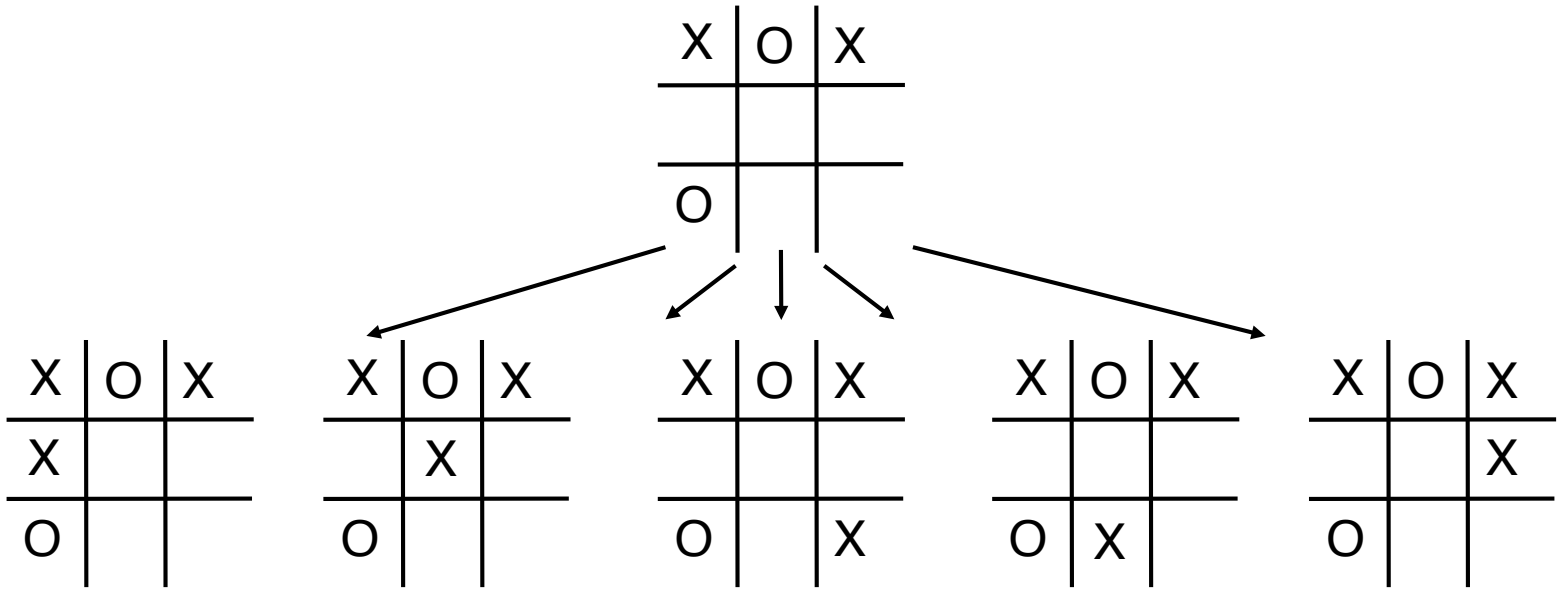
General Game Playing
Statistical Evaluation

Michael Genesereth
Computer Science Department
Stanford University

Evaluation of Non-Terminal States



How do we evaluate non-terminal states?

Monte Carlo

Basic Idea: To estimate the value of a state/node, play out a few games and assign the average payoff.

(1) Simulate "depth charges" to game end, making *random* choices for moves of all players.

(3) Assign expected utility by summing utilities of each depth charge and dividing by number of depth charges.

Programme

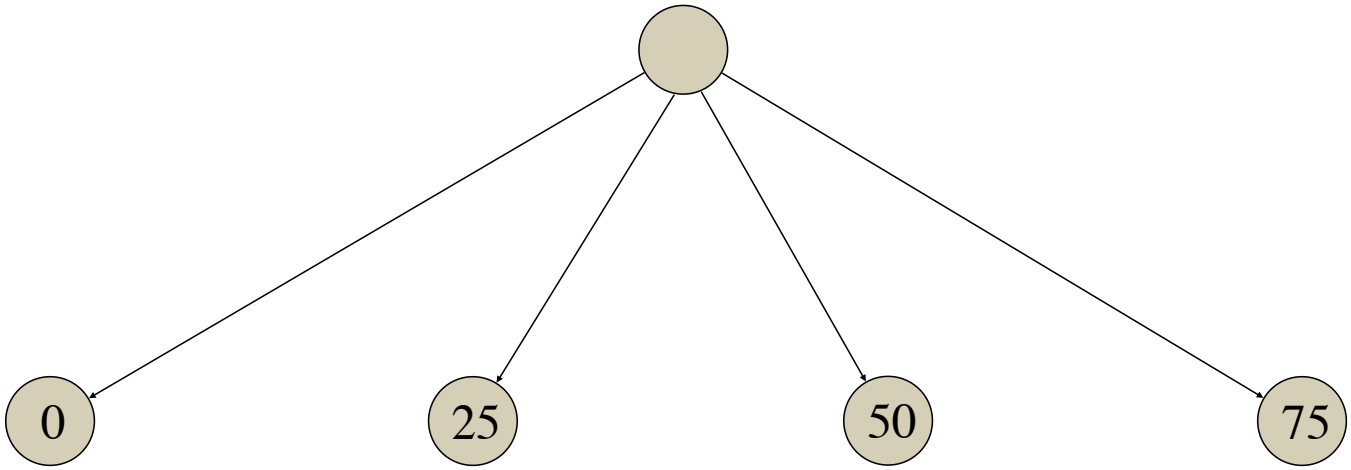
Monte Carlo Search
- One Step
- Monte Carlo Evaluation

Monte Carlo Tree Search
- Persistent Tree Search
- Monte Carlo Evaluation
- Exploration and Exploitation
- Upper Confidence Bounds on Trees (UCT)

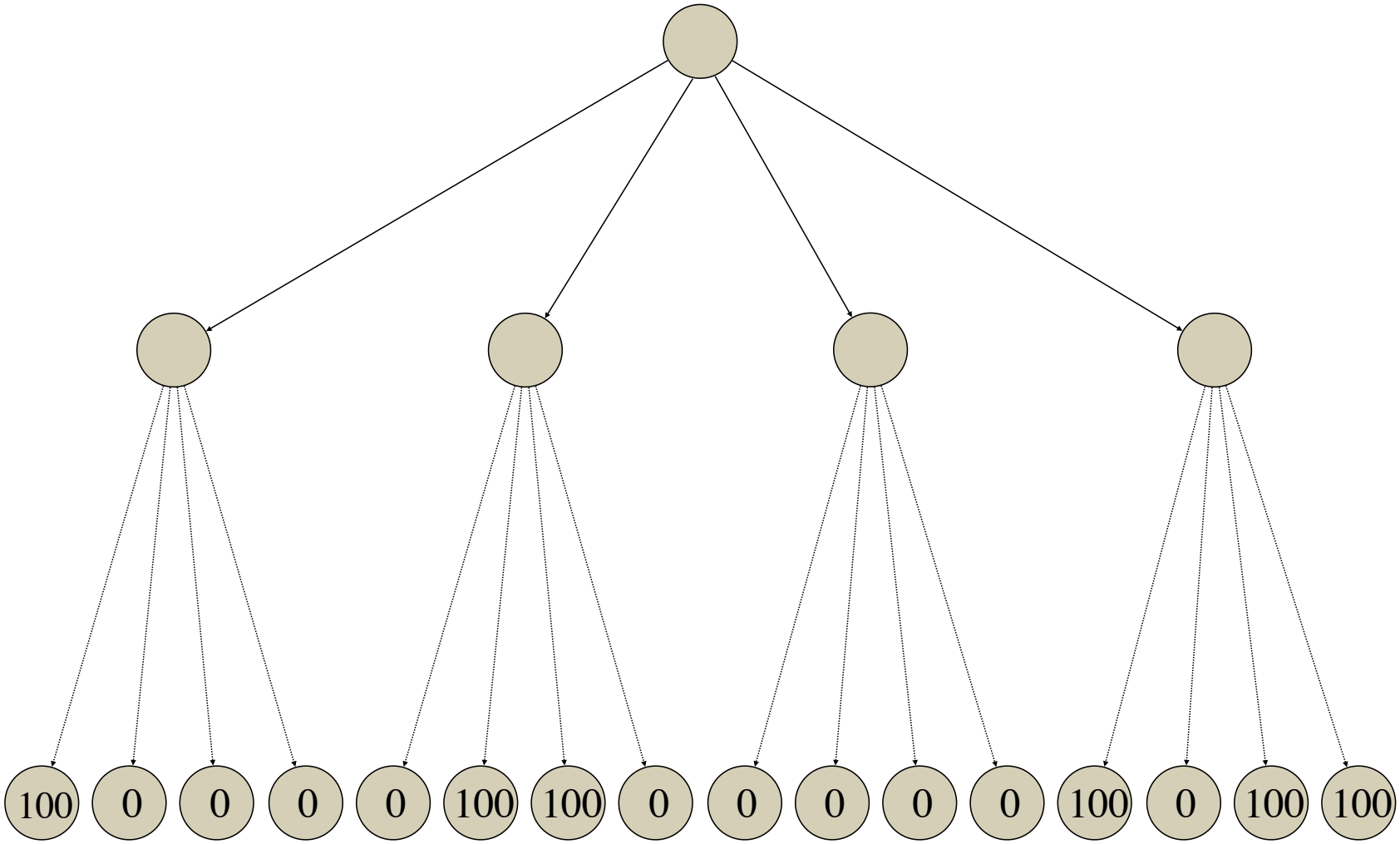
Techniques to speed up / get more Depth Charges
- Indexing
- Differential Update

Monte Carlo Search

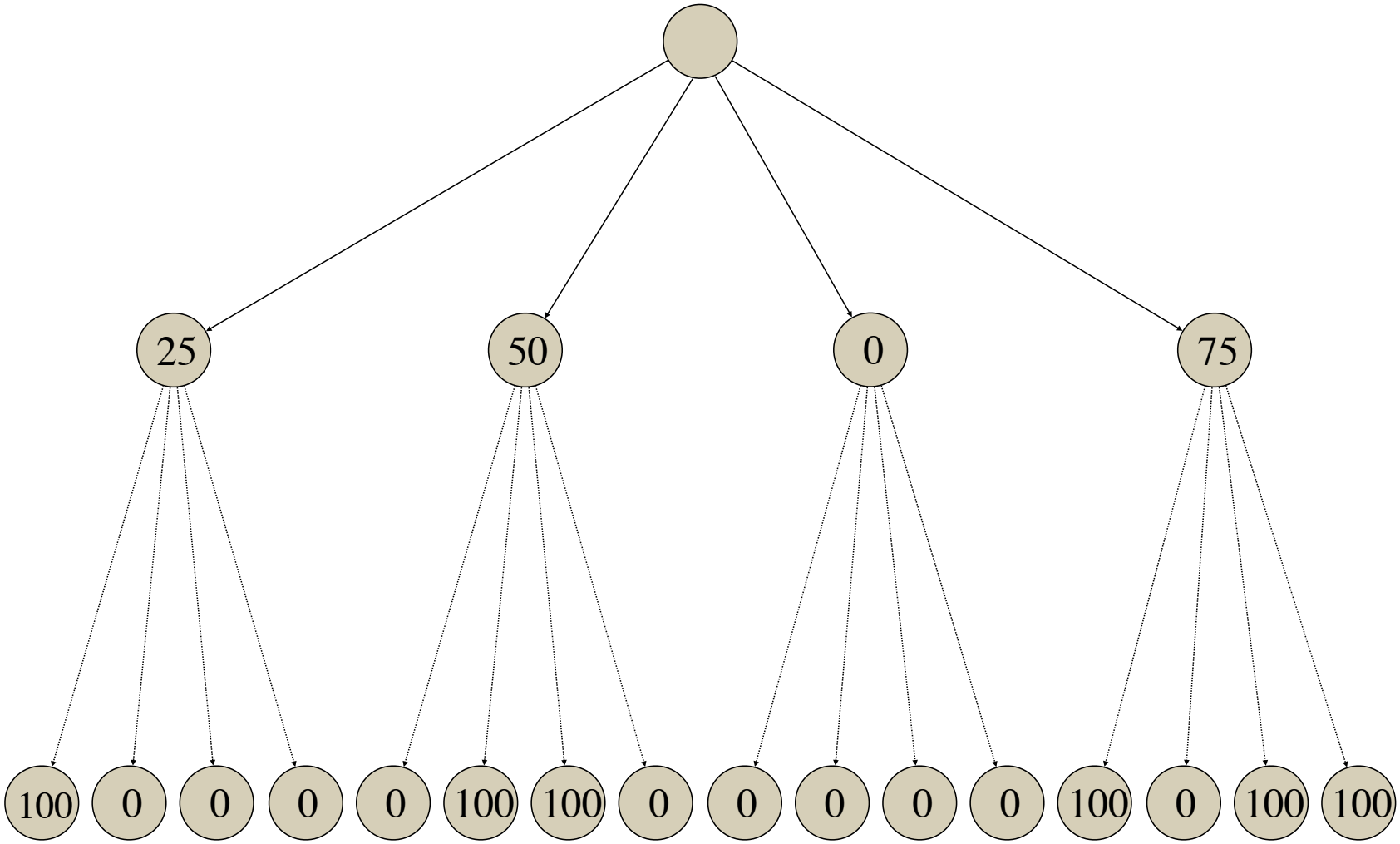
One Step



One Step with MCS



One Step MCS



playmcs

```
function play (move)
 {if (move!==nil) {state = simulate(move,state,library)};
  if (findcontrol(state,library)!==role) {return false};
  return playmcs(role)}

function playmcs ()
 {var deadline = Date.now()+Math.floor(playclock-2)*1000;
  var actions = shuffle(findlegals(state,library));
  if (actions.length===0) {return false};
  if (actions.length===1) {return actions[0]};
  var scores = [];
  var visits = [];
  for (var i=0; i<actions.length; i++)
      {states[i] = simulate(actions[i],state,library);
       scores[i] = 0;
       visits[i] = 0};
  explore(states,scores,visits,deadline);
  var move = selectaction(actions,scores,visits);
  return move}
```

explore

```
function explore (states,scores,visits,deadline)
 {var index = 0;
  while (Date.now()<deadline)
   {if (index>=states.length) {index = 0};
    var result = depthcharge(states[index]);
    scores[index] = scores[index] + result;
    visits[index] = visits[index] + 1;
    depthcharges++; index++};
  return true}
```

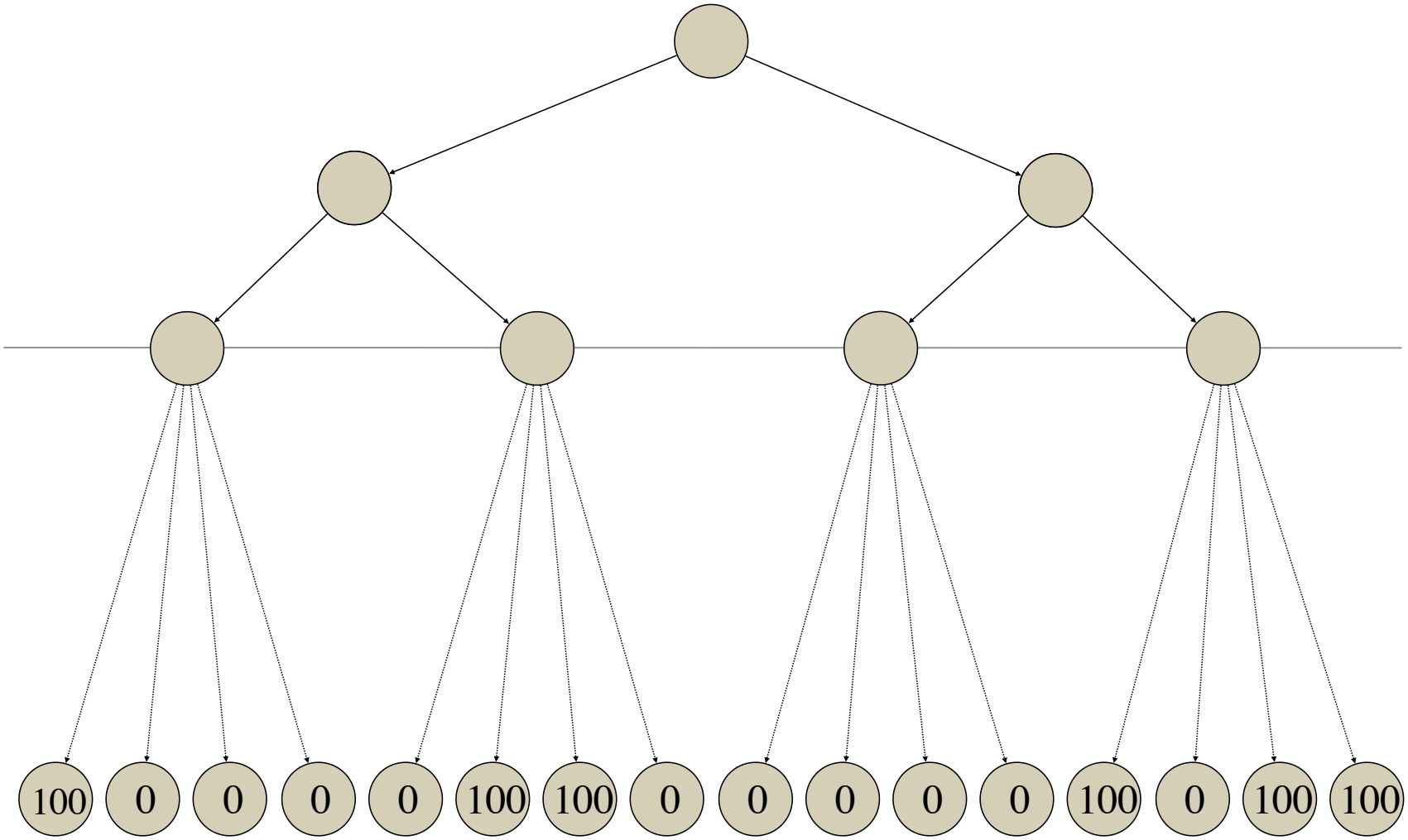
depthcharge

```
function depthcharge (state)
 {if (findterminalp(state,ruleset))
     {return findreward(role,state,ruleset)}*1;
  var actions = findlegals(state,library);
  if (actions.length===0) {return 0};
  var best = randomindex(actions.length);
  var newstate = simulate(actions[best],state,library);
  return depthcharge(newstate)}
```

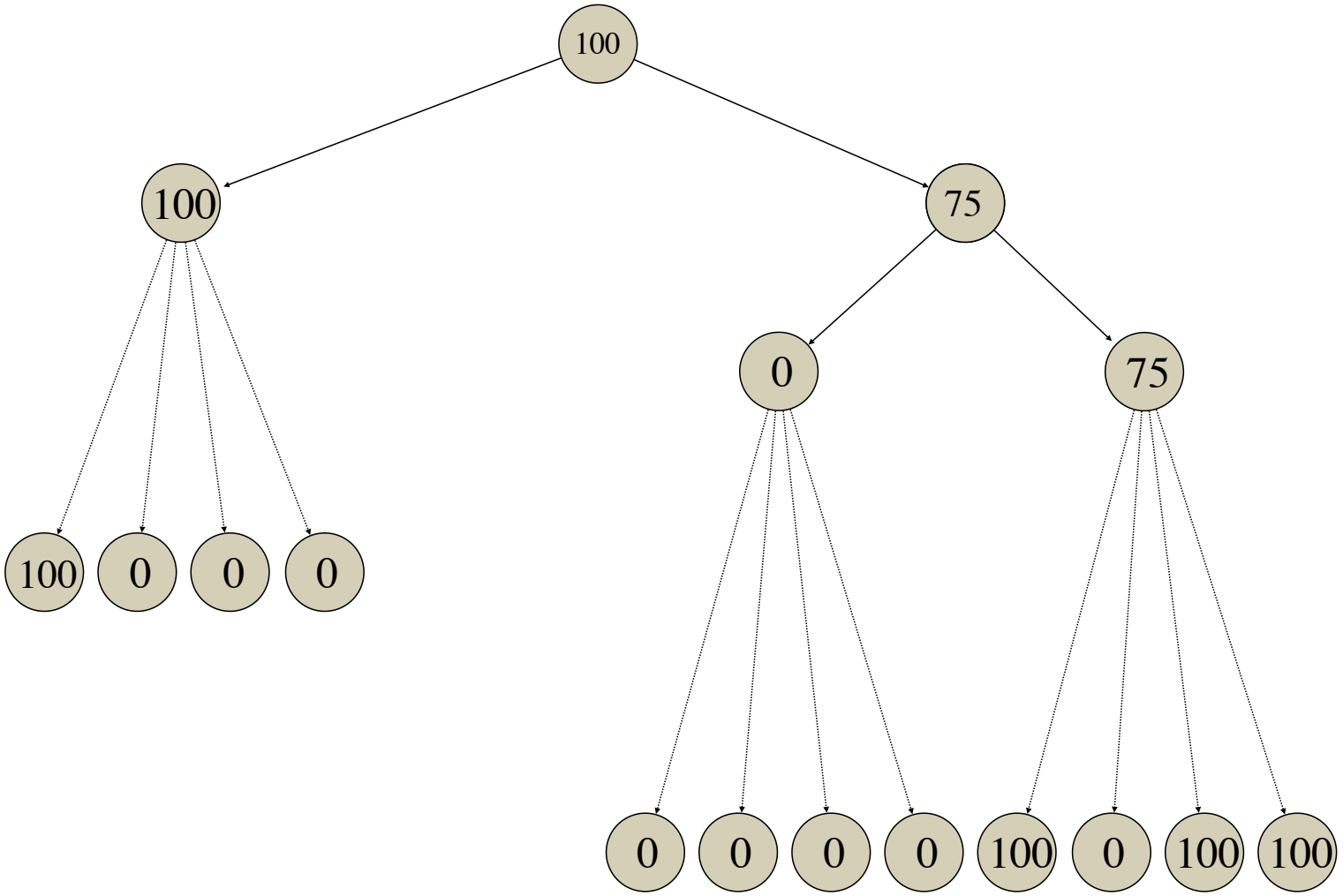
selectaction

```
function selectaction (actions,scores,visits)
 {var action = actions[0];
  var score = -1;
  var probes = 0;
  for (var i=0; i<actions.length; i++)
      {var newscore = Math.round(scores[i]/visits[i]);
       if (newscore===100) {return actions[i]};
       if (newscore>score)
          {action = actions[i];
           score = newscore;
           probes = visits[i]}};
  return action}
```

Depth-Limited or Iterative Deepening MCS



Non-Uniform Search MCS



Problems and Features

Problems

- Optimistic - opponent might not respect probabilities
- Does not utilize game structure in any useful way

Problems and Features

Benefits of depthcharging

- Fast because no branching
- Small space because nothing stored in probes
- Provides guidance when other heuristics fail

Problems

- Optimistic - opponent might not respect probabilities
- Does not utilize game structure in any useful way

Throws away depth charge info from previous steps

Monte Carlo Tree Search

Basic Idea

Monte Carlo Tree Search (MCTS) is a blend of PTS and MCS with selection based on exploration and exploitation

Like MCS:

Random probes to end of game to estimate state values

Like PTS:

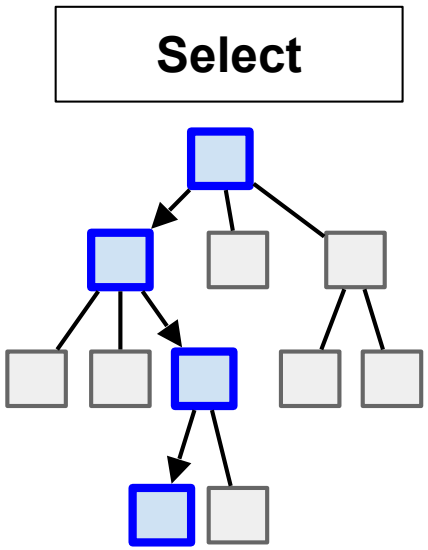
MCTS expands non-uniformly

Keeps track of depth charges across expansions

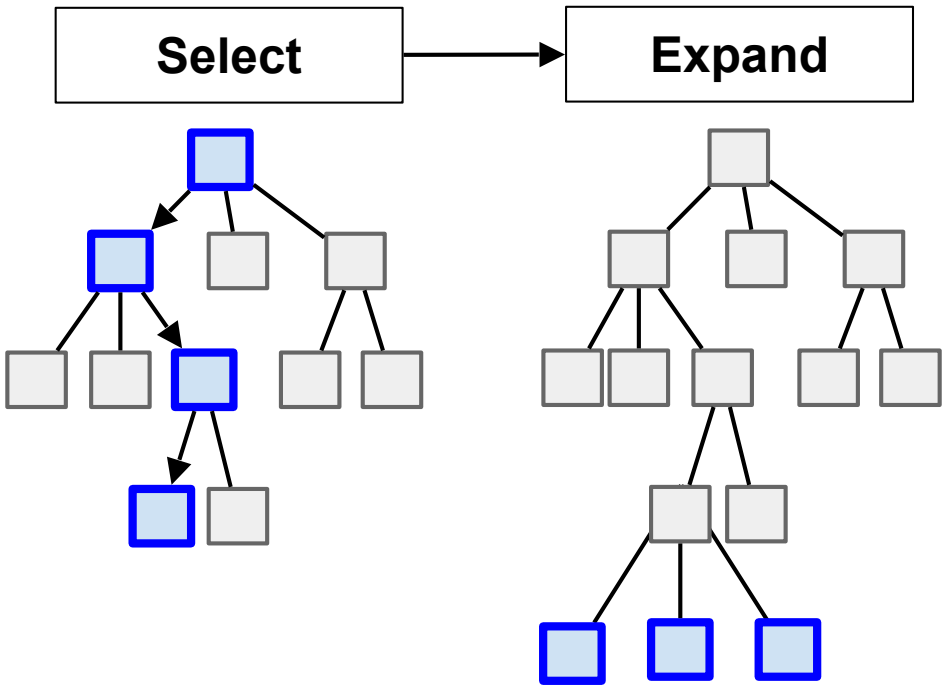
Uses combination of Exploitation and Exploration

Upper Confidence Bounds on Trees (UCT) formula

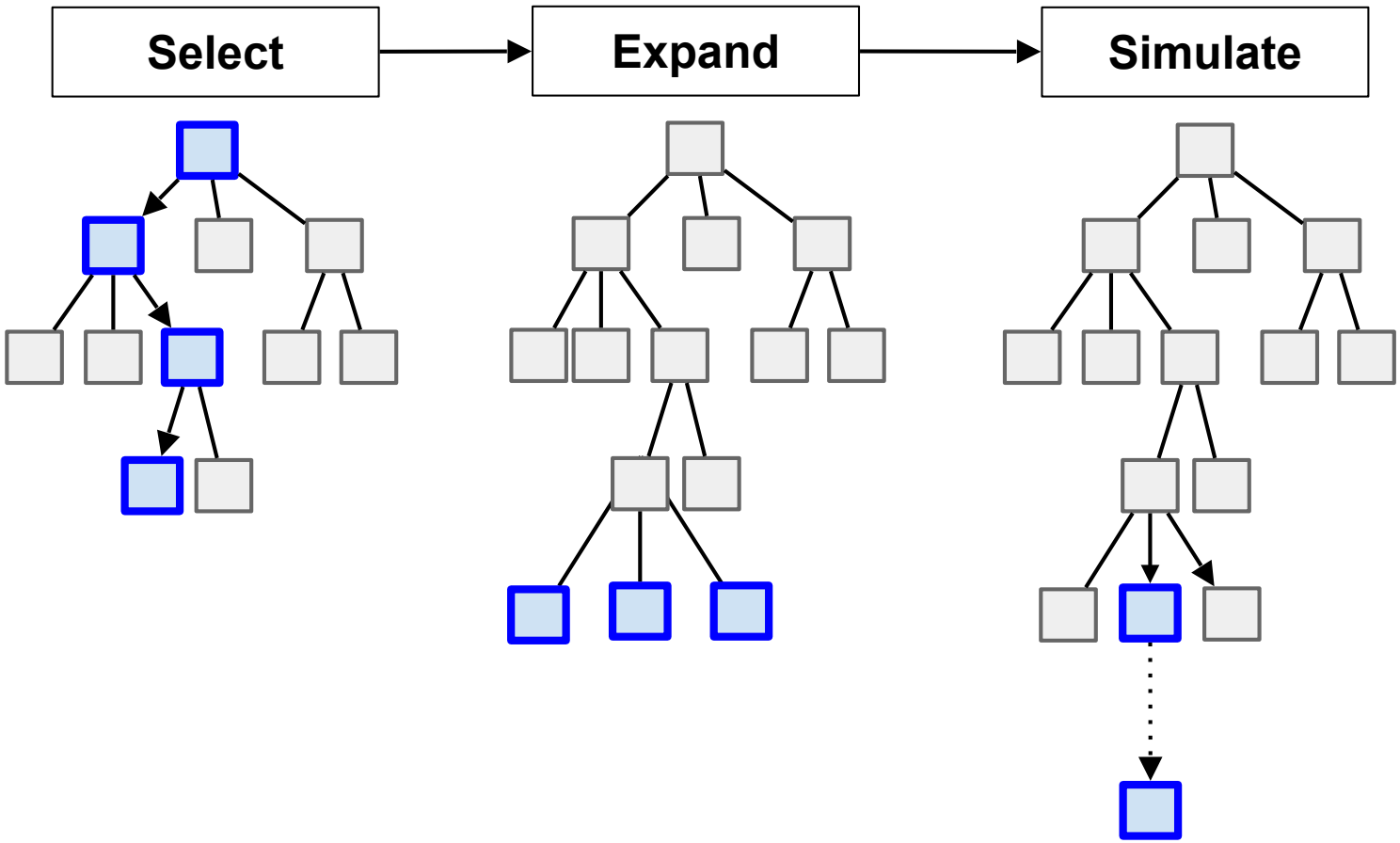
Overview



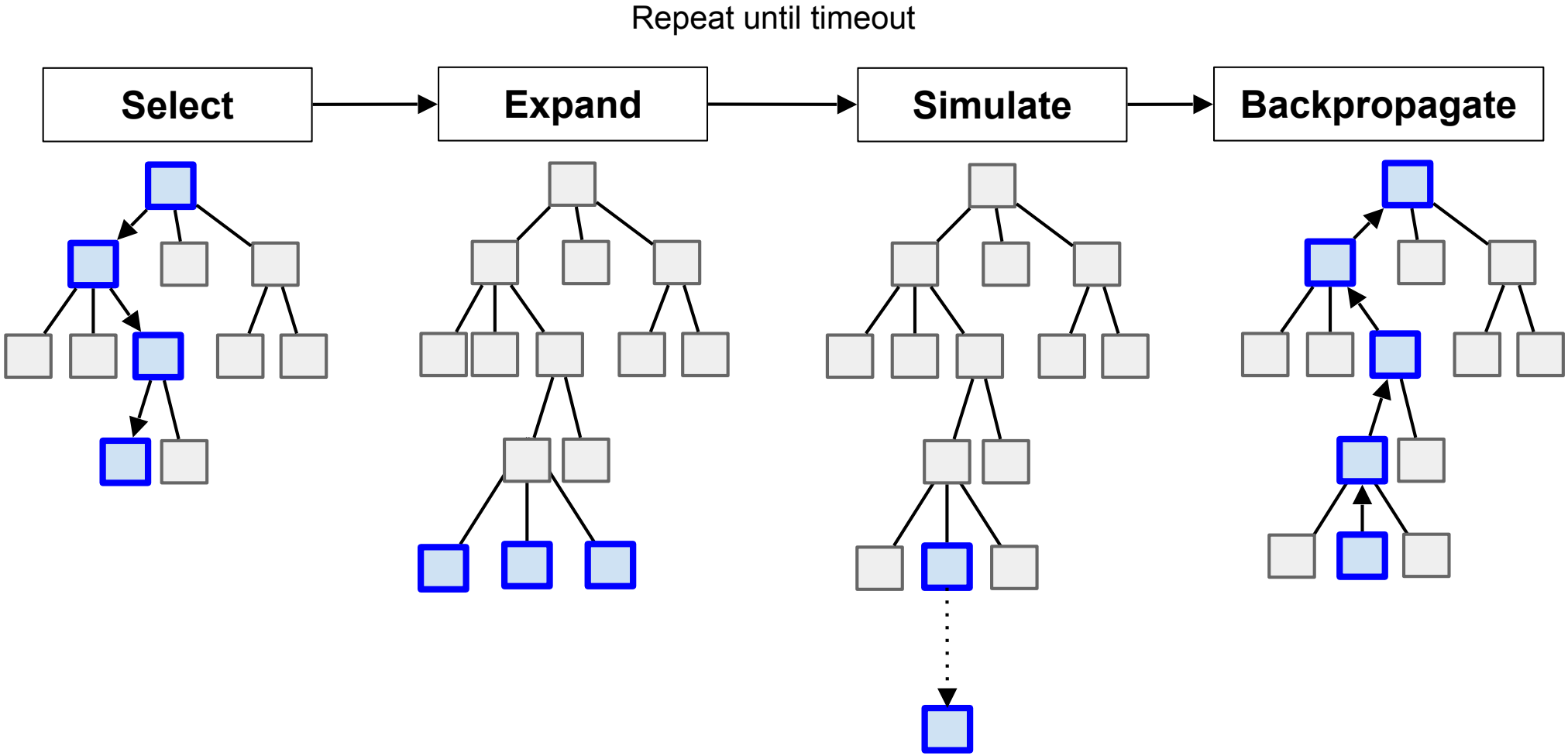
Overview



Overview



Overview


Repeat until timeout
Select
Expand
Simulate
Backpropagate

Ideas

Some players use different formulas for combining Exploitation and Exploration.

$$\text{selectValue} = \frac{\text{node.utility}}{\text{node.visits}} + C \times \sqrt{\frac{\ln(\text{node.parent.visits})}{\text{node.visits}}}$$

Some players find value in recording the **standard deviation** of scores at each node or where the **change in estimated values** and favor expanding those nodes with the highest value to clarify the uncertainty.

More Information

https://en.wikipedia.org/wiki/Monte_Carlo_tree_search

Indexing

Rule Evaluation

State Data

```
p(a)
p(b)
p(c)
q(d)
```

Game Rule

```
terminal :- p(X) & q(X)
```

Rule Evaluation *Without Indexing*

State Data

```
p(a)
p(b)
p(c)
q(d)
```

Game Rule

```
terminal :- p(X) & q(X)
```

```
p(X) & q(X)?
  p(X)=p(a)? yes if X=a
      q(a)=p(a)? no
      q(a)=p(b)? no
      q(a)=p(c)? no
      q(a)=q(d)? no
  p(X)=p(b)? yes if X=b
      q(b)=p(a)? no
      q(b)=p(b)? no
      q(b)=p(c)? no
      q(b)=q(d)? no
  p(X)=p(c)? yes if X=c
      q(c)=p(a)? no
      q(c)=p(b)? no
      q(c)=p(c)? no
      q(c)=q(d)? no
  p(X)=q(d)? no
```

Assumptions

Worst Case Analysis based on *number of unifications* in terms of the number of objects *n*

(1) We will first consider analysis with *no indexing*.
(2) Then we will look at analysis with *indexed data*.

Example

Example

```
r(a,c) :- p(a,Y) & p(Y,c)
```

Cost of computing whether `r(a,c)` is true

$$n\text{^}2 + n*n\text{^}2 = n\text{^}2 + n\text{^}3$$

Suppose $n = 3$

$$3\text{^}2 + 3*3\text{^}2 = 3\text{^}2 + 3\text{^}3 = 36$$

Example

Example

```
r(X,Z) :- p(X,Y) & p(Y,Z)
```

Cost of computing *all* instances of `r`

$$n\text{^}2 + n\text{^}2 * n\text{^}2 = n\text{^}2 + n\text{^}4$$

Suppose $n = 3$

$$3\text{^}2 + 3\text{^}2 * 3\text{^}2 = 3\text{^}2 + 3\text{^}4 = 90$$

Full Indexing

In *full indexing*, each factoid appears on the list of factoids associated with each constant in that factoid.

Example: {p(a,b), p(b,c), q(b), q(c)}

Index on p: {p(a,b), p(b,c)}
Index on q: {q(b), q(c)}

Index on a: {p(a,b)}
Index on b: {p(a,b), p(b,c), q(b)}
Index on c: {p(b,c), q(c)}

NB: No *compound* indices (e.g. all factoids with a *and* b)
NB: No *positional* indices (e.g. p(a,b) vs. p(b,a))

Worst Case

Example:

`p(a,a)`	`p(b,a)`	`p(c,a)`
`p(a,b)`	`p(b,b)`	`p(c,b)`
`p(a,c)`	`p(b,c)`	`p(c,c)`

Index on `p`: {`p(a,a), ... , p(c,c)`}

Index on `a`: {`p(a,a)`,`p(a,b)`,`p(a,c)`,`p(b,a)`,`p(c,a)`}

Index on `b`: {`p(a,b)`,`p(b,a)`,`p(b,b)`,`p(b,c)`,`p(c,b)`}

Index on `c`: {`p(a,c)`,`p(b,c)`,`p(c,a)`,`p(c,b)`,`p(c,c)`}

Example with Indexing

Example

```
r(a,c) :- p(a,Y) & p(Y,c)
```

Cost of computing `r(a,c)` *without* indexing

$$n^2 + n*n^2 = n^2 + n^3$$

Suppose $n = 3$

$$3^2 + 3*3^2 = 3^2 + 3^3 = 36$$

Cost of computing whether `r(a,c)` *with* indexing

$$(2n-1) + n*(2n-1) = 2n^2 + n - 1$$

Suppose $n = 3$

$$2*3^2 + 3 - 1 = 18 + 2 = 20$$

Example with Indexing

Example

```
r(X,Z) :- p(X,Y) & p(Y,Z)
```

Cost of computing `r(X,Z)` *without* indexing

$$n^2 + n^2 * n^2 = n^2 + n^4$$

Suppose $n = 3$

$$3^2 + 3^2 * 3^2 = 3^2 + 3^4 = 90$$

Cost of computing *all* instances *with* indexing:

$$n^2 + n^2 * (2n-1) = n^2 + 2n^3 - n^2 = 2n^3$$

Suppose $n = 3$

$$2 * 3^3 = 54$$

depthcharge

```
function depthcharge (state)
 {if (findterminalp(state,library))
     {return findreward(role,state,library)*1};
  var actions = findlegals(state,library);
  if (actions.length===0) {return 0};
  var best = randomindex(actions.length);
  var newstate = simulate(actions[best],state,library);
  return depthcharge(newstate)}
```

depthcharge with Indexed

```
function depthcharge (state)
 {if (findterminalp(state,library))
     {return findreward(role,state,library)*1};
  var actions = findlegals(state,library);
  if (actions.length===0) {return 0};
  var best = randomindex(actions.length);
  var newstate = simulate(actions[best],state,library);
  newstate = definemorefacts([],newstate);
  return depthcharge(newstate)}
```

explore

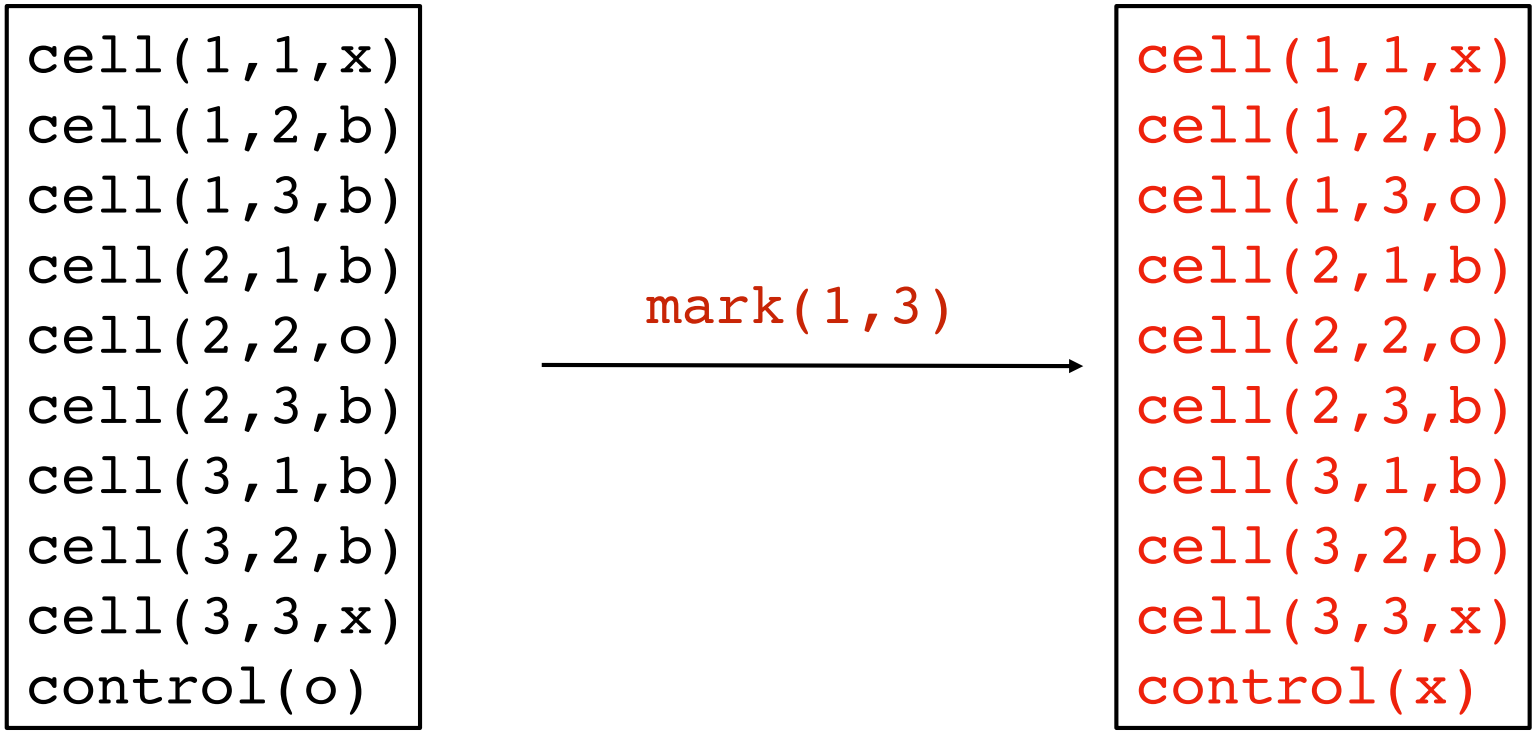
```
function explore (states,scores,visits,deadline)
 {var index = 0;
  while (Date.now()<deadline)
   {if (index>=states.length) {index = 0};
    var result = depthcharge(states[index]);
    scores[index] = scores[index] + result;
    visits[index] = visits[index] + 1;
    depthcharges++; index++};
  return true}
```

explore with Index

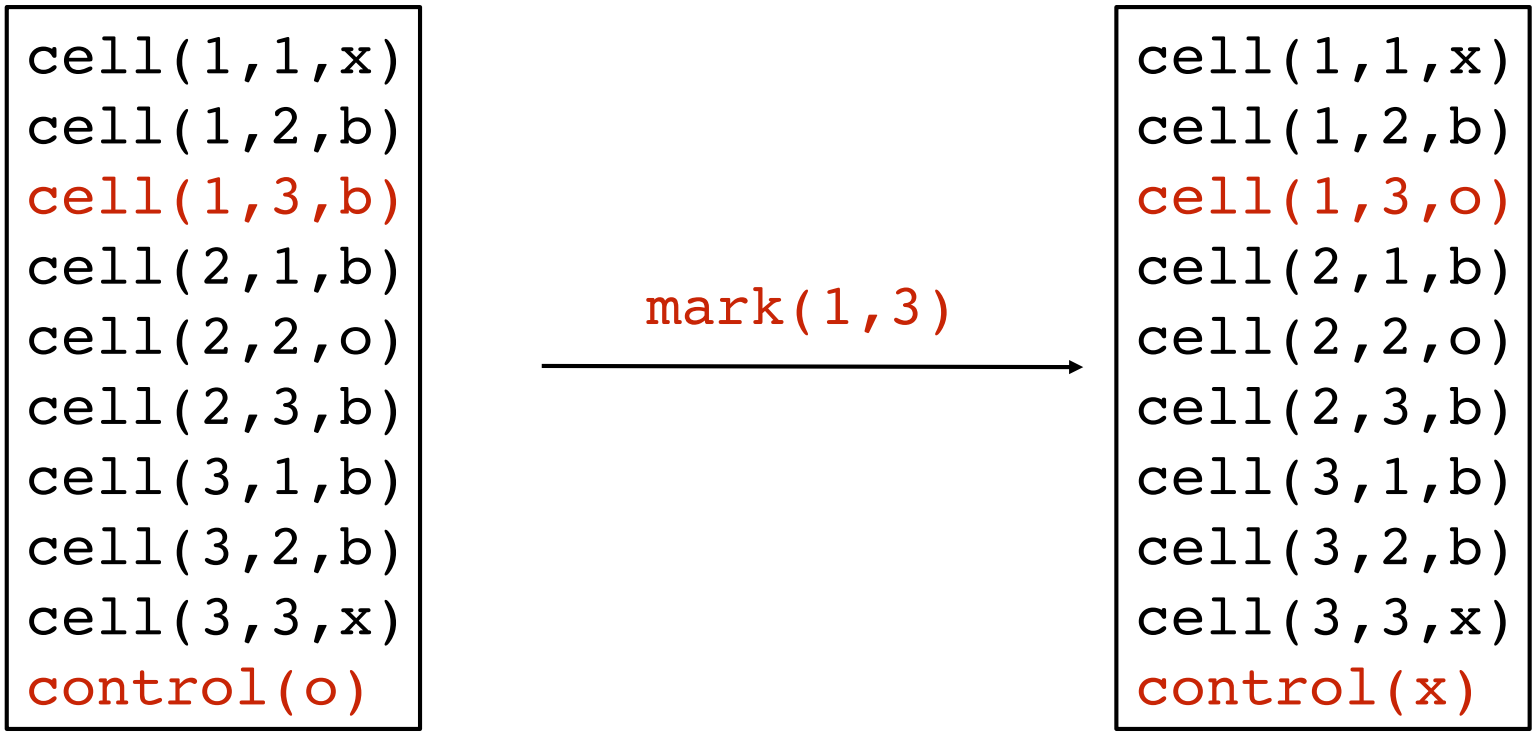
```
function explore (states,scores,visits,deadline)
 {var index = 0;
  while (Date.now()<deadline)
   {if (index>=actions.length) {index = 0};
    dataindexing = true;
    var copy = definemorefacts([],states[index]);
    var result = depthcharge(copy);
    dataindexing = false;
    scores[index] = scores[index] + result;
    visits[index] = visits[index] + 1;
    index++};
  return true}
```

Differential Update

Simulation with Copying



Differential Update



Copying versus Differential Update

Copying update *with* indexing:

```
function depthcharge (state)
 {if (findterminalp(state,library))
     {return findreward(role,state,library)*1};
  var actions = findlegals(state,library);
  if (actions.length===0) {return 0};
  var best = randomindex(actions.length);
  var newstate = simulate(actions[best],state,library);
  newstate = definemorefacts([],newstate);
  return depthcharge(newstate)}
```

Differential update *with* indexing:

```
function depthcharge (state)
 {if (findterminalp(state,library))
     {return findreward(role,state,library)*1};
  var actions = findlegals(state,library);
  if (actions.length===0) {return 0};
  var best = randomindex(actions.length);
  compexecute(actions[best],state,library);
  return depthcharge(state)}
```

Results on Alquerque

MCS with no indexing and copy update:

Probes: 67

Visits: 8

MCS with indexing and differential update:

Probes: 83

Visits: 10

Results on Leafy McFall Face

MCS with no indexing and copy update:

Probes: 1417

Visits: 709

MCS with indexing and differential update:

Probes: 2154

Visits: 1077

Results on Tic Tac Toe

MCS with no indexing and copy update:

Probes: 13547

Visits: 1506

MCS with indexing and differential update:

Probes: 21896

Visits: 2433

Results on Skirmish

MCS with no indexing and copy update:

Probes: 11

Visits: 1

MCS with indexing and differential update:

Probes: 18

Visits: 1

GENERAL
GAME
PLAYING