Markov Decision Problem
Policy Iteration Method

This Hypercard stack was prepared by:
Dennis L. Bricker,
Dept. of Industrial Engineering,
University of Iowa,
Iowa City, Iowa 52242
e-mail: dbricker@icaen.uiowa.edu

author

☞ **Policy-Iteration Algorithm without Discounting**

Optimizes the "average", i.e., expected, cost or return per period in steady state.

☞ **Policy-Iteration Algorithm with Discounting**

Optimizes the present value of all future expected costs

Policy-Iteration Algorithm without Discounting

For each policy R=(k_1,k_2, ... k_n), define

$\pi^R = (\pi_1^R, \pi_2^R, \ldots \pi_n^R)$ to be the steady state distribution using policy R

$g(R) = \sum_{i\in S} \pi_i^R C_i^{k_i}$ to be the expected cost per stage (in steady state) if policy R is used.

$v_i^n(R)$ = total expected cost during the next n stages if the system starts in state i & follows policy R

$$v_i^n(R) = C_i^{k_i} + \sum_{j\in S} p_{ij}^{k_i} v_j^{n-1}(R)$$

For "large" n,

$$v_i^n(R) \approx n\,g(R) + v_i(R)$$

where

$v_i(R)$ = effect on total expected cost (to ∞) due to the system's starting in state i

$$\begin{cases} v_i^n(R) = C_i^{k_i} + \sum_{j\in S} p_{ij}^{k_i} v_j^{n-1}(R) \\ v_i^n(R) \approx n\,g(R) + v_i(R) \end{cases}$$

$$\Rightarrow \quad n\,g(R) + v_i(R) = C_i^{k_i} + \sum_{j\in S} p_{ij}^{k_i}\left[(n-1)\,g(R) + v_j(R)\right]$$

$$= C_i^{k_i} + \sum_{j\in S} p_{ij}^{k_i} v_j(R) + (n-1)\,g(R) \sum_{j\in S} p_{ij}^{k_i}$$

$$\Rightarrow \quad \boxed{g(R) + v_i(R) = C_i^{k_i} + \sum_{j\in S} p_{ij}^{k_i} v_j(R) \quad \forall\, i\in S}$$

$$\boxed{g(R) + v_i(R) = C_i^{k_i} + \sum_{j\in S} p_{ij}^{k_i} v_j(R) \quad \forall\, i\in S}$$

Given a policy R, this will be a system of n linear equations with n+1 unknowns, i.e.,

$$g(R), v_1(R), v_2(R), \ldots v_n(R)$$

To find a solution, therefore, we may assign an arbitrary value (usually zero) to one of the unknowns $v_i(R)$, say $v_n(R)$

Policy-Iteration Algorithm

Step 0: **Initialization**

Start with any policy R.

Step 1: **Value Determination**

Solve the system of linear equations

$$g(R) + v_i(R) = C_i^{k_i} + \sum_{j\in S} p_{ij}^{k_i} v_j(R) \quad \forall\, i\in S$$

for $g(R), v_1(R), v_2(R), \ldots v_{n-1}(R)$,

letting $v_n(R)=0$

Policy-Iteration Algorithm

Step 2: **Policy Improvement**

Find an improved policy R' such that

$$R' = (k'_1, k'_2, \ldots k'_n)$$

and

$$C_i^{k'_i} + \sum_{j} p_{ij}^{k'_i} v_j(R) \le g(R) + v_i(R) \quad \forall\, i\in S$$

with strict inequality for at least one state i. If no such improved policy exists, stop; otherwise, return to step 1.

$$\underbrace{C_i^{k'_i} + \sum_j p_{ij}^{k'_i} v_j(R)}_{\text{expected cost if at stage 1 we take action } k'_i\text{, and then follow policy } R.} \le \underbrace{g(R) + v_i(R)}_{\text{expected cost if, beginning at stage 1, we follow policy } R} \quad \forall\ i\in S$$

Taxicab Problem

State	Action
1 Town A	1 Cruise
2 Town B	1 Cruise
3 Town C	1 Cruise

g(R) = ¯9.2

i	Vi
1	¯1.33333
2	¯7.46667
3	0

Taxicab Problem

Policy Improvement Step: Evaluation of alternate actions

State #1, Town A

Current Policy: action #1, Cruise
g(R)+Vi(R) = ¯10.5333

k	name	C'	ΔC
1	Cruise	¯10.5333	0
2	Cabstand	¯8.43333	2.1
3	Wait for call	¯5.51667	5.01667

C'[k] = cost if action k is selected for one stage
ΔC[k] = improvement (if <0)

Taxicab Problem

Policy Improvement Step: Evaluation of alternate actions

State #2, Town B

Current Policy: action #1, Cruise
g(R)+Vi(R) = ¯16.6667

k	name	C'	ΔC
1	Cruise	¯16.6667	0
2	Cabstand	¯21.6167	¯4.95

C'[k] = cost if action k is selected for one stage
ΔC[k] = improvement (if <0)

Taxicab Problem

Policy Improvement Step: Evaluation of alternate actions

State #3, Town C

Current Policy: action #1, Cruise
g(R)+Vi(R) = ¯9.2

k	name	C'	ΔC
1	Cruise	¯9.2	0
2	Cabstand	¯9.76667	¯0.566667
3	Wait for call	¯5.96667	3.23333

C'[k] = cost if action k is selected for one stage
ΔC[k] = improvement (if <0)

Taxicab Problem

State	Action
1 Town A	1 Cruise
2 Town B	2 Cabstand
3 Town C	2 Cabstand

g(R) = ¯13.1515

i	Vi
1	3.87879
2	¯12.8485
3	0

Taxicab Problem

Policy Improvement Step: Evaluation of alternate actions

State #1, Town A

Current Policy: action #1, Cruise
g(R)+Vi(R) = ¯9.27273

k	name	C'	ΔC
1	Cruise	¯9.27273	0
2	Cabstand	¯12.1439	¯2.87121
3	Wait for call	¯4.88636	4.38636

C'[k] = cost if action k is selected for one stage
ΔC[k] = improvement (if <0)

Taxicab Problem

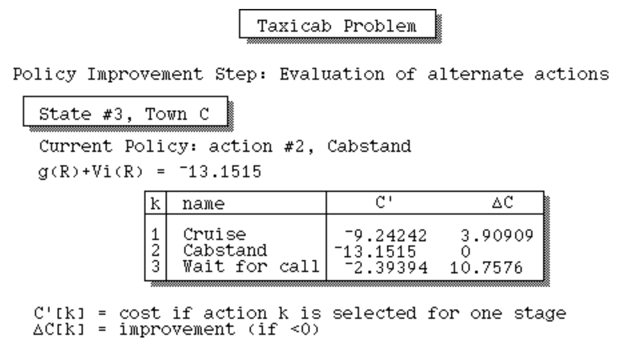
Policy Improvement Step: Evaluation of alternate actions

State #2, Town B

Current Policy: action #2, Cabstand
g(R)+Vi(R) = ¯26

k	name	C'	ΔC
1	Cruise	¯14.0606	11.9394
2	Cabstand	¯26	0

C'[k] = cost if action k is selected for one stage
ΔC[k] = improvement (if <0)



Taxicab Problem

Policy Improvement Step: Evaluation of alternate actions

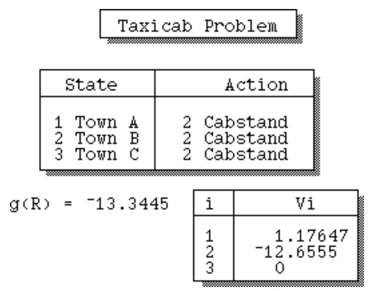
State #3, Town C

Current Policy: action #2, Cabstand
g(R)+Vi(R) = ¯13.1515

k	name	C'	ΔC
1	Cruise	¯9.24242	3.90909
2	Cabstand	¯13.1515	0
3	Wait for call	¯2.39394	10.7576

C'[k] = cost if action k is selected for one stage
ΔC[k] = improvement (if <0)

©Dennis Bricker, U. of Iowa, 1998



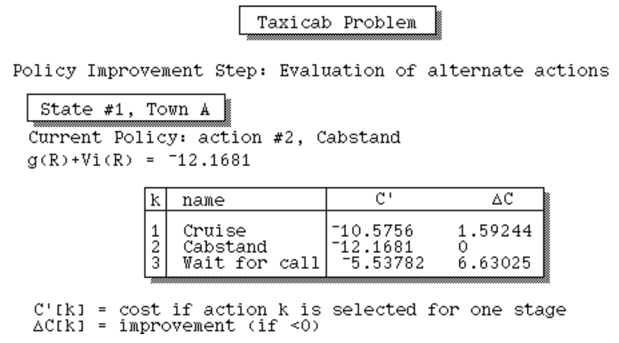
Taxicab Problem

State	Action
1 Town A	2 Cabstand
2 Town B	2 Cabstand
3 Town C	2 Cabstand

g(R) = ¯13.3445

i	Vi
1	1.17647
2	¯12.6555
3	0

©Dennis Bricker, U. of Iowa, 1998



Taxicab Problem

Policy Improvement Step: Evaluation of alternate actions

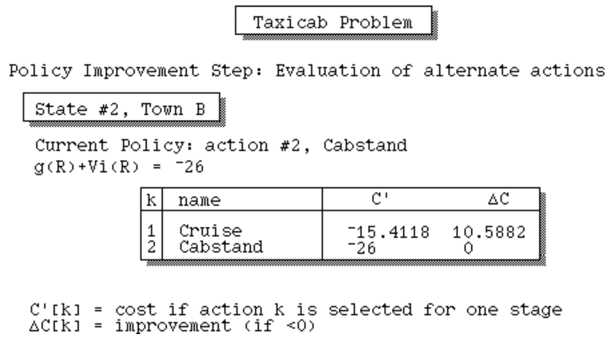
State #1, Town A

Current Policy: action #2, Cabstand
g(R)+Vi(R) = ¯12.1681

k	name	C'	ΔC
1	Cruise	¯10.5756	1.59244
2	Cabstand	¯12.1681	0
3	Wait for call	¯5.53782	6.63025

C'[k] = cost if action k is selected for one stage
ΔC[k] = improvement (if <0)

©Dennis Bricker, U. of Iowa, 1998



Taxicab Problem

Policy Improvement Step: Evaluation of alternate actions

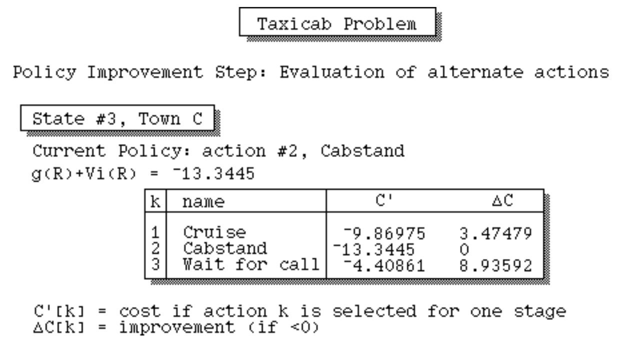
State #2, Town B

Current Policy: action #2, Cabstand
g(R)+Vi(R) = ¯26

k	name	C'	ΔC
1	Cruise	¯15.4118	10.5882
2	Cabstand	¯26	0

C'[k] = cost if action k is selected for one stage
ΔC[k] = improvement (if <0)

©Dennis Bricker, U. of Iowa, 1998



Taxicab Problem

Policy Improvement Step: Evaluation of alternate actions

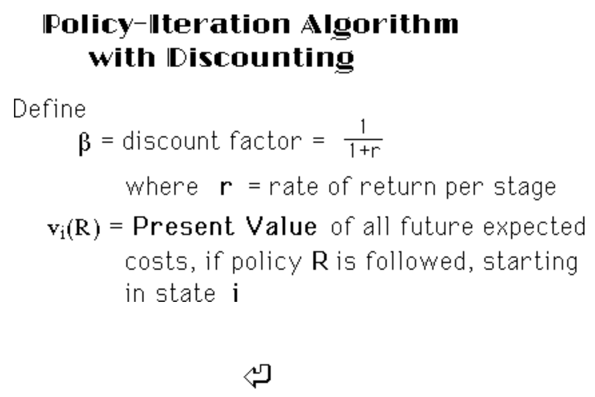
State #3, Town C

Current Policy: action #2, Cabstand
g(R)+Vi(R) = ¯13.3445

k	name	C'	ΔC
1	Cruise	¯9.86975	3.47479
2	Cabstand	¯13.3445	0
3	Wait for call	¯4.40861	8.93592

C'[k] = cost if action k is selected for one stage
ΔC[k] = improvement (if <0)

©Dennis Bricker, U. of Iowa, 1998



Policy-Iteration Algorithm with Discounting

Define

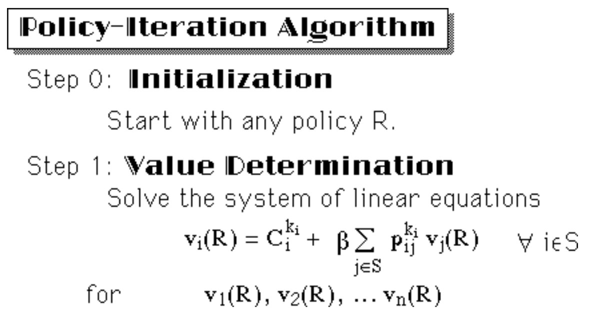
β = discount factor = $\frac{1}{1+r}$

where **r** = rate of return per stage

$v_i(R)$ = **Present Value** of all future expected costs, if policy **R** is followed, starting in state **i**

⏎

©Dennis Bricker, U. of Iowa, 1998



Policy-Iteration Algorithm

Step 0: **Initialization**

Start with any policy R.

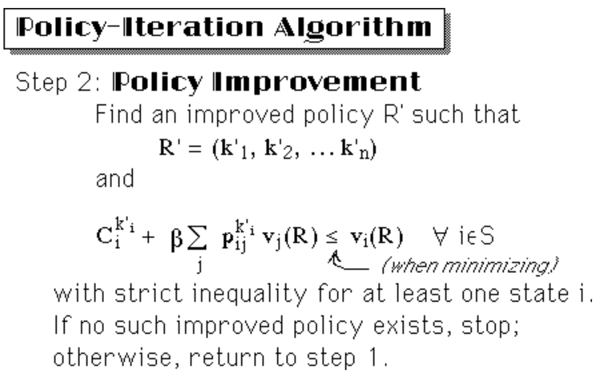
Step 1: **Value Determination**

Solve the system of linear equations

$$v_i(R) = C_i^{k_i} + \beta \sum_{j \in S} p_{ij}^{k_i} v_j(R) \qquad \forall\, i \in S$$

for $v_1(R), v_2(R), \ldots v_n(R)$

©Dennis Bricker, U. of Iowa, 1998



Policy-Iteration Algorithm

Step 2: **Policy Improvement**

Find an improved policy R' such that

$$R' = (k'_1, k'_2, \ldots k'_n)$$

and

$$C_i^{k'_i} + \beta \sum_j p_{ij}^{k'_i} v_j(R) \underset{\nwarrow \text{(when minimizing)}}{\le} v_i(R) \qquad \forall\, i \in S$$

with strict inequality for at least one state i. If no such improved policy exists, stop; otherwise, return to step 1.

©Dennis Bricker, U. of Iowa, 1998