Planning and Action Selection in Options-based Agents

Rich Sutton
Keen Technologies, University of Alberta, Amii, Openmind Research Institute
Reinforcement Learning and Artificial Intelligence Lab

openmindresearch.org

Life must be lived one small step at a time

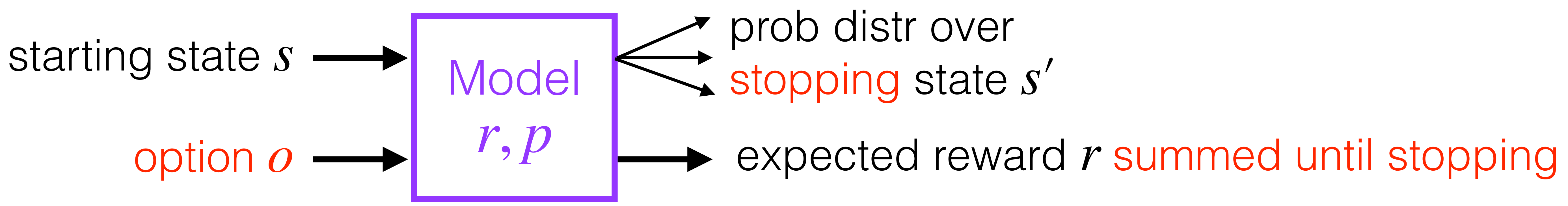
- Good behavior involves rapid variations in actions
  - as in almost all skilled motion, such as running, fighting, playing the piano
- In immediate response to the latest observations
  - as in escaping predation (the tiger jumps out)

But all the small steps must be understood and coordinated at much larger time scales

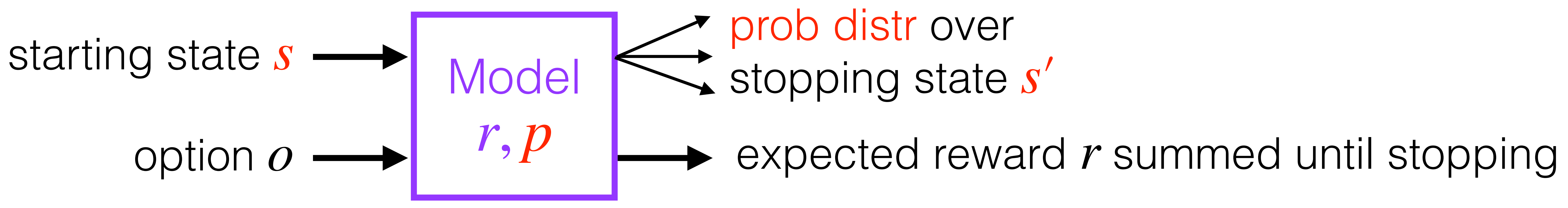
- Moving an object, walking to work, traveling to a city, taking a job

To understand requires a transition model.
Transition models for large time scales require options

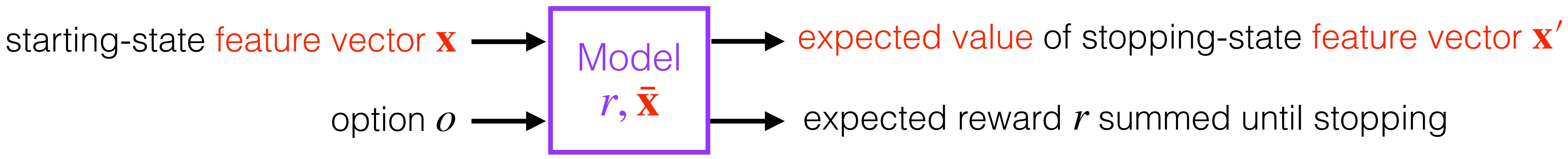
- It's clear that options and option models are the right way to plan with temporal abstraction
- An option is the minimum temporal abstraction
    - an option is just a way of behaving, and a way of stopping $o = (\pi, \gamma)$
        - a policy π : States $\rightarrow$ Pr(Actions)
        - a termination condition γ : States $\rightarrow$ $[0,1]$
- The form of an option model is dictated by Semi-MDP theory



- The form of an option model is dictated by Semi-MDP theory



With function approximation, everything must be featurized

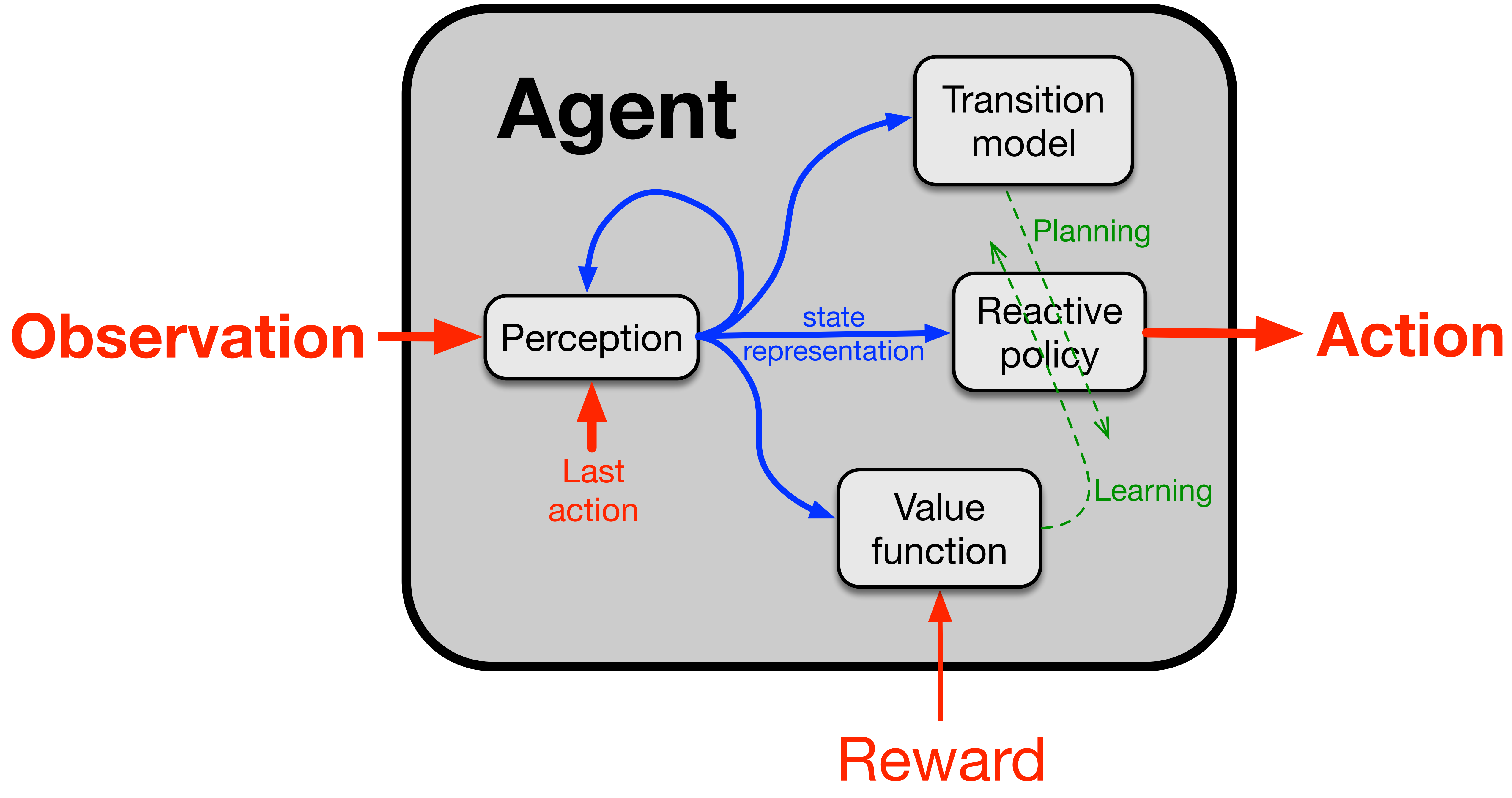


Option models with FA are our best bet for a formalization of empirical world knowledge

- Clear, general, expressive, learnable, scalable, resource efficient
- Nothing else comes close, but it is still a big thing to claim

The common model of the intelligent agent

Common to AI, psychology, control theory, economics, neuroscience, operations research…



The agent comprises four components:

Perception produces the state-feature vector used by all components

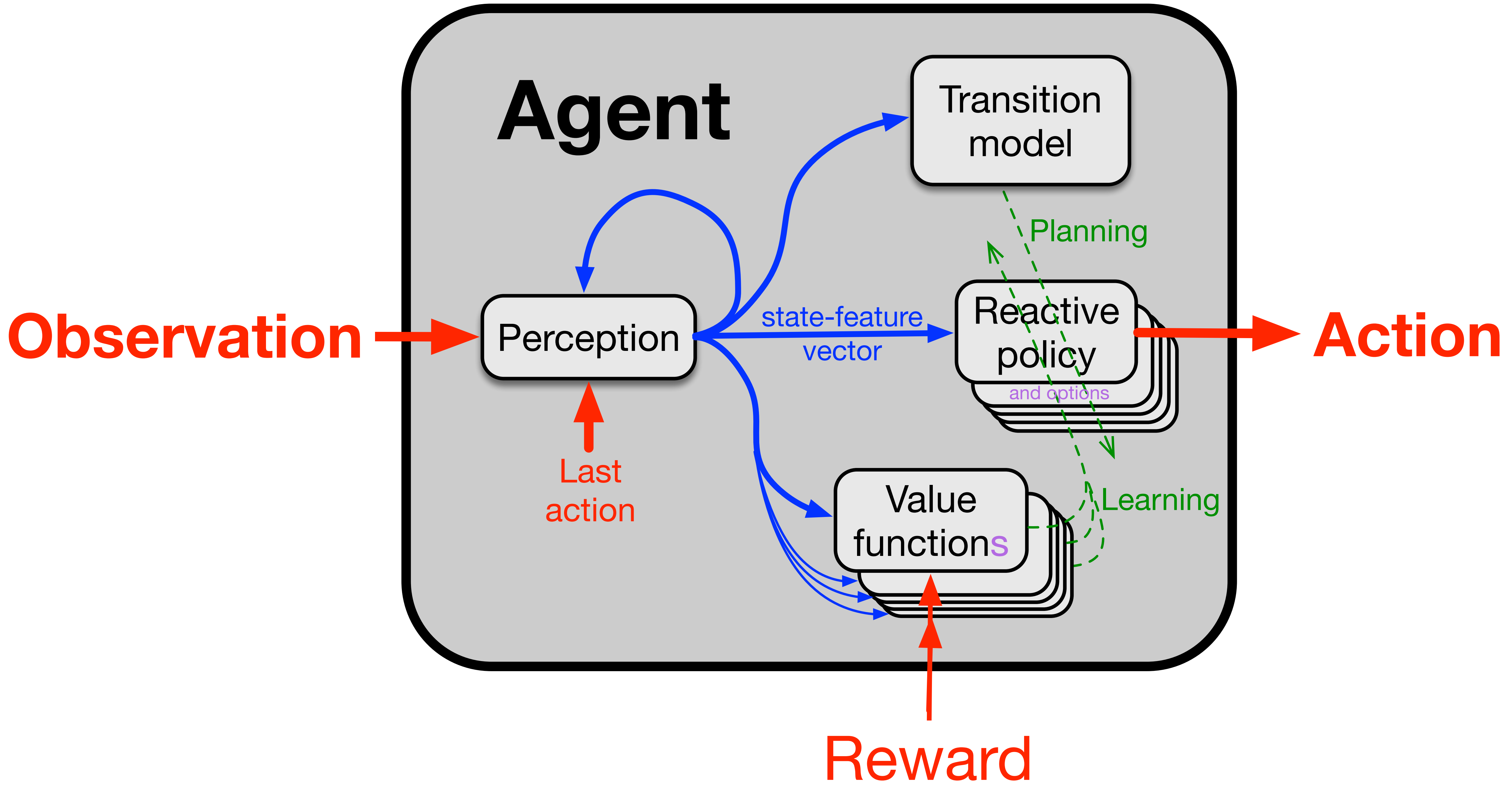
Reactive Policy quickly produces an action appropriate to the state

Value Function evaluates how well things are going, and changes the policy (learning)

Transition model predicts the consequences of alternate actions,
and changes the policy (planning)

The common model does not have options or temporal abstraction

Options-based agents add auxiliary subproblems and options to solve them



The agent comprises 4 kinds of components:

Perception produces the state-feature vector used to solve the problem and all subproblems

Reactive Policy and options quickly produce an action or option appropriate to the state for the main problem and each subproblem

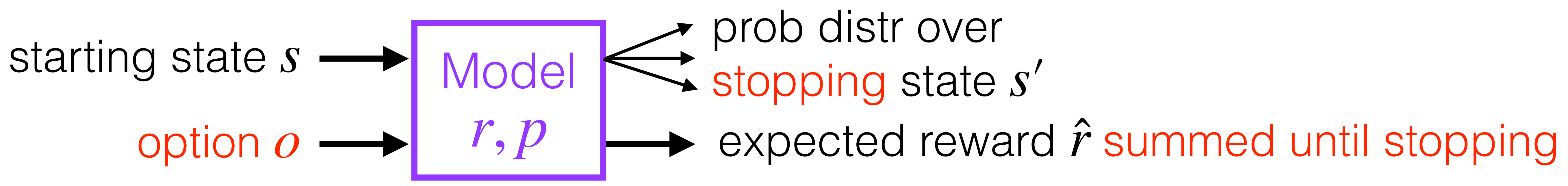
Value Functions, evaluate how well things are going on the main problem & each subproblem, and change the policy and options (learning)

Transition model predicts the consequences of alternate actions and options, and changes the policy over actions and options (planning)

The subproblems could be reward-respecting and feature attaining (as in Oak), or they could maximize auxiliary rewards (as in eigenoptions), or…

Options generalize actions.
Wherever an action appears, you can substitute an option

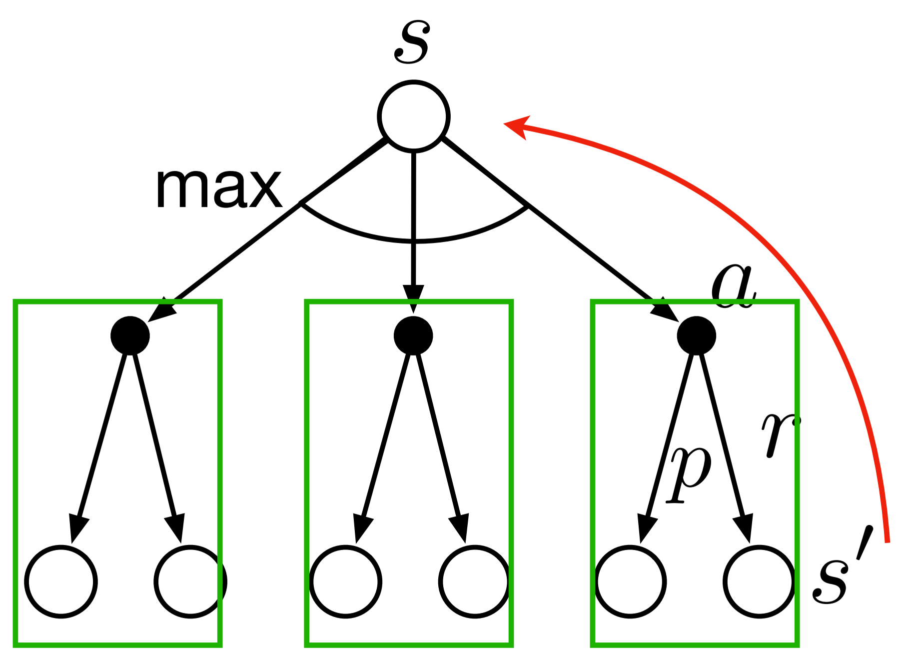
- Action values → option values $q_\pi(s, o)$
- Policies over actions → policies over options $\pi(s) \mapsto o,\ \pi(o \mid s)$
- Action models → option models
    - Transition models over options, assuming execution until termination:



Subproblems generalize the main problem (reward max). Values and policies now indexed by the problem number

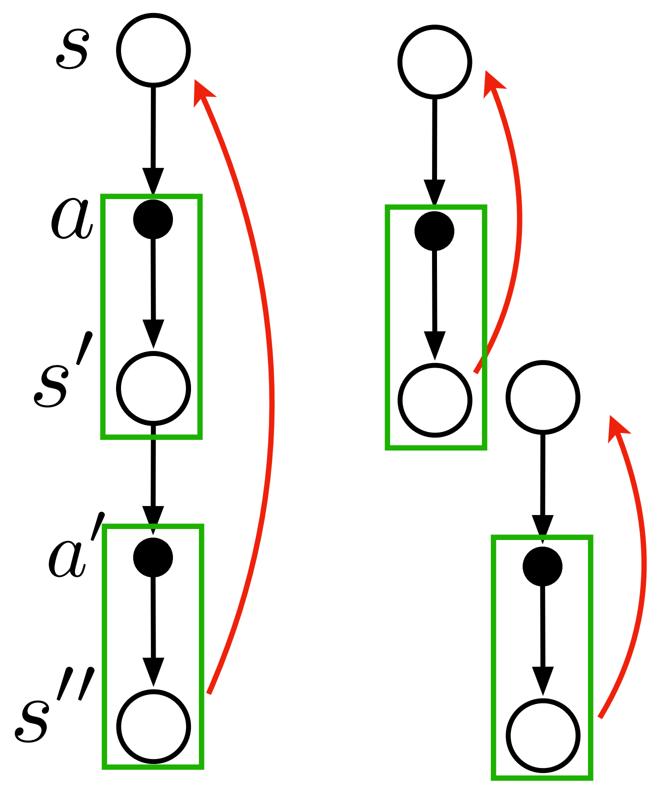
- The main problem (reward max) can be considered problem 0
- State-value functions $v^i_\pi(s)$, approximations $\hat{v}(s, w^i)$
- Option-value functions $q^i_\pi(s, o)$, approximations $\hat{q}(s, o, w^i)$
- Policies $\pi^i(s) \mapsto o, \quad \pi^i(o \mid s)$
- Greedy option for the main problem at t: $O^*_t = \arg\max_o \hat{q}\left(S_t, o, w^0_t\right)$
- In Oak, the subproblems, options, and selected features are 1-to-1, so is and os can both be feature numbers

All planning methods are quite similar



- Planning proceeds by using the model to look ahead from states, imagining something about the future
  - Each imagining from a state-action pair is called a lookahead
- Then, after one or more projections, something computed at the leaves is passed back to the starting state to update its stored policy or value estimate
  - This is called a backup

$$V(s) \leftarrow \max_o \left[r(s,o) \; + \sum_{s'} p(s'|s,o) V(s') \right]$$



- E.g., value iteration

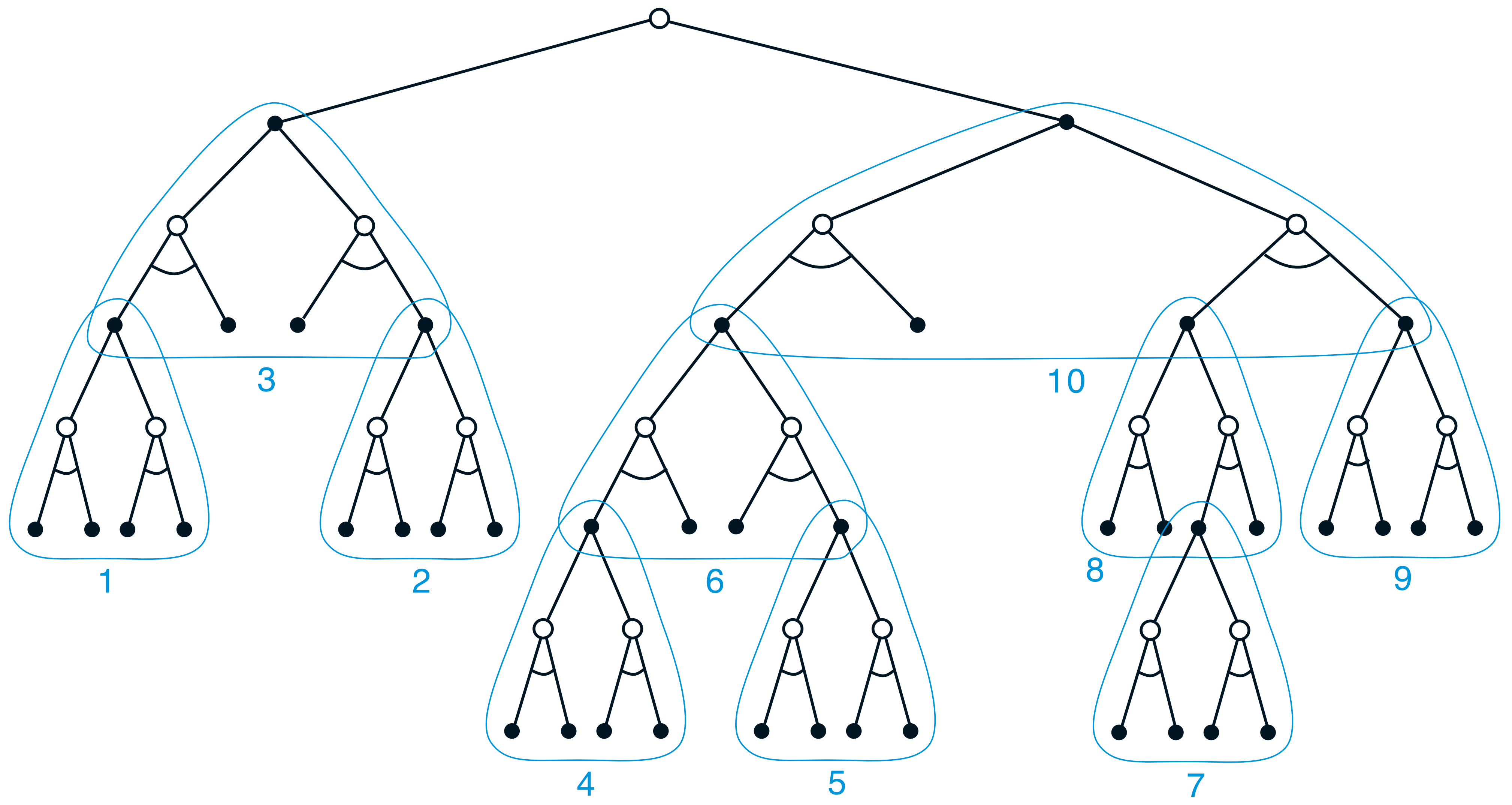
$$Q(s,o) \leftarrow r(s,o) \; + \sum_{s'} p(s'|s,o) \max_{o'} Q(s',o')$$

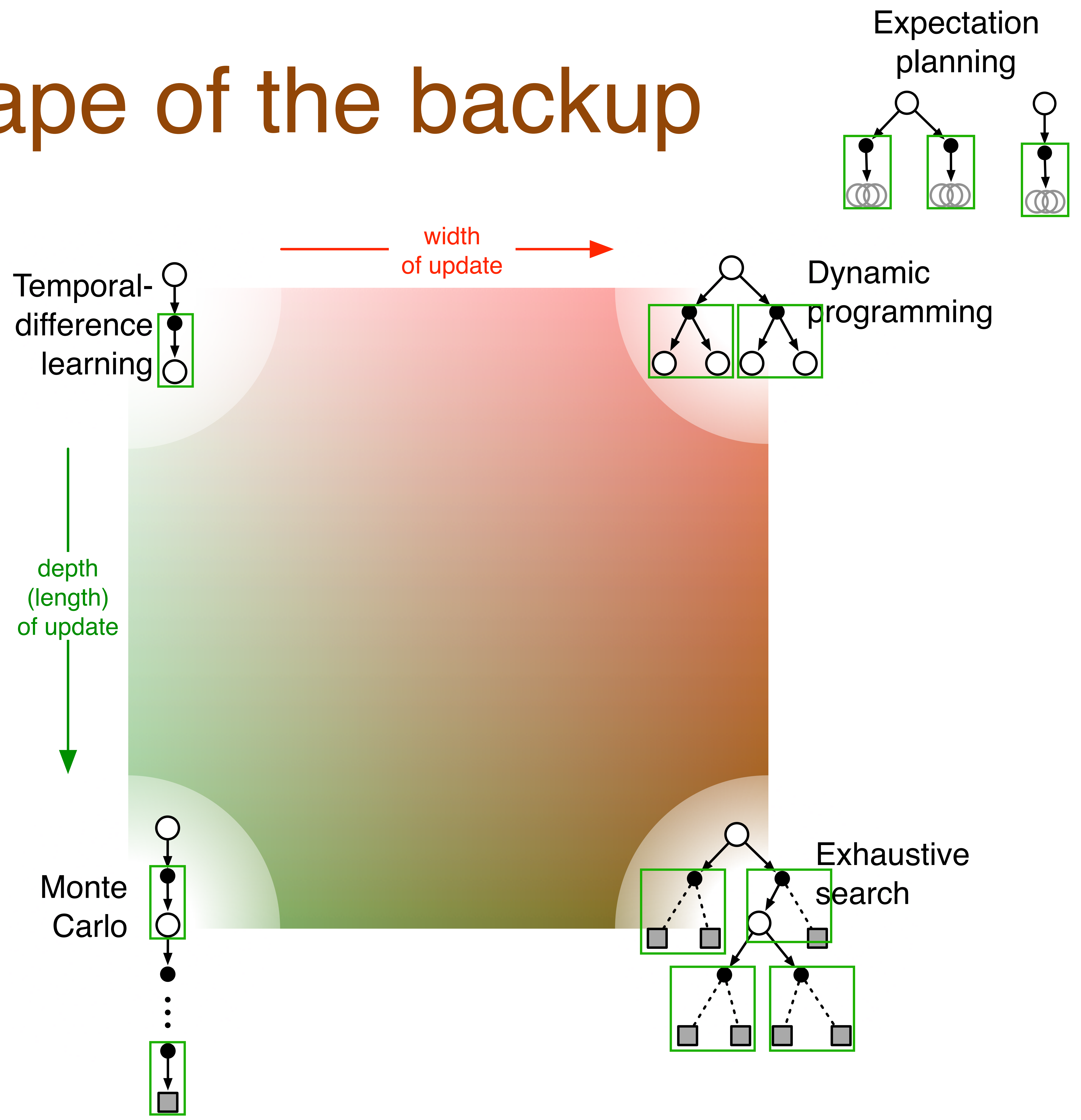
- Or option-value iteration
- There are also half backups (Q from V, and V from Q)

$$V(s) \leftarrow \max_o Q(s,o)$$

Planning shape — the shape of the backup

- Shown here are various shapes of backups, from the RL book, 2nd edition
- Planning backups can also have all these shapes
  - plus all lookaheads could be multi-step, w/options
  - plus all lookaheads could be expectations
- All backup shapes have the same ultimate abilities, given that planning consists of many backups
- E.g., deep search can come from shallow backups:





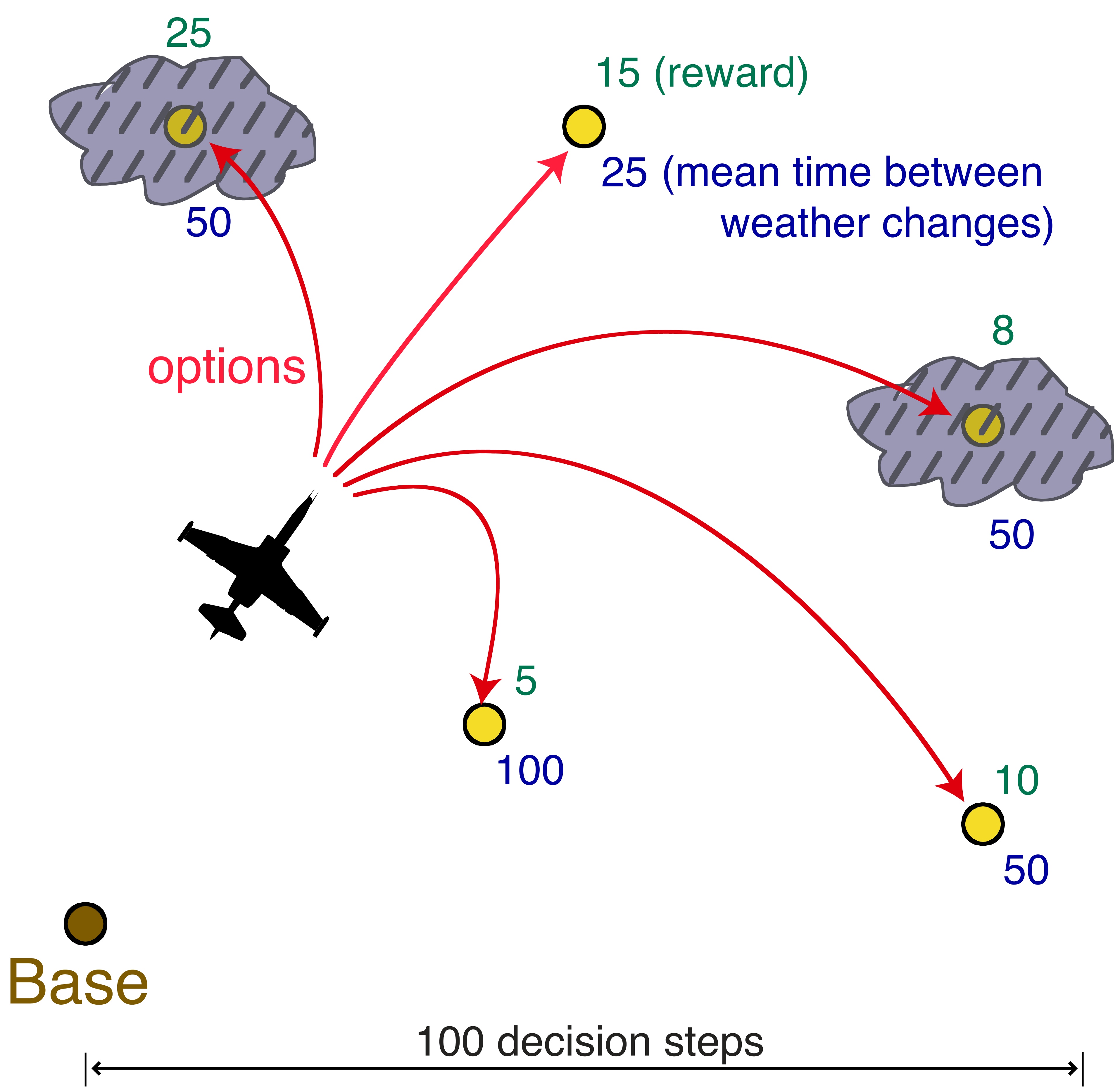
Options Should Be Activated, Not Executed

- The apparent best option at time t is the one with maximal estimated option value
  - $O_t^* = \arg\max_o \hat{q}\left(S_t, o, w_t^0\right)$
- To <u>execute</u> option $O_t^* = (\pi, \gamma)$ at t is to select actions $A_t, A_{t+1}, A_{t+2}, \ldots$ according to π until termination is signalled by γ
  - This gets coherent behavior for the duration of the option's execution
  - But is problematic because the option cannot be interrupted
    - What if $O_{t+1}^* \neq O_t^*$? You are stuck executing O_t^* until its γ says "stop"
- To <u>activate</u> option $O_t^* = (\pi, \gamma)$ at t is to select action A_t according to π, with no commitment to subsequent time steps
  - Termination conditions play no role in activation

An options-based agent has continual processes

- There is a continual planning/learning process that approximates option values and greedy policies over options (for all problems)
    - This process forms non-primitive options as solutions to the subsidiary problems
- On each step, we activate one option, which determines the action taken
    - termination conditions play no role in this
- There is just one transition model for all problems
    - its formation can be considered part of the process in the first bullet—
      if the model is considered a collection of additional predictive subproblems
- Everything that can be planned, can also be learned, and vice versa

Spy Plane Planning Example



- Mission: Fly over (photograph) most valuable sites and return to base
- Stochastic weather affects observability (cloudy or clear) of sites
- Limited fuel
- Planning is intractable with classical optimal control methods
- Temporal scales:
  - Actions: which direction to fly for one step
  - Options: which site to head for
- Options compress space and time
  - Reduce steps from ~600 to ~6
  - Reduce states from ~10^{10} to ~10^{6}
- Value iteration over sites, fuel, weather:

$$\hat{V}(s) \leftarrow \max_{o} \left[r(s,o) + \sum_{s'} p(s' \mid s,o)\hat{V}(s') \right]$$

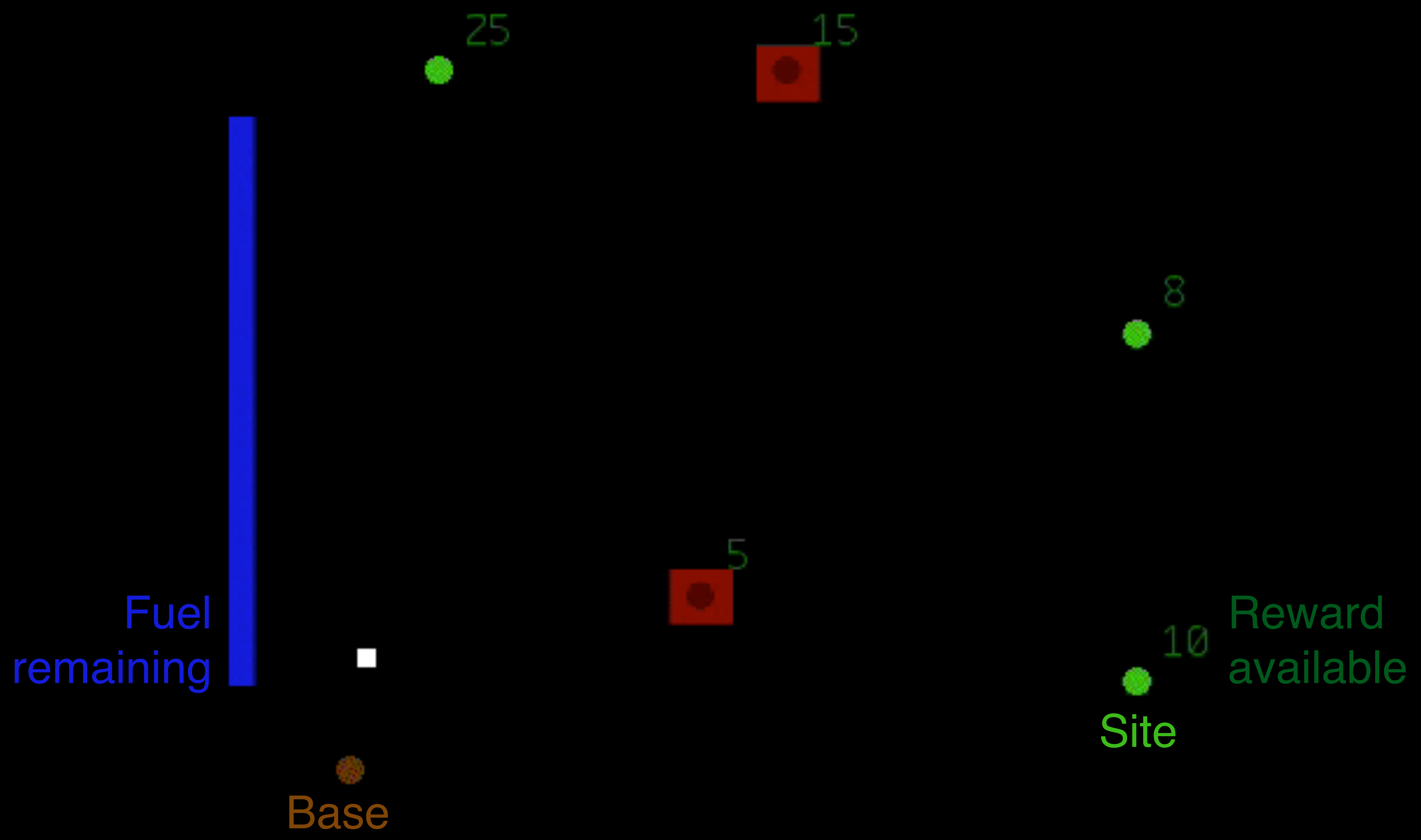
- Activate at t the greedy option according to

$$\hat{Q}(s,o) \doteq r(s,o) + \sum_{s'} p(s' \mid s,o)\hat{V}(s')$$

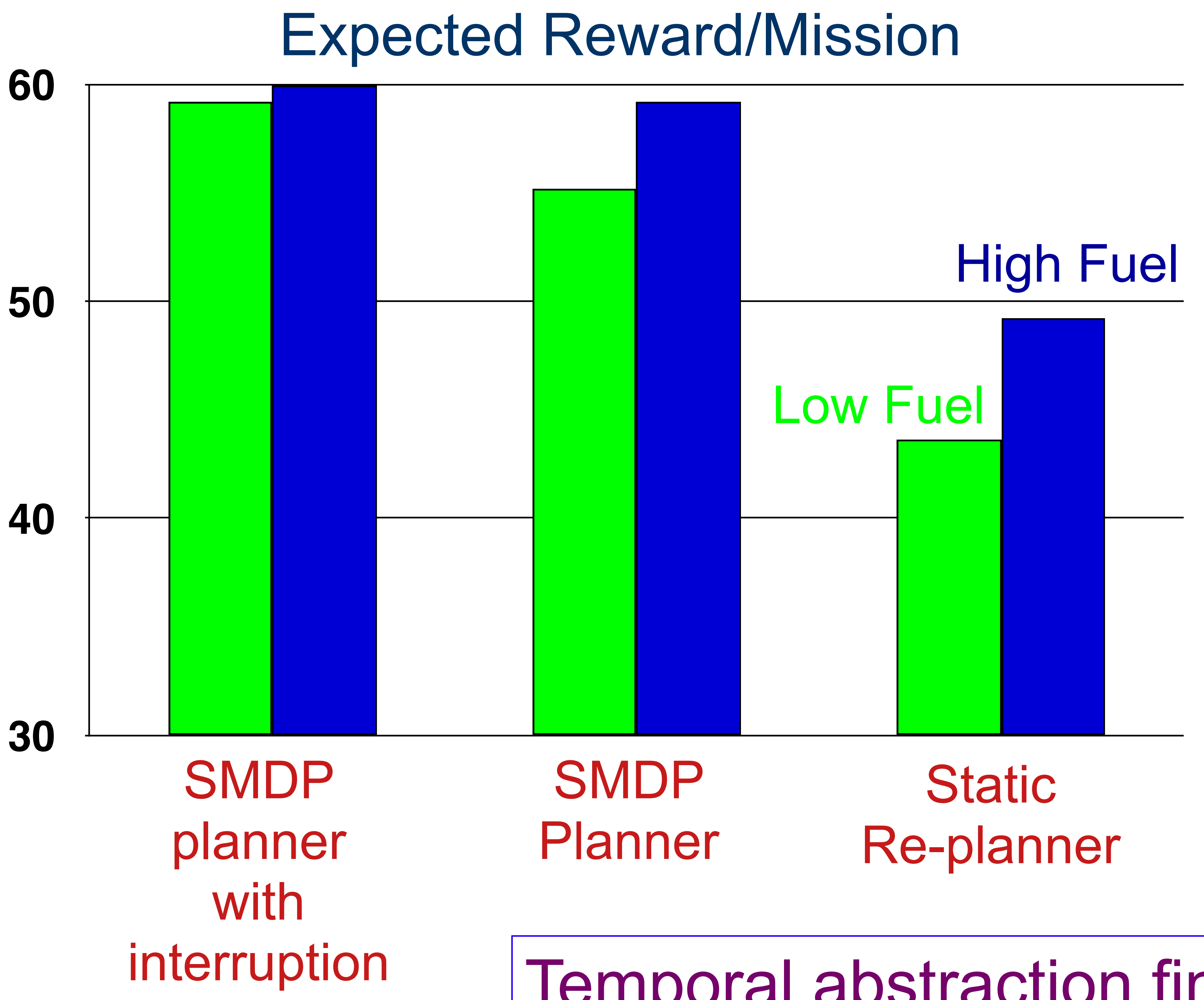
any state ~10^{10}

sites, fuel, weather only ~10^{6}

Spy Plane Action Selection by Option Activation



Spy Plane Example (Results)



Temporal abstraction finds better approximation than static planner, with little more computation than SMDP planner

- SMDP planner:
  - Assumes options followed to completion
  - Plans optimal SMDP solution
- SMDP planner with interruption
  - Plans as if options must be followed to completion
  - But actually takes them for only one step
  - Re-picks a new option on every step
- Static planner:
  - Assumes weather will not change
  - Plans optimal tour among clear sites
  - Re-plans whenever weather changes

Conclusions

- Never execute options. Follow them while they are active
- How to do planning is clear and flexible; all flows to $\hat{v}^i$, $\hat{q}^i$, and π^i
- The bottleneck is still Steps 1 & 2 of The Alberta Plan
- There are two areas that may be ripe for immediate further research
  - Establishing the best subproblems?
  - Exploration via “learning feels good” internal reward
  - Laying out a multi-option pattern of intention

What about exploration?

- When time steps are short, you don’t want to be ε-greedy
- I recommend greedy action selection and online learning
  - together with giving yourself reward when your learning makes progress
    - a form of “intrinsic” reward
    - Pioneered by Linke, Ady, White, Degris & White (2020) “Adaptive Behavior via Intrinsic Reward”
  - First idea: “Learning progress” = changes in weights
  - 2nd idea: “learning prog” = changes in weights after step-size optimization
  - 3rd idea: Some weights are more important (for reward) than others
- This idea has great potential together with state construction

What about the option keyboard?

- I cannot yet justify talk of an “action keyboard”
  - the idea is not yet clear and clearly useful to me
- But there are related ideas that seem important
- First, consider deeply hierarchical policies over options
  - they might involve many steps before grounding out into actions
    - $\pi(S_t)$ selects o_1, but then the π of o_1 selects o_2, whose π selects o_3, …
  - $flat(\pi) : \mathcal{S} \to \mathcal{A} \quad flat(\pi)(s) \doteq \begin{cases} \pi(s) & \text{if } \pi(s) \text{ is an action} \\ flat(\pi(\pi(s))) & \text{if } \pi(s) \text{ is an option} \end{cases}$
- In such a case, perhaps you want o_1, o_2, and o_3 to all be represented, to all “light up”, but perhaps over several time steps with a similar state

Conclusions

- Never execute options. Follow them while they are active
- How to do planning is clear and flexible; all flows to $\hat{v}^i$, $\hat{q}^i$, and π^i
- The bottleneck is still Steps 1 & 2 of The Alberta Plan
- There are two areas that may be ripe for immediate further research
  - Establishing the best subproblems?
  - Exploration via "learning feels good" internal reward
  - Laying out a multi-option pattern of intention

Thank you for your attention