

16-350

Planning Techniques for Robotics

Search Algorithms:

Markov Property, Dependent variables

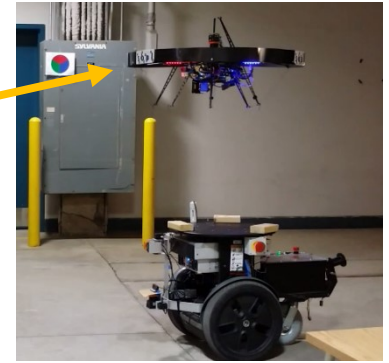
Maxim Likhachev

Robotics Institute

Carnegie Mellon University

Consider Planning with Battery Constraint

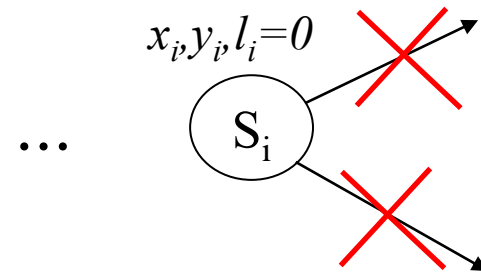
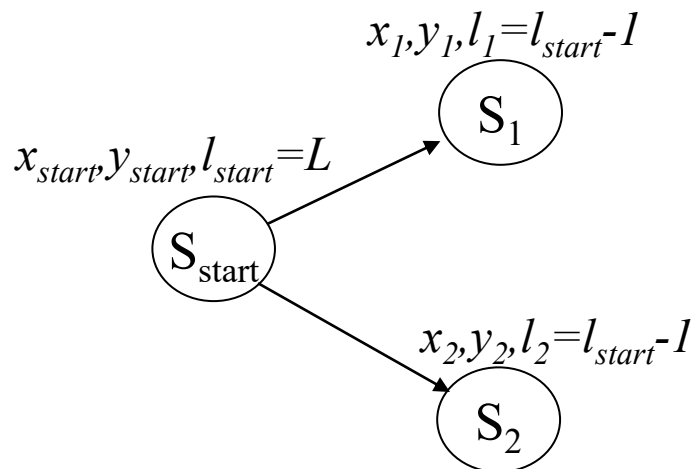
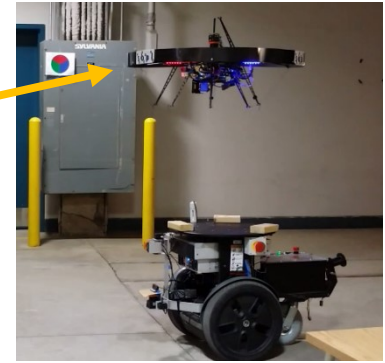
- Suppose we are planning 2D (x,y) path for UAV
 - want a collision-free path to $s_{goal} = (x_{goal}, y_{goal})$
 - want to minimize some cost function associated with each transition (for example, minimize the risk of flying close to people)
 - subject to the trajectory being feasible given the UAV battery level L



*What should be the variables defining each state
(i.e., dimensions of the search)?*

Consider Planning with Battery Constraint

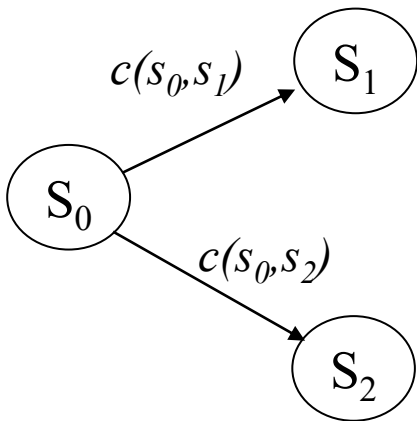
- Suppose we are planning 2D (x,y) path for UAV
 - want a collision-free path to $s_{goal} = (x_{goal}, y_{goal})$
 - want to minimize some cost function associated with each transition (for example, minimize the risk of flying close to people)
 - subject to the trajectory being feasible given the UAV battery level L
 - **Planning needs to be in (x,y,l) , where l is the remaining battery level**



states with battery level 0 have no successors

Markov Property

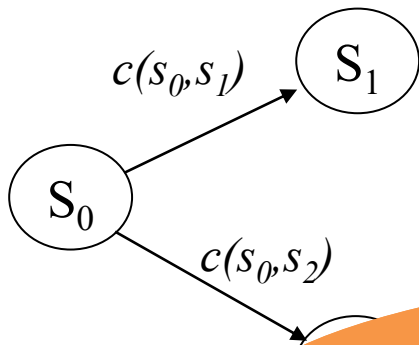
- *Cost and Set of Successors needs to depend ONLY on the current state (no dependence on the history of the path leading up to it!)*



*for all states s : $\text{succ}(s) = \text{function of } s$
for all s' in $\text{succ}(s)$: $c(s, s') = \text{function of } s, s'$*

Markov Property

- *Cost and Set of Successors needs to depend ONLY on the current state (no dependence on the history of the path leading up to it!)*



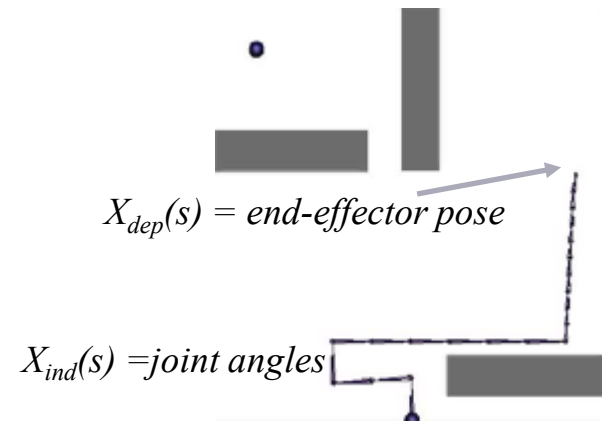
*for all states s : $\text{succ}(s) = \text{function of } s$
for all s' in $\text{succ}(s)$: $c(s, s') = \text{function of } s, s'$*

Clearly true in an **explicit** (given) graph

Can be violated in **implicit** (dynamically generated) graphs, where $\text{succ}(s)$ and $c(s, s')$ are computed on-the-fly as a function of s ,
when using dependent variables

Independent vs. Dependent Variables

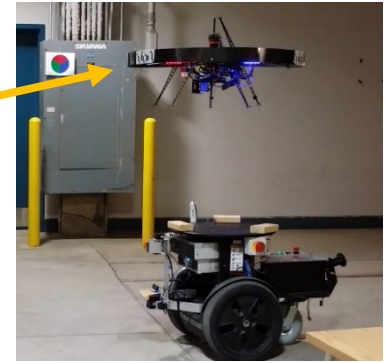
- $X(s)$ – variables associated with s
- $X(s) = \{X_{ind}(s), X_{dep}(s)\}$
- $X_{ind}(s)$ – independent variables
- $X_{dep}(s)$ – dependent variables



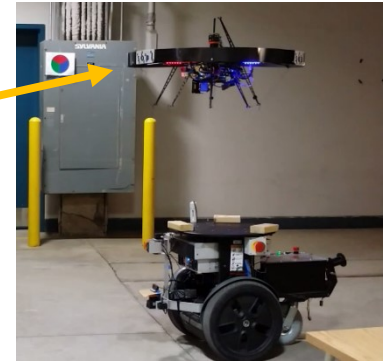
- **Independent Variables** are used to define state s
 - *two states s and s' are considered to be the same state if and only if $X_{ind}(s) = X_{ind}(s')$*
- **Dependent Variables** often used to help with computing cost or list of successor states
 - *if for all s , $X_{dep}(s) = f(X_{ind}(s))$ (that is, only depends on independent variables, then Markov Property holds true)*
 - **Sometimes however, $X_{dep}(s)$ is computed based on the path leading up to $X_{ind}(s)$**

Consider Planning with Battery Constraint

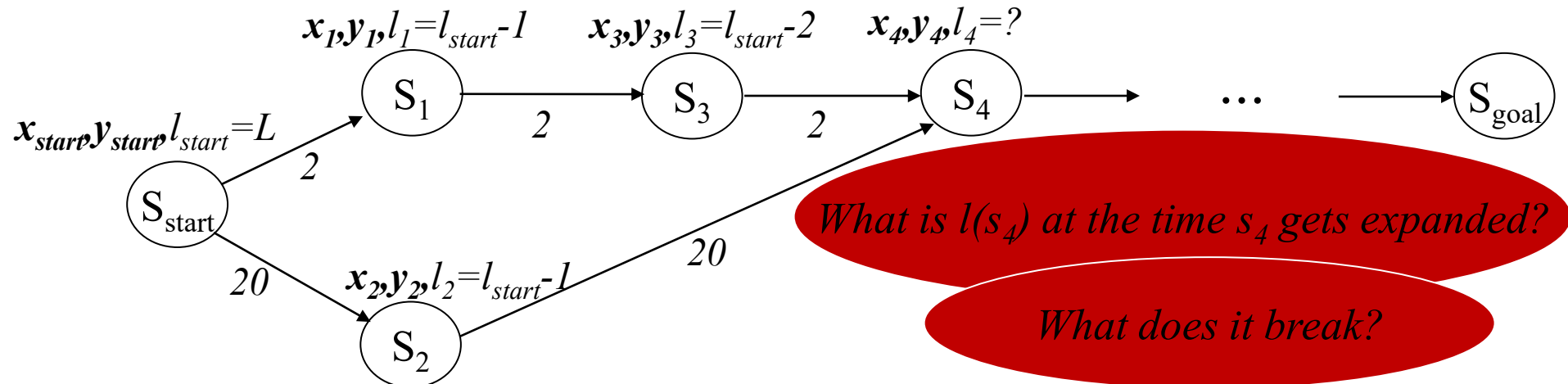
- Suppose we are planning 2D (x,y) path for UAV
 - want a collision-free path to $s_{goal} = (x_{goal}, y_{goal})$
 - want to minimize some cost function associated with each transition (for example, minimize the risk of flying close to people)
 - subject to the trajectory being feasible given the UAV battery level L
 - **Consider $X_{ind}=(x,y)$, $X_{dep}=(l)$, where l is the remaining battery level**



Consider Planning with Battery Constraint

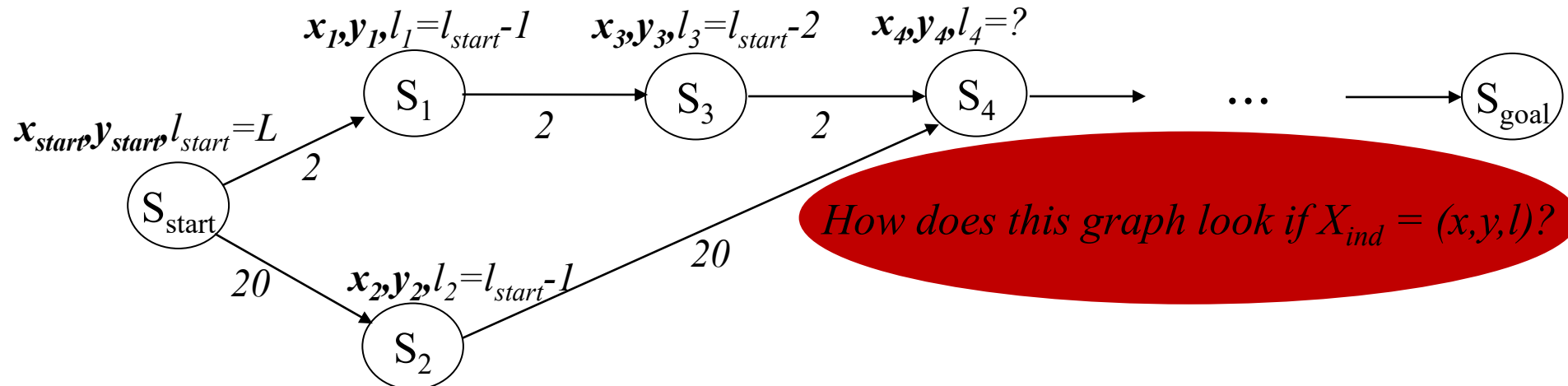
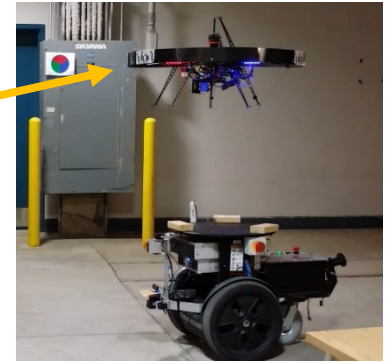


- Suppose we are planning 2D (x,y) path for UAV
 - want a collision-free path to $s_{goal} = (x_{goal}, y_{goal})$
 - want to minimize some cost function associated with each transition (for example, minimize the risk of flying close to people)
 - subject to the trajectory being feasible given the UAV battery level L
 - **Consider $X_{ind}=(x,y)$, $X_{dep}=(l)$, where l is the remaining battery level**



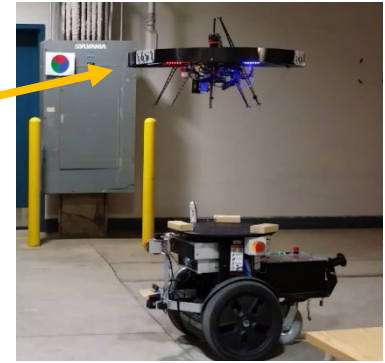
Consider Planning with Battery Constraint

- Suppose we are planning 2D (x,y) path for UAV
 - want a collision-free path to $s_{goal} = (x_{goal}, y_{goal})$
 - want to minimize some cost function associated with each transition (for example, minimize the risk of flying close to people)
 - subject to the trajectory being feasible given the UAV battery level L
 - **Consider $X_{ind}=(x,y)$, $X_{dep}=(l)$, where l is the remaining battery level**



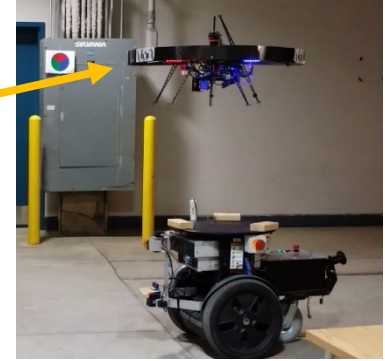
Back to Planning with Battery Constraint

- Suppose we are planning 2D (x,y) path for UAV
 - want a collision-free path to $s_{goal} = (x_{goal}, y_{goal})$
 - assume cost function is battery consumption
 - subject to the trajectory being feasible given the UAV battery level L
 - **Consider $X_{ind}=(x,y)$, $X_{dep}=(l)$, where l is the remaining battery level**

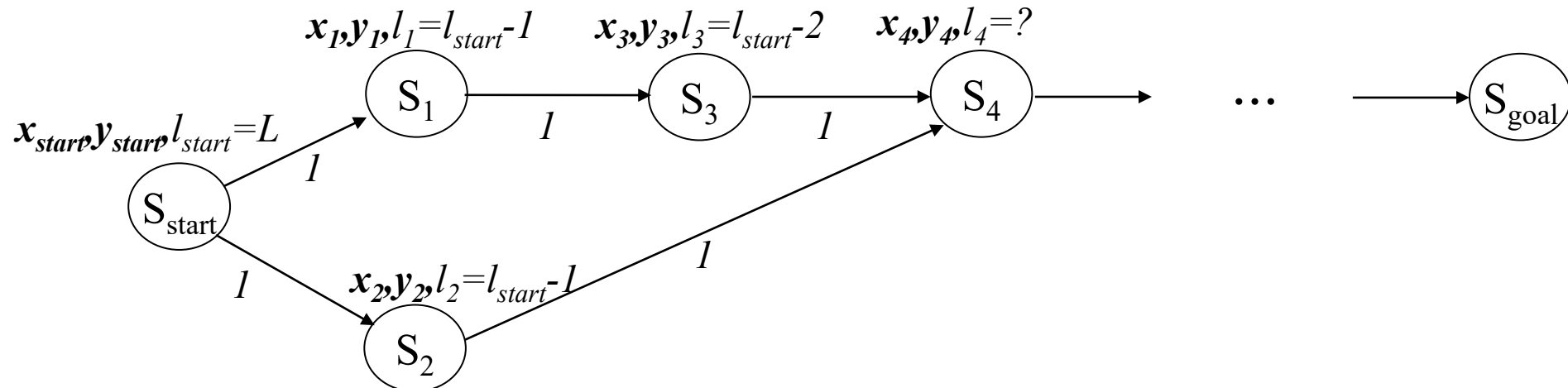


Is it incomplete?

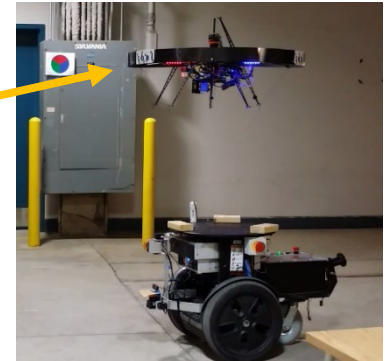
Back to Planning with Battery Constraint



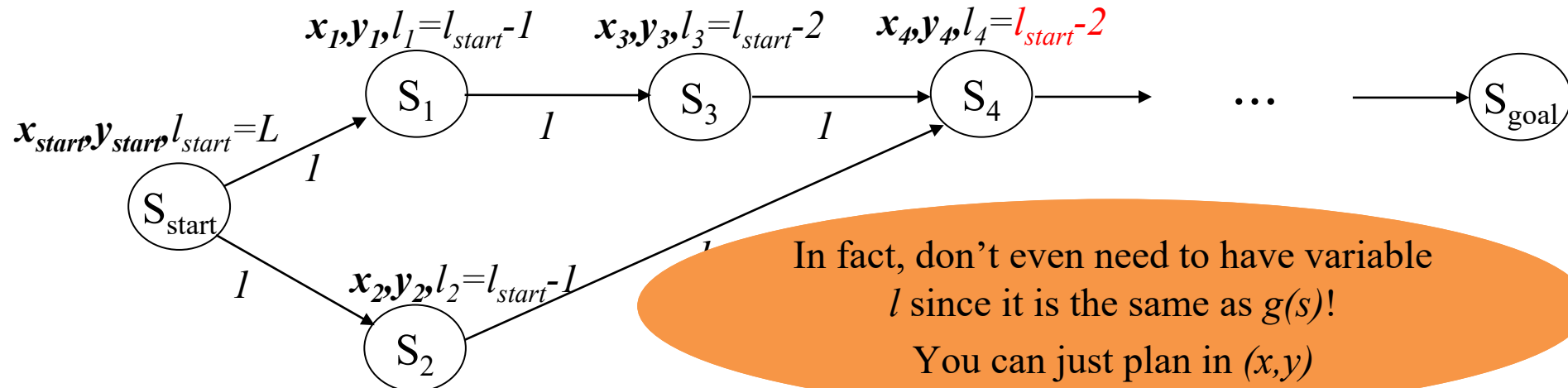
- Suppose we are planning 2D (x,y) path for UAV
 - want a collision-free path to $s_{goal} = (x_{goal}, y_{goal})$
 - assume cost function is battery consumption
 - subject to the trajectory being feasible given the UAV battery level L
 - **Consider $X_{ind}=(x,y)$, $X_{dep}=(l)$, where l is the remaining battery level**



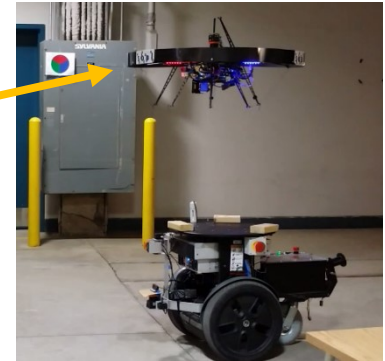
Back to Planning with Battery Constraint



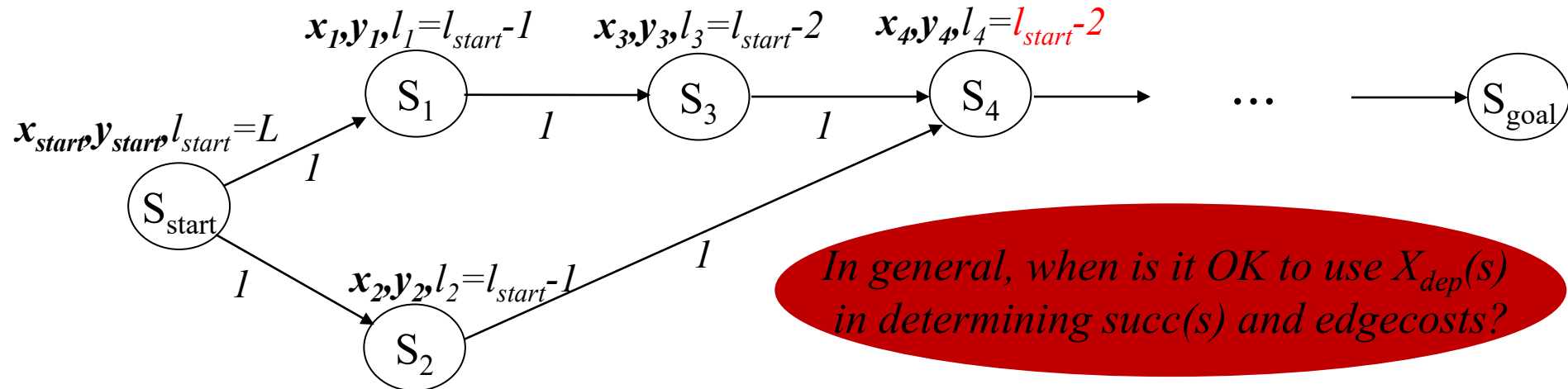
- Suppose we are planning 2D (x,y) path for UAV
 - want a collision-free path to $s_{goal} = (x_{goal}, y_{goal})$
 - assume cost function is battery consumption
 - subject to the trajectory being feasible given the UAV battery level L
 - **Consider $X_{ind}=(x,y)$, $X_{dep}=(l)$, where l is the remaining battery level**



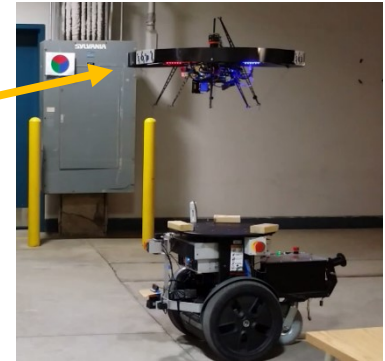
Back to Planning with Battery Constraint



- Suppose we are planning 2D (x,y) path for UAV
 - want a collision-free path to $s_{goal} = (x_{goal}, y_{goal})$
 - assume cost function is battery consumption
 - subject to the trajectory being feasible given the UAV battery level L
 - Consider $X_{ind}=(x,y)$, $X_{dep}=(l)$, where l is the remaining battery level

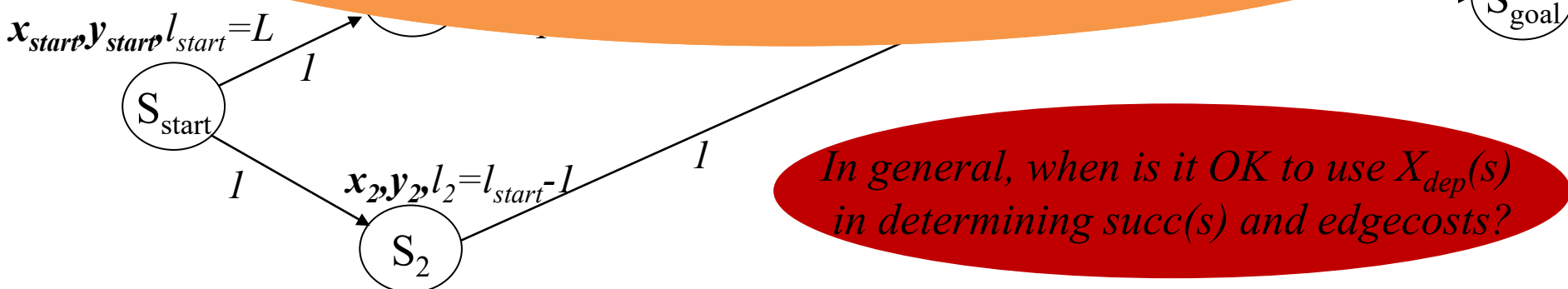


Back to Planning with Battery Constraint



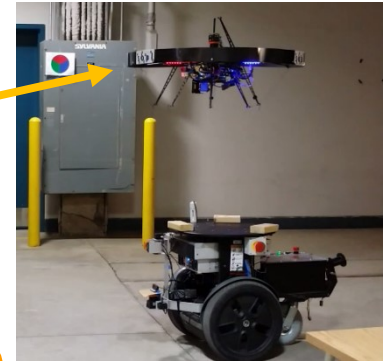
- Suppose we are planning 2D (x,y) path for UAV
 - want a collision-free path to $s_{goal} = (x_{goal}, y_{goal})$
 - assume cost function is battery consumption
 - subject to the constraint that the path will not drop the UAV battery level L

Whenever you can guarantee that for any state s :
 if we have two paths $\pi_1(s_{start}, s)$ and $\pi_2(s_{start}, s)$ s.t. $c(\pi_1) \geq c(\pi_2)$,
 then it implies that $c_1(s, s') \geq c_2(s, s')$,
 where $c_i(s, s')$ – cost of a least-cost path from s to s' after s is
 reached from s_{start} via path π_i



In general, when is it OK to use $X_{dep}(s)$ in determining $succ(s)$ and edgecosts?

Back to Planning with Battery Constraint

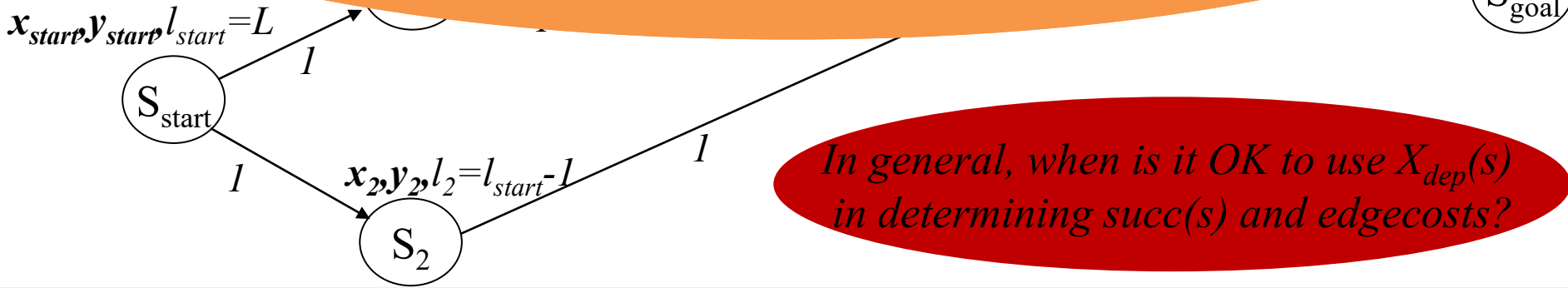


- Suppose we are planning 2D (x,y) path for UAV

- want a path π from s_{start} to s_{goal} such that $c(\pi)$ is minimized
- assume $c(\pi)$ is the sum of edge costs
- subject to the constraint that the UAV battery level L is not exceeded

Assuming we are running optimal search (such as A).*

Whenever you can guarantee that for any state s :
 if we have two paths $\pi_1(s_{start}, s)$ and $\pi_2(s_{start}, s)$ s.t. $c(\pi_1) \geq c(\pi_2)$,
 then it implies that $c_1(s, s') \geq c_2(s, s')$,
 where $c_i(s, s')$ – cost of a least-cost path from s to s' after s is reached from s_{start} via path π_i



In general, when is it OK to use $X_{dep}(s)$ in determining $succ(s)$ and edge costs?

Dominance Relationship

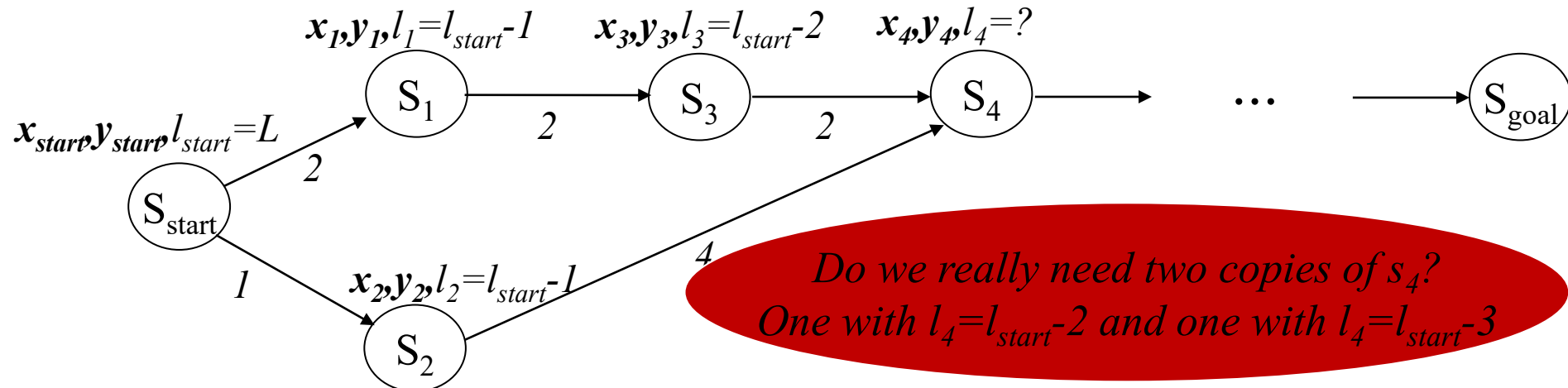
- Suppose we are planning 2D (x,y) path for UAV

- want a collision-free path to $s_{goal} = (x_{goal}, y_{goal})$
- want to minimize some cost function associated with each transition (for example, minimize the path length)

- subject to

What are the general conditions for pruning “dominated” states?

- Consider $\mathbf{X}_{ind} = (x, y, l)$



*Do we really need two copies of s_4 ?
One with $l_4 = l_{start} - 2$ and one with $l_4 = l_{start} - 3$*

Dominance Relationship

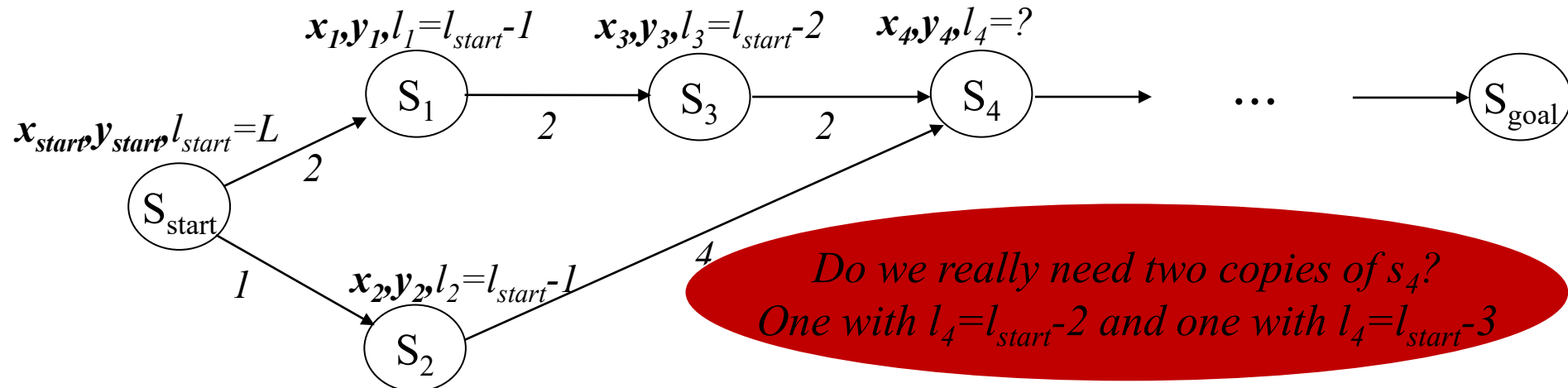
if $(g(s) \leq g(s'))$ and s **dominates** s' , then s' can be pruned by search
 s dominates s' implies s' cannot be part of a solution that is better than the solution from s

- want to minimize $g(s)$ (cost associated with each transition (for example, minimize the number of nodes visited))

- subject

What are the general conditions for pruning “dominated” states?

- Consider $X_{ind}=(x,y,l)$



A* Search with Dominance Check

Main function

$g(s_{start}) = 0$; all other g -values are infinite; $OPEN = \{s_{start}\}$;

ComputePath();

publish solution;

ComputePath function

while(s_{goal} is not expanded and $OPEN \neq \emptyset$)

 remove s with the smallest $[f(s) = g(s) + h(s)]$ from $OPEN$;

 insert s into $CLOSED$;

 for every successor s' of s such that s' not in $CLOSED$

 if $g(s') > g(s) + c(s, s')$

$g(s') = g(s) + c(s, s')$;

if there exists state s'' such that $(g(s'') \leq g(s'))$ AND s'' dominates s'

continue; //skip inserting state s' into $OPEN$, i.e., prune

 insert s' into $OPEN$;

What You Should Know...

- Dependent vs. Independent variables
- Definition of Markov Property and what happens if it is violated
- Dominance relationship and how it can be used within search
- Understand what planning problems have a Dominance relationship