbf{s})$ evaluates the reward achieved at state $\mathbf{s}$.

We don't place any constraints on the form of these functions, other than restricting the budget to monotonically decrease as more actions are taken.

Methodology

Overview of methodology

The approach is a 4-step dynamic program.

1. Compute *rewarding sets*.
2. Generate a *full graph*: Explore states and available actions that have potential for increased reward.
3. Evaluate the *reduced graph*: Combination of all subtrees that maximize expectation of reward.
4. Find the *optimal decision tree:* The subtree within the reduced graph that maximizes a secondary objective function.

Steps 3 and 4 use conventional dynamic programming to find the optimal strategy. So will focus on Steps 1 and 2 in this talk.

What would naïve dynamic programming (DP) do?

It would simply explore ALL states available to determine the optimal strategy, within the state set (*the full graph*) below:

$$\mathcal{S}_{\mathrm{FG}} = \left\{ \mathbf{s} : \mathbf{s}_{\mathrm{root}} \in \mathcal{S}_{\mathrm{FG}};\ \text{if } \mathbf{s} \in \mathcal{S}_{\mathrm{FG}},\ \mathbf{s} + \Delta \ \in\ \mathcal{S}_{\mathrm{FG}},\ \forall (\mathrm{p}, \Delta) \in \mathrm{r},\ \forall \mathrm{r} \in \mathcal{R}(\mathbf{s}) \right\}$$

Colloquially, starting at root state $\mathbf{s}_{\mathrm{root}}$, it would recursively apply all available actions at all states until no actions remain. This is expensive.

What if we can reduce the size of the full graph, while retaining optimality?

We can accelerate DP using knowledge of rewards and the action dependency graph.

If we have a set of states $\mathcal{S}_\rho$ that we know give rewards, we can simply restrict ourselves to trying actions that have potential for increased reward. We can identify such trajectories to rewards, called a *rewarding set*

$$\mathcal{P}(\mathbf{s}_{\text{root}}, \mathbf{s}_k) = \left\{ \left\{ (r, j) \right\} : \mathbf{s}_k = \mathbf{s}_{\text{root}} + \sum_{(r,j) \in \mathcal{P}(\mathbf{s}_{\text{root}}, \mathbf{s}_k)} \sum_{(p_j, \Delta_j) \subseteq r} \Delta_j \right\},$$

Rewarding set from root state to end state = { actions and associated outcomes so that we reach the end state from the root state }

How do we find rewarding sets? **Solve a network flow problem through the action dependency graph.**

Example of computing (dominating) rewarding sets

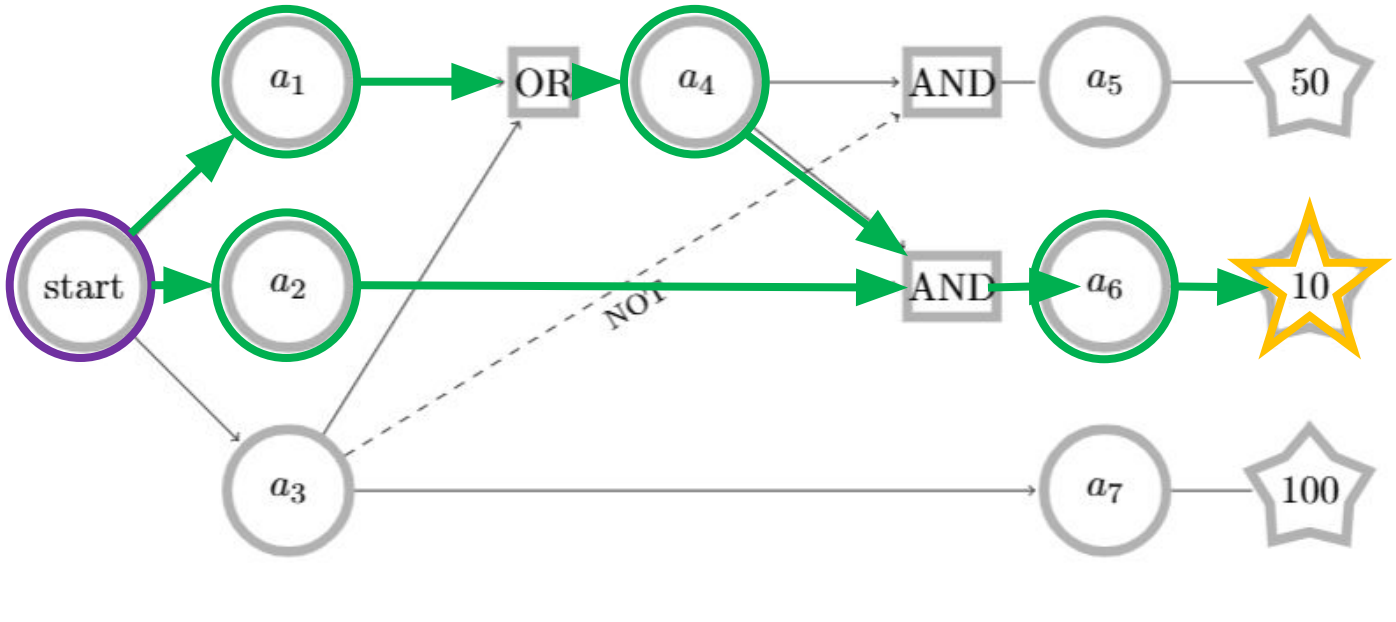


Figure 2: Action dependency graph for illustrative example.

For a reward of 10, $\{(a_1, 2), (a_2, 2), (a_4, 2), (a_6, 2)\}$ is one of the rewarding sets.

All (dominating) rewarding sets

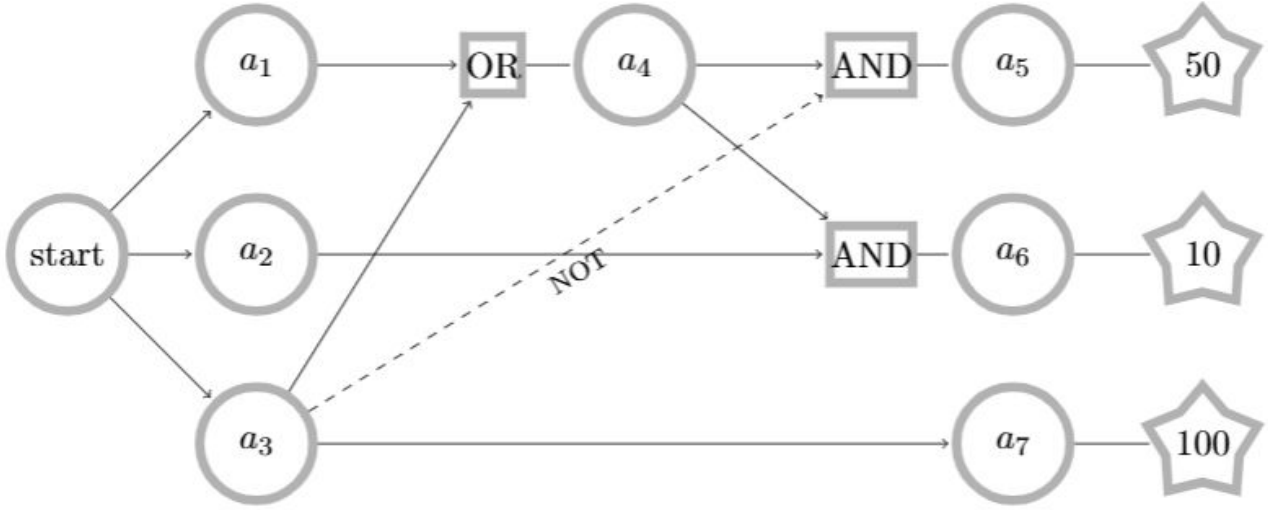


Figure 2: Action dependency graph for illustrative example.

1. For a reward of 50, $\{(a_1, 2), (a_4, 2), (a_5, 2)\}$,
2. For a reward of 10, $\{(a_1, 2), (a_2, 2), (a_4, 2), (a_6, 2)\}$,
3. For a reward of 10, $\{(a_2, 2), (a_3, 2), (a_4, 2), (a_6, 2)\}$,
4. For a reward of 100, $\{(a_3, 2), (a_7, 2)\}\}$.

Tradeoffs of the rewarding set approach for DP

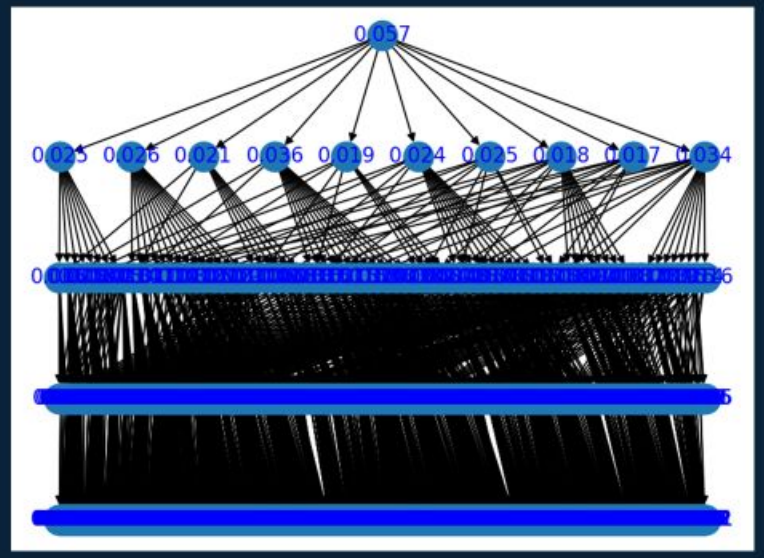
- If there are few rewarding states,
  - Gets you the optimal strategy much quicker.
- If there are many rewarding states,
  - Potentially makes the problem more tractable, or
  - Allows quick heuristic solutions only looking at most promising reward states.
  - Can also expand an existing strategy, as time allows, augmenting the search by allowing more reward states.

Downside: Need to be able to compute rewarding states and/or sets a priori

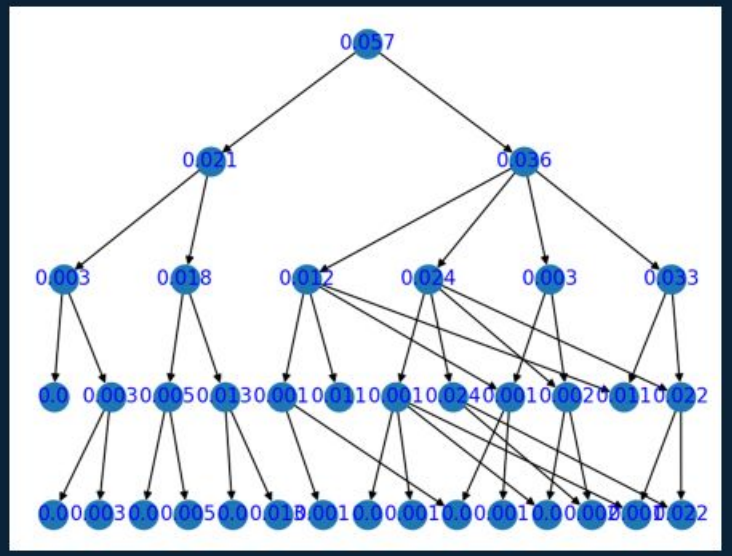
(which we do by interpreting the action dependency graph).

Results

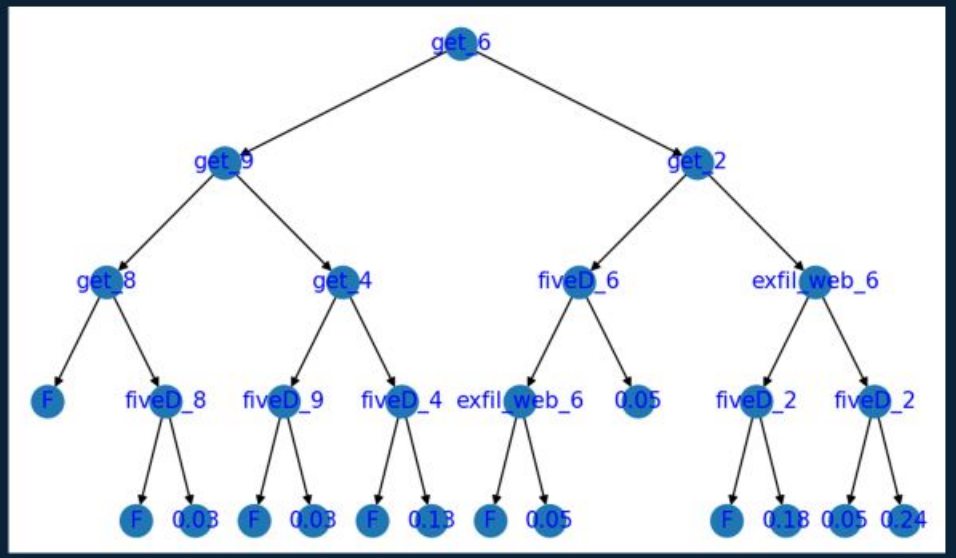
Examples of full, reduced graph and tree



Full graph:
Fast enumeration of alternatives



Reduced graph:
Determination of all equally good strategies



Optimal decision tree:
Final strategy via secondary optimization

Naïve DP

Index	N	B	$\Phi(\mathbf{s}_{\text{root}})$	T_{total}	$\lvert\mathcal{S}_{\text{FG}}\rvert$	$\lvert\mathcal{S}_{\text{RG}}\rvert$	$\lvert\mathcal{S}_{\text{DT}}\rvert$
5	15	10	0.466594	8.95	9466	6758	314
7	17	10	0.181530	23.57	18775	4064	456
8	20	10	0.026389	163.21	83854	525	76
10	20	10	0.021860	83.62	47320	760	90

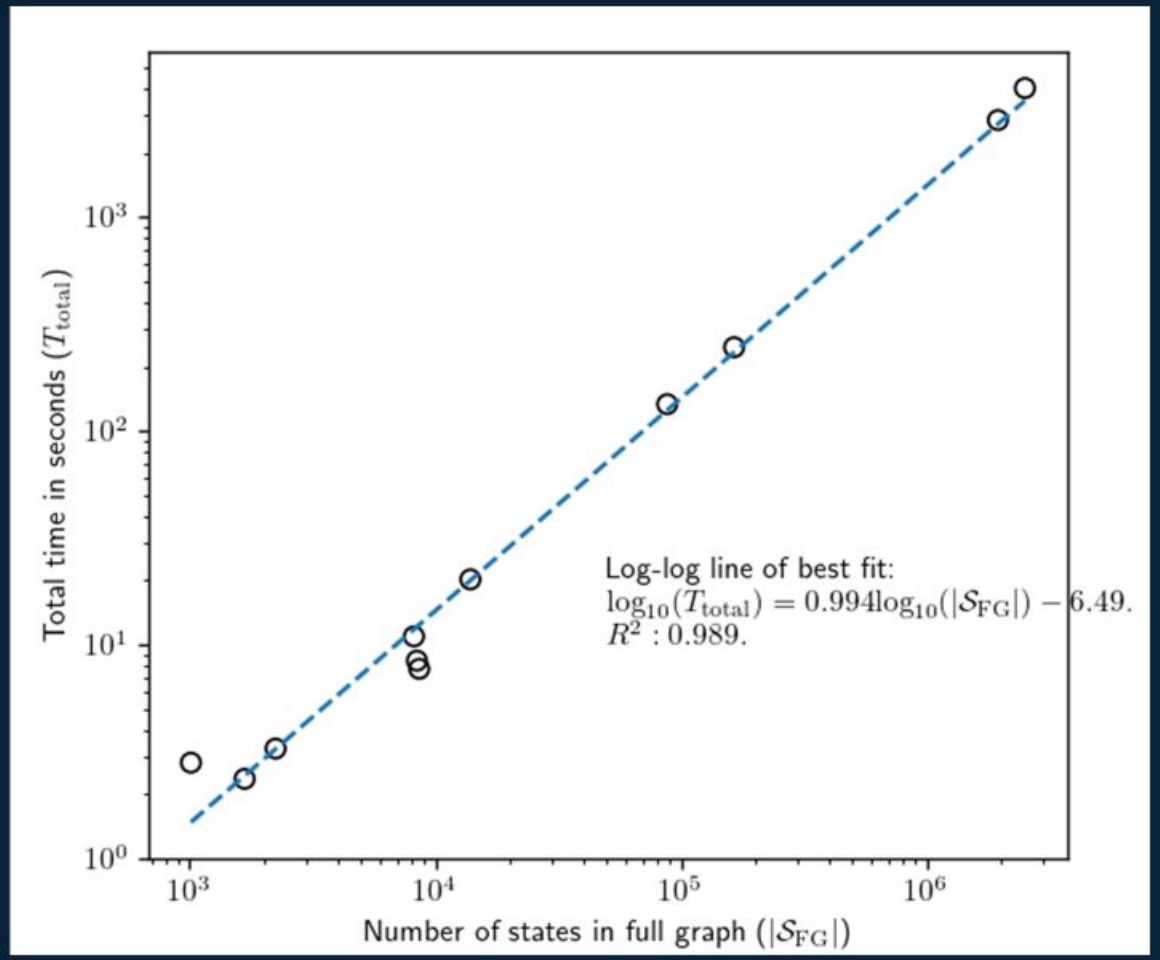
Accelerated DP

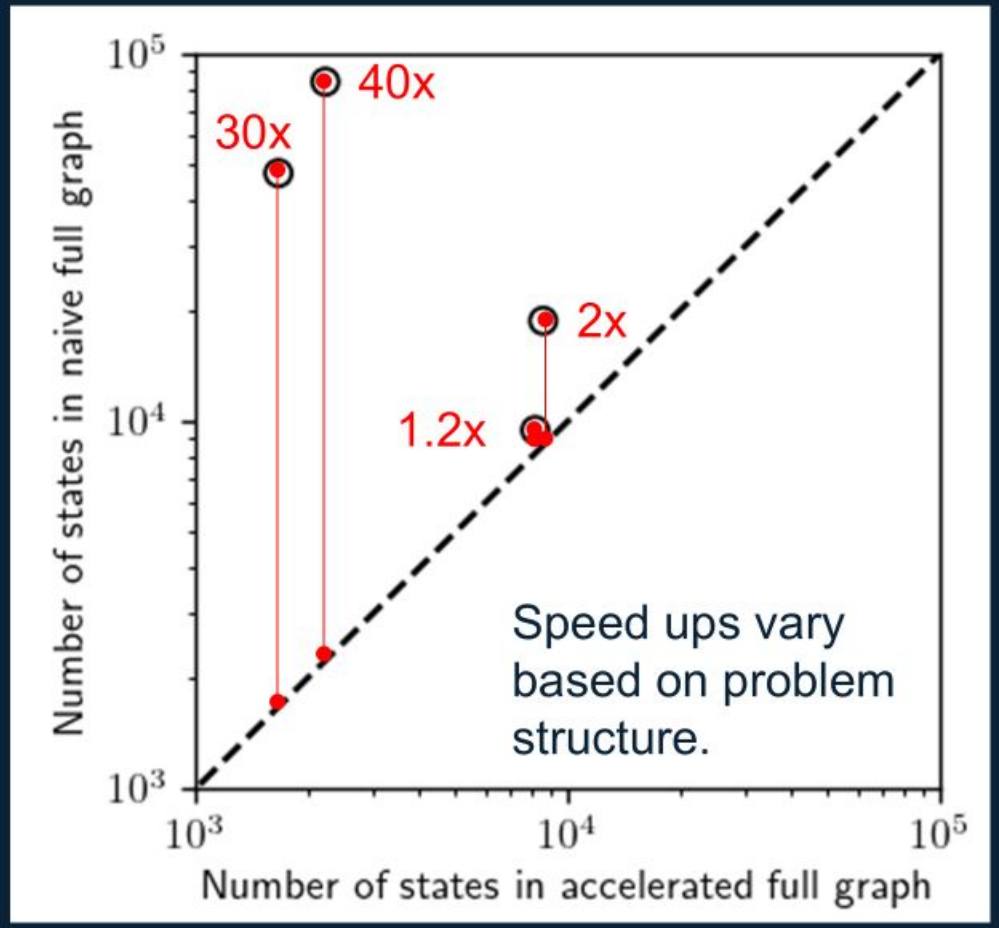
Index	N	B	$\Phi(\mathbf{s}_{\text{root}})$	T_{total}	$\lvert\mathcal{S}_{\text{FG}}\rvert$	$\lvert\mathcal{S}_{\text{RG}}\rvert$	$\lvert\mathcal{S}_{\text{DT}}\rvert$	$\lvert\mathbb{P}(\mathbf{s}_{\text{root}})\rvert$
0	25	15	0.000243	4031.24	2518548	2239364	235	20
1	20	15	0.000971	2855.48	1954163	1820303	309	28
2	20	15	0.038782	246.94	164758	105235	1347	13
3	20	15	0.034325	133.72	87950	76747	954	12
4	15	15	0.467120	20.24	13892	13892	324	12
5	15	10	0.466594	10.95	8186	6758	314	12
6	20	10	0.000968	8.43	8416	1312	86	27
7	17	10	0.181530	7.72	8599	3188	456	12
8	20	10	0.026389	3.27	2237	217	76	12
9	25	10	0.000227	2.81	1011	102	40	20
10	20	10	0.021860	2.36	1673	344	90	10

Benchmarks

- We generated random decision problems with N actions and B budget.
- Most problems could not be solved by naïve DP within a 4000s time limit.
- Rewarding sets speed up DP solution time dramatically but nonuniformly (except for the smallest problems).

The approach scales linearly with respect to number of states explored (with varying speed-ups).





Live GUI Demo

Conclusion

- *Linear* chains of actions doesn't account for failure probabilities

- Using modern methods can input potential actions and derive *best tree* of decision which most impact mission while accounting for
  - Probability of actions failing
  - Prerequisites and preclusions of actions
  - Impact of individual actions if successful
  - New intel as it is discovered
- Fast, globally optimal and scalable
- Applicable to cyber, wargaming, healthcare

Efficient Tree Generation for Globally Optimal Decisions under Probabilistic Outcomes

Dr. Les Servi, FS
lds112@gmail.com

Dr. She'ifa Punla-Green
The MITRE Corporation
spunla-green@mitre.org

Dr. Berk Ozturk
The MITRE Corporation
bozturk@mitre.org

August 2025

MITRE | SOLVING PROBLEMS FOR A SAFER WORLD

Efficient Tree Generation for Globally Optimal Decisions under Probabilistic Outcomes*

Dr. Berk Ozturk, Dr. She'ifa Punla-Green, Dr. Les Servi

EXECUTIVE SUMMARY: Many real-world problems require making sequences of decisions where the outcomes of each decision are probabilistic and uncertain, and the availability of different actions is constrained by the outcomes of previous actions. There is a need to generate policies that are adaptive to uncertainty, globally optimal, and yet scalable as the state space grows. In this paper, we propose the generation of optimal decision trees, which dictate which actions should be implemented in different outcome scenarios, while maximizing the expected reward of the strategy. Using a combination of dynamic programming and mixed-integer linear optimization, the proposed methods scale to problems with large but finite state spaces, using problem-specific information to prune away large subsets of the state space that do not yield progress towards rewards. We demonstrate that the presented approach is able to find the globally optimal decision tree in linear time with respect to the number states explored.

*PRS: Case 25-0045. Submitted to *European Journal on Operations Research*, Apr, 2025