```
function [pol_pi,policyStable] =
jcr_policy_improvement(pol_pi,V,gamma,Ra,Pa,Rb,Pb,max_num_cars_can_transfer)
% JCR_POLICY_EVALUATION - Performs policy evaluation evaluations returning the
%  state-value function for the jacks car rental example.
%
% The Basic Algorithm is: Iterate the Bellman equation:
%
% V(s) <- \sum_a \pi(s,a) \sum_{s'} P_{s,s'}^a (R_{s,s'}^a + \gamma V(s'))
%
% where the policy is given as input.  These iterations are done IN PLACE.
%
% See ePage 262 in the Sutton book.
%
% Input:
%   V = An array to hold the values of the state-value function
%
% Written by:
% --
% John L. Weatherwax                2007-12-03
%
% email: wax@alum.mit.edu
%
% Please send comments and especially bug reports to the
% above email address.
%
%-----

if( nargin < 3 ) gamma = 0.9; end

% the maximum number of cars at each site (assume equal):
max_n_cars = size(V,1)-1;

% the total number of states (including the states (0,Y) and (X,0)):
nStates = (max_n_cars+1)^2;

% assume the policy is stable (until we learn otherwise below):
policyStable = 1; tm = NaN;

% For each state in \cal{S}:
fprintf('beginning policy improvement...\n');
for si=1:nStates,
    % get the number of cars (ones based) at each site (at the END of the day):
    [na1,nb1] = ind2sub( [ max_n_cars+1, max_n_cars+1 ], si );
    na = na1-1; nb = nb1-1; % (zeros based)
    %fprintf( 'prev state took = %10.5f (min); considering state =
%5d=(na=%5d,nb=%5d)...\n', tm/60, si,na,nb );

    % our policy says in this state do the following:
    b = pol_pi(na1,nb1);

    % are there any BETTER actions for this state?
    % --consider all possible actions in this state:
    % we can transfer up to na cars from site A to B
    % we can transfer up to nb cars from site B to site A (introducing a negative sign)
    %
    % It seemed there were various ways to interpret this problem:
    %
    % 1) assume the number of cars we can transfer is restricted only by the number we
have
```

```
    %
    posA = min([na,max_num_cars_can_transfer]);
    posB = min([nb,max_num_cars_can_transfer]);

    posActionsInState = [ -posB:posA ]; npa = length(posActionsInState);
    Q = -Inf*ones(1,npa);
    tic;
    for ti = 1:npa,
      ntrans = posActionsInState(ti);

      %---
      % based on the state and action compute the expectation over
      %     all possible states we may transition to i.e. s'
      % We need to consider 1) the number of possible returns at site A/B
      %                     2) the number of possible rentals at site A/B
      %---
      Q(ti) =
jcr_rhs_state_value_bellman(na,nb,ntrans,V,gamma,Ra,Pa,Rb,Pb,max_num_cars_can_transfer);
    end % end ntrans
    tm=toc;

    % check if this policy gives the best action
    [dum,imax] = max( Q );
    maxPosAct  = posActionsInState(imax);
    if( maxPosAct ~= b )      % this policy in fact does NOT give the best action ...
      policyStable = 0;
      pol_pi(na1,nb1) = maxPosAct; % <- so we update our policy
    end
end % end state loop
fprintf('ended policy improvement...\n');
```