Online-Appendix zu

„Stochastic Optimization of Bioreactor Control Policies Using a Markov Decision Process Model"

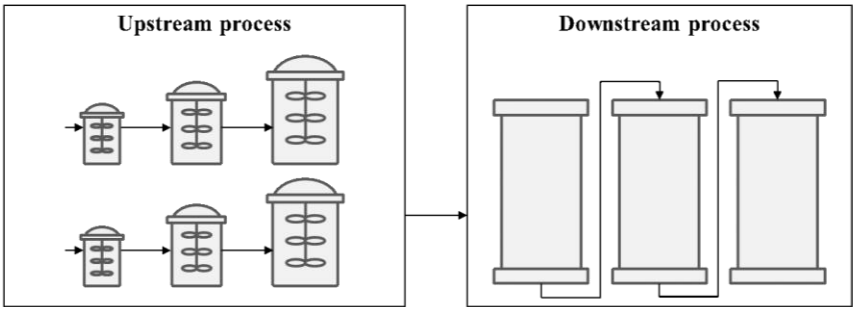
Quirin Stockinger

Technische Universität München

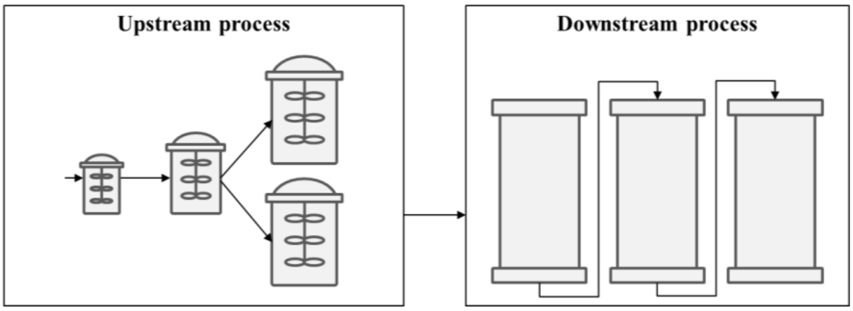
Junior Management Science 5(1) (2020) 50-80

Appendices

Appendix 1: Schematic view of a biopharmaceutical production process with two parallel seed-trains and one purification process



Appendix 2: Schematic view of a biopharmaceutical production process with a seed-train inoculating two parallel production reactors which are harvested into one purification process



Appendix 3: Comparison of problem characteristics between this contribution and selected existing literature

Reference[8]	System-level policies	USP & DSP decisions	Production uncertainty	Resin performance decay	DSP uncertainties	Financial trade-off
Schmidt (1996)	✓		✓			✓
Liu et al. (2014)		✓	✓	✓[9]		✓
Liu et al. (2016)		✓	✓		✓	✓
Martagan et al. (2016)	✓		✓			✓
Martagan et al. (2018)	✓		✓		✓	✓
Martagan et al. (Accepted/In press)	✓	(✓)[10]			✓	✓
This work	✓	✓	✓	✓	✓	✓

[8] Non-exhaustive list of existing literature

[9] Only deterministic decay of resin binding capacity

[10] Greatly limited upstream decision space to how much protein to produce

Appendix 4: Formal problem statement

Given:

- A predetermined biopharmaceutical protein to produce
- A predetermined setup of up- and downstream processes with which the protein can be produced, including the number and volume of parallel production reactors and the chromatography resin used in the first step
- A set of decision epochs in which the system is observed and influenced
- A set of states, approximating the fermentation of the specific protein in each production reactor
- A set of states, representing the remaining performance of a predetermined chromatography resin per the number of purified batches
- A minimum allowed capacity with which the resin can still be used to purify
- Feasible actions related to the control of the production reactors and chromatography step per up- and downstream state
- Upstream transition probabilities, including the per-cycle probability of process failure due to fermentation upsets
- Downstream transition probabilities, determining the distribution of performance decay
- A function determining the accumulation of the protein during production
- Upstream operating costs (e.g., cost of materials such as growth and production medium and labor) and revenues (e.g., salvage value of spent growth medium, value of the produced protein)
- Material costs of chromatography resin

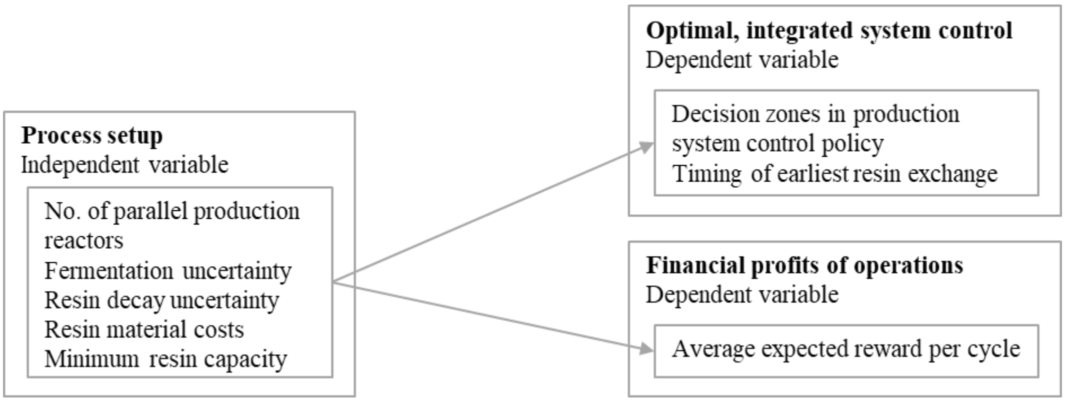
Determine:

- The policy for the integrated control of the two sub-problems
  - *upstream fermentation:* preparing the reactors, starting and continuing culture growth, starting and continuing protein production, harvesting, and restarting a batch
  - *downstream resin exchange:* accepting harvested media for purification and exchanging spent resin
- The associated maximum financial value

Objective:

- Maximize average per-cycle operating profits

Appendix 5: Non-directional hypotheses regarding the presented research questions



Appendix 6: Summary of MDP parameters and their respective values in the numerical case study

Parameter	Description	Value
$MaxBatch$	Number of purified batches before resin must be exchanged	11
$cap_{MaxBatch}$	Minimum allowed resin capacity	65 %
$p\left(cap_m, cap_{m+\{0,1,2\}}, 2\right)$,	Probability of no performance decay, performance decay by one or by two steps	5 %, 90 %, 5 %
n_g, n_p	Number of growth and production states, respectively	8, 36
$p(s_t^U, s_{t+1}^U, a_{t+1}^U)$	Non-zero, non-one transition probabilities during USP	99.3 %
$fixed(a^k)$	Fixed costs of taking out an action in either USP or DSP component k	100 $
c_g, c_p	Cost of growth and production medium, respectively	8 $/L, 12.8 $/L
c_b	Cost of microcarriers	0 $/L
v_g	Value of growth medium, based on a 1.5 mg/L TPA concentration	36 $/L
v_p	Value of TPA	24,000 $/g
V	Production reactor volume	160 L
x_{max}	Maximum concentration of TPA in the production medium	33.5 mg/L
c_{resin}	Material cost of exchanging resin	96,400 $

Appendix 7: Model and hypotheses-test code for the 1:1 TPA case study

```
%==========================================================================
%
% Run the MDP case study with the parameters set below.
%
%==========================================================================


% Set process parameters
% Syntax: [reactor volume,
%          cost of microcarriers ($/L),
%          cost of growth medium ($/L),
%          cost of production medium ($/L),
%          value of TPA ($/g),
%          fixed costs of operations ($)
%          cost of resin,
%          probability of successful upstream transition,
%          probability of no resin decay,
%          probability of resin decay by one step,
%          probability of resin decay by two steps,
%          minimum capacity of resin (\in [0..1])
%          maximum concentration of tPa in production medium (mg/L)], cf Datar, 1993: tPa production in CHO cells
processParameters = [160, 0, 12.8, 2, 24000, 100, 96480, 0.993, 0.05, 0.9, 0.05, 0.65, 33.5];


% set MDP parameters
discountFactor = 0.99;

% must be \in {"average","policyIteration","valueIteration","LP"}
solutionMethod = "average";


tic;
[V, policy, averageReward] = MarkovDecisionProcess(discountFactor,processParameters,solutionMethod);
time=toc;


runTests=false;

if runTests==true
    numberOfRuns = 50;

    % Hypo a: cheaper resin -> earlier first exchange
    % Hypo b: cheaper resin -> higher average reward
    firstExchange = zeros(numberOfRuns);
```

```
    firstExchangeDuringProduction = zeros(length(firstExchange));
    timingCostOfResin = zeros(length(firstExchange));
    for i=1:length(firstExchange)
        costOfResin(i) = 192960-192960/(length(firstExchange)-1)*(i-1);
    end
    policiesCostOfResin = [];
    averageRewardsCostOfResin = [];
    for i=1:length(firstExchange)
        processParameters(7) = costOfResin(i);
        tic;
        [policy, averageReward] = MarkovDecisionProcess(discountFactor,processParameters,solutionMethod);
        timingCostOfResin(i) = toc;
        firstExchange(i) = find(~cellfun(@isempty,regexp(policy,'\w+3')),1);

        averageRewardsCostOfResin = [averageRewardsCostOfResin; averageReward];
        policiesCostOfResin = [policiesCostOfResin; policy];
    end

    x=costOfResin.';

    %Reset processParameter(7) if both test run back to back
    processParameters(7) = 96480;


    % Hypo c: steeper decay and lower minimum capacity -> earlier first exchange
    % Hypo d: steeper decay and lower minimum capacity -> higher average
    % reward
    firstExchangeCap = zeros(numberOfRuns);
    firstExchangeDuringProductionCap = zeros(length(firstExchangeCap));
    timingMinCap = zeros(length(firstExchangeCap));

    for i=1:length(firstExchangeCap)
        minimumCapacitiesCap(i) = 1.0-1.0/(length(firstExchangeCap)-1)*(i-1);
    end

    averageRewardsMinCap = [];
    policiesMinimumCapacity = [];
    for i=1:length(firstExchangeCap)
        processParameters(12) = minimumCapacitiesCap(i);
        tic;
        [policy, averageReward] = MarkovDecisionProcess(discountFactor,processParameters,solutionMethod);
        timingCostOfResin(i) = toc;
```

```
        firstExchangeCap(i) = find(~cellfun(@isempty,regexp(policy,'\w+3')),1);

        averageRewardsMinCap = [averageRewardsMinCap; averageReward];
        policiesMinimumCapacity = [policiesMinimumCapacity; policy];
    end
end
```

```
function [V, policyMatrixReadable, averageReward] = MarkovDecisionProcess(lambda,processParameters,solutionMethod)
clc
disp("One USP reactor into one chromatography reactor")
%Process parameters
V=processParameters(1);
cb=processParameters(2);
cg=processParameters(3);
cp=processParameters(4);
vp=processParameters(5);
% value of growth medium per L: concentration of 1.5mg of TPA in spent
% growth medium times the per gram value of un-purified TPA
vg=vp*1.5e-3;
fix=processParameters(6);
Cresin=processParameters(7);
p = processParameters(8);
pNoDecay = processParameters(9);
pDecayByOne = processParameters(10);
pDecayByTwo = processParameters(11);
minimumCapacity = processParameters(12);
%sufficiently large number to disincentivize infeasible ac-tions
M=99999999;
%Upstream reactor states
%let "empty"                        be state 1
%let "ready"                        be state 2
%let "growth i"                     be state 3..3+ng
%let "production j"                 be state 3+ng+1..3+ng+1+np
%let "upset"                        be state 3+ng+1+np+1
%number of growth and production states, respectively
ng = 8;
np = 36;
e=1; r=2;
growth = r+1:r+1+ng-1;
production = growth(end)+1:growth(end)+1+np-1;
Gp = growth(6):growth(8);
```

```
u = production(end)+1;
Susp=u;
productionProbabilities = zeros(length(production),0);
declinePhase=[0.84,0.67,0.5,0.34,0.17,0.05];
productionProbabilities(1:length(production)-length(declinePhase)-1)=p;
productionProbabilities(length(production)-length(declinePhase):length(production)-1)=declinePhase(1:length(declinePhase));
productionProbabilities(length(production))=1;
concentration = setConcentrations(np,processParameters(end));
%Downstream reactor states
%let "resin capacity in cycle i"            be state 1..ncap
%must be > 3
ncap = 12;
Sdsp=ncap;
%USP reactor actions
%let "none"                                 be action 1
%let "add growth medium"                    be action 2
%let "add production medium"                be action 3
%let "harvest"                              be action 4
%let "prepare"                              be action 5
%let "harvest and prepare"                  be action 6
actionsUsp=["none","addgm","addpm","harvest","prep","hprep"];
Ausp=length(actionsUsp);
%DSP reactor actions
%let "none"                                 be action 1
%let "accept"                               be action 2
%let "exchange resin"                       be action 3
actionsDsp=["none","accept","exresin"];
Adsp=length(actionsDsp);
%P(current, next, action)
Pusp=zeros(Susp,Susp,Ausp);
%action none = 1
Pusp(e,e,1)=1;
Pusp(r,r,1)=p;
Pusp(r,u,1)=1-p;
%doing nothing during growth -> upset
for i=growth(1):growth(end)
    Pusp(i,u,1)=1;
end
%doing nothing during production -> upset
for i=production(1):production(end)
    Pusp(i,u,1)=1;
end
```

```
Pusp(u,u,1)=1;
%action add growth medium = 2
for i=1:Susp
    % addgm during growth but before last growth state or in ready to start
    % growth
    if ismember(i,[r,growth(1):growth(end)-1])
        Pusp(i,i+1,2)=p;
        Pusp(i,u,2)=1-p;
    else
        %infeasible everywhere else
        Pusp(i,u,2)=1;
    end
end
%action add production medium = 3
for i=1:Susp
    % addpm to continue production until second to last pro-duction state
    if ismember(i,production(1):production(end)-1)
        %fetch probabilities from probability vector including
        %deterioration towards the end
        ProdStage2=i-3-ng+1;
        probability = productionProbabilities(ProdStage2);

        %row sums must equal 1, state transitions either with-out problem or
        %transitions to upset
        Pusp(i,i+1,3)=probability;
        Pusp(i,u,3)=1-probability;
    % addpm to convert production-competent growth state into production state
    elseif ismember(i,Gp)
        Pusp(i,production(1),3)=p;
        Pusp(i,u,3)=1-p;
    else
        %infeasible everywhere else
        Pusp(i,u,3)=1;
    end
end
%action harvest/ dump = 4
Pusp(r,e,4)=1;
% dump all growth medium to abort process
for i=growth(1):growth(end)
    Pusp(i,e,4)=1;
end
% harvest production medium is only feasible if a(DSP)=accept and a(USPj)!=4 or 6
for i=production(1):production(end)
    Pusp(i,e,4)=1;
end
% dump upset reactor's contents to abort process
Pusp(u,e,4)=1;
Pusp(e,e,4)=1;
%action prepare = 5
for i=1:Susp
```

```
    %infeasible everywhere except if reactor is empty
    if i==e
        Pusp(e,r,5)=p;
        Pusp(e,u,5)=1-p;
    else
        Pusp(i,u,5)=1;
    end
end
%action harvest/dump and prepare = 6
Pusp(r,r,6)=Pusp(e,r,5);
Pusp(r,u,6)=Pusp(e,u,5);
% dump all growth medium and prepare reactor to re-start pro-cess
for i=growth(1):growth(end)
    Pusp(i,r,6)=Pusp(e,r,5);
    Pusp(i,u,6)=Pusp(e,u,5);
end
% harvest production medium is only feasible if a(DSP) = ac-cept and prepare
% reactor to restart process
for i=production(1):production(end)
    %only feasible for USPi if a(DSP)=accept and a(USPj)!=4 or 6
    Pusp(i,r,6)=Pusp(e,r,5);
    Pusp(i,u,6)=Pusp(e,u,5);
end
Pusp(u,r,6)=Pusp(e,r,5);
Pusp(u,u,6)=Pusp(e,u,5);
Pusp(e,r,6)=Pusp(e,r,5);
Pusp(e,u,6)=Pusp(e,u,5);
%P(current, next, action)
Pdsp=zeros(Sdsp,Sdsp,Adsp);
%action none = 1
for i = 1:ncap
    %feasible only if i<ncap, must exchange if ncap is reached
    Pdsp(i,i,1) = 1;
end
%action accept = 2
%resin is used once for purification and its capac-ity deterio-rates
%stochastically
for i = 1:(ncap-1)
    %feasible only before resin has to be exchanged and if a(USP)=harvest OR harvest+prep
    if i==ncap-1
        %simplifying assumption as discussed in the-sis
        Pdsp(i,i,2) = pNoDecay;
        Pdsp(i,i+1,2) = pDecayByOne+pDecayByTwo;
    else
        Pdsp(i,i,2) = pNoDecay;
        Pdsp(i,i+1,2) = pDecayByOne;
        Pdsp(i,i+2,2) = pDecayByTwo;
    end
end
%infeasible
```

```
Pdsp(ncap,ncap,2) = 1;
%action exchange resin = 3
%resin is exchanged for a full capacity one
for i = 1:ncap
    Pdsp(i,1,3) = 1;
end
Pall = constructSystemProbabilities(Pusp,Pdsp);
%Test for  non-one row sums in transition probabilities
% rowSumProductUSP = 1;
% for i=1:Susp
%     rowSumProductUSP = rowSumProductUSP.*sum(Pusp(i,:,:));
% end
%
% rowSumProductDSP = 1;
% for i=1:Sdsp
%     rowSumProductDSP = rowSumProductDSP.*sum(Pdsp(i,:,:));
% end
%
% rowSumProductAll = 1;
% for i=1:Susp*Sdsp
%     rowSumProductAll = rowSumProductAll.*sum(Pall(i,:,:));
% end
%Upstream Process Reactor
%R(current, action)
%Maximize rewards (costs are negative, rewards positive)
Rusp=zeros(Susp,Ausp);
Rdsp=zeros(Sdsp,Adsp);
%action none = 1
for i=1:Susp
    if ismember(i,growth)
        %opportunity cost of lost growth medium because system moves to upset state
        Rusp(i,1)=-vg*V;
    elseif ismember(i,production)
        %opportunity cost of lost batch
        Rusp(i,1)=-vp*V*concentration(i-3-ng+1);
    else
        Rusp(i,1)=0;
    end
end
%action add growth medium = 2
for i=1:Susp
    if ~ismember(i,[r,growth(1):growth(end)-1])
        Rusp(i,2)=-M;
    else
        Rusp(i,2)=-cg*V-fix;
    end
end
%action add production medium = 3
for i=1:Susp
    if ~ismember(i,[Gp,production(1):production(end)-1])
        Rusp(i,3)=-M;
```

```
    else
        Rusp(i,3)=-cp*V-fix;
    end
end
%action harvest/ dump = 4
for i=1:Susp
    if ismember(i,growth)
        %dump growth medium: capture value of spent growth medium
        Rusp(i,4)=vg*V-fix;
    elseif ismember(i,[r,u])
        Rusp(i,4)=-fix;
    elseif ismember(i,production)
        %only feasible if a(DSP)=accept
        %set rewards in system matrix
        Rusp(i,4)=-M;
    else
        Rusp(i,4)=-M;
    end
end
%action prepare = 5
for i=1:Susp
    if i==e
        Rusp(i,5)=-(cg+cb)*V-fix;
    else
        Rusp(i,5)=-M;
    end
end
%action harvest/ dump + prepare = 6
for i=1:Susp
    if ismember(i,growth)
        %dump growth medium: capture value of spent growth medium - cost
        %of preparing
        Rusp(i,6)=vg*V-(cg+cb)*V-fix;
    elseif ismember(i,[r,u])
        Rusp(i,6)=-fix;
    elseif ismember(i,production)
        %only feasible if a(DSP)=accept
        %set rewards in system matrix
        Rusp(i,6)=-M;
    else
        Rusp(i,6)=-M;
    end
end
%Downstream Process Reactor
%R(current, action)
%let do nothing be action 1
for i=1:(Sdsp-1)
    Rdsp(i,1)=0;
end
%infeasible action: have to exchange resin when min capacity is reached
Rdsp(Sdsp,1)=-M;
%let accept be action 2
%rewards are set in system reward matrix due to influence of USP reactor
%state
```

```
for i=1:(Sdsp-1)
    Rdsp(i,2)=-M;
end
%infeasible action: have to exchange resin when min capacity is reached
Rdsp(Sdsp,2)=-M;
%let exchange resin be action 3
for i=1:Sdsp
    Rdsp(i,3)=-Cresin;
end
%Rtotal=zeros(Susp*Sdsp,Ausp*Adsp);
Rall=constructSystemRewards(Rusp,Rdsp);
%USP: let harvest/ dump be action 4
%USP: let harvest/ dump + prep be action 6
%DSP: let accept be action 2
%rewards are set in system reward matrix due to influence of USP reactor
%state on DSP decision's reward
capacity = setCapacities(ncap,minimumCapacity);
for UspStage=1:Susp
    %during the production phase of the USP, harvesting is feasible ...
    if ismember(UspStage,production)
        %... only if DSP capacity has not passed its minimum viable capacity
        for DspStage=1:ncap-1
            %Position in rewards matrix
            x=(UspStage-1)*Sdsp+DspStage;

            ProdStage=UspStage-3-ng+1;

            %(4,2) and (6,2) are feasible
            a2=2;
            a1=4;
            y=(a1-1)*Adsp+a2;
            Rall(x,y)=vp*V*concentration(ProdStage)*capacity(DspStage)-fix;
            a1=6;
            y=(a1-1)*Adsp+a2;
            Rall(x,y)=vp*V*concentration(ProdStage)*capacity(DspStage)-fix-(cg+cb)*V-fix;
        end
    end
end
%Check model
mdp_check(Pall,Rall);
%Set discount rate
discount = lambda;
averageReward = 0;
%Solve MDP
if solutionMethod=="average"
    [policy, average_reward] = mdp_relative_value_iteration(Pall,Rall);
    averageReward = average_reward;
elseif solutionMethod=="policyIteration"
    [V, policy] = mdp_policy_iteration(Pall, Rall, discount);
elseif solutionMethod=="valueIteration"
```

```
    [policy] = mdp_value_iteration(Pall, Rall, dis-
count);
elseif solutionMethod=="LP"
    [V, policy] = mdp_LP(Pall, Rall, discount);
    disp("here");
else
    disp("No viable solution method defined!");
end
%Make policy legible
readableSolution = constructReadableSolution(pol-
icy);
policyMatrixReadable = constructPolicyMa-trix(reada-
bleSolution,Susp,Sdsp);
% Titre maximizing policy
% Uncomment code sections
% Assumptions:
% - Converts all Gp into production states
% - Continues production until last production state
% - exchanges resin only in state ncap
% PolicyMaxTitre = zeros(Susp*Sdsp,1);
%
% Set breakpoint here and paste policy data from Ex-
cel workbook into the
% PolicyMaxTitre variable; resume
%
% readableSolutionMaxTitre = constructReadableSolu-
tion(PolicyMaxTitre);
% policyMatrixReadable = constructPolicyMatrix(read-
ableSolutionMaxTitre,Susp,Sdsp);
%
% ValueMaxTitre = mdp_eval_policy_itera-
tive(Pall,Rall,discount,PolicyMaxTitre);
%
% policyMatrixReadable = constructPolicyMatrix(read-
ableSolutionMaxTitre,Susp,Sdsp);
disp("Done! Have a look at the readable policy ma-
trix.")
```

```
function vector = setConcentrations(production-
Stages,maxConcentration)

vector = zeros(1,productionStages);

%simplified linear relationship between number of
production stages and concentration
% 0 at start of p1
% max concentration in p_np
for index=1:productionStages
    vector(index)=(maxConcentration/(production-
Stages-1)*(index-1))/1000;
end
```

```
function Ptotal = constructSystemProbabili-
ties(P1,P2)

    sa = size(P1);
    sb = size(P2);
```

```

    Ptotal=zeros(sa(1)*sb(1),sa(2)*sb(2),sa(3)*sb(3));

    %multiply transition probability P(j,i) in first matrix with the entire
    %second matrix
    for i = 1:sa(1)
        for j = 1:sa(2)
            %position in systems rewards matrix
            x1 = (i-1)*sb(1)+1;
            x2 = x1+sb(1)-1;
            y1 = (j-1)*sb(2)+1;
            y2 = y1+sb(2)-1;

            for a1 = 1:sa(3)
                for a2=1:sb(3)
                    Ptotal(x1:x2,y1:y2,(a1-1)*sb(3)+a2) = P1(i,j,a1)*P2(:,:,a2);
                end
            end
        end
    end
end
```

```
function R = constructSystemRewards(R1,R2)

    sa=size(R1);
    sb=size(R2);

    for s1=1:sa(1)
        for s2=1:sb(1)
            for a1=1:sa(2)
                for a2=1:sb(2)
                    %Position in system reward matrix
                    x = (s1-1)*sb(1)+s2;
                    y = (a1-1)*sb(2)+a2;
                    R(x,y)=R1(s1,a1)+R2(s2,a2);
                end
            end
        end
end
```

```
function vector = setCapacities(capacityStages,minCapacity)

vector = zeros(1,capacityStages);

for index=1:4
    vector(index)=1;
end

if minCapacity==0.65 && capacityStages==12
    % Liu et al (2014) case
```

```
    for index=5:capacityStages-1
        vector(index)=vector(index-1)-0.05;
    end
else
    % otherwise linear decay to minimum capacity
    for index=5:capacityStages-1
        vector(index)=1-(1-minCapacity)/(capacityStages-1-3)*(index-3);
    end
end
```

```
function readableSolution = constructReadableSolution(policy)

Ausp = 6;
Adsp = 3;

lookupTable = strings(1,Ausp*Adsp);

%Construct lookup list
for action1=1:Ausp
    for action2=1:Adsp
        indexNumber = (action1-1)*Adsp+action2;
        lookupTable(indexNumber) = strcat(string(action1),string(action2));
    end
end


readableSolution = [];

for index=1:length(policy)

    readableSolution = [readableSolution; lookupTable(policy(index))];
end
```

```
function outputMatrix = constructPolicyMatrix(policy,Susp,Sdsp)

outputMatrix = strings(Susp,Sdsp);

for UspState = 1:Susp
    for DspState = 1:Sdsp
        policyIndex = (UspState-1)*Sdsp + DspState;
        outputMatrix(UspState,DspState) = policy(policyIndex);
   end
end
```

Appendix 8: Model and hypotheses-test code for the 1:1 TPA case study with longer decision epochs; only parts of files differing from the 1:1 version are shown

```
function [V, policyMatrixReadable, averageReward] = MarkovDecisionProcess(lambda,processParameters,solutionMethod)
clc
disp("One USP reactor into one chromatography reactor")

%Process parameters

V=processParameters(1);
cb=processParameters(2);
cg=processParameters(3);
cp=processParameters(4);
vp=processParameters(5);
% value of growth medium per L: concentration of 1.5mg of TPA in spent
% growth medium times the per gram value of un-purified TPA
vg=vp*1.5e-3;
fix=processParameters(6);
Cresin=processParameters(7);

p = processParameters(8);

pNoDecay = processParameters(9);
pDecayByOne = processParameters(10);
pDecayByTwo = processParameters(11);

minimumCapacity = processParameters(12);

%sufficiently large number to disincentivize infeasible actions
M=99999999;

%Upstream reactor states
%let "empty"                                 be state 1
%let "ready"                                 be state 2
%let "growth i"                              be state 3..3+ng
%let "production j"                          be state 3+ng+1..3+ng+1+np
%let "upset"                                 be state 3+ng+1+np+1

%number of growth and production states, respectively
ng = 3;
np = 12;

e=1; r=2;

growth = r+1:r+1+ng-1;
production = growth(end)+1:growth(end)+1+np-1;
```

```
Gp = growth(3);

u = production(end)+1;

Susp=u;

productionProbabilities = zeros(length(production),0);

declinePhase=[0.84,0.34];

productionProbabilities(1:length(production)-length(declinePhase)-1)=p;
productionProbabilities(length(production)-length(declinePhase):length(production)-1)=declinePhase(1:length(declinePhase));
productionProbabilities(length(production))=1;

concentration = setConcentrations(np,processParameters(end));
…
```

Appendix 9: Model and hypotheses-test code for the 2:1 TPA case study; only files differing from the 1:1 version are shown

```
%=============================================================================
%
% Run the 2:1 TPA case study with the parameters set below.
%
%=============================================================================


% Set process parameters
% Syntax: [reactor volume,
%           cost of microcarriers ($/L),
%           cost of growth medium ($/L),
%           cost of production medium ($/L),
%           value of TPA ($/g),
%           fixed costs of operations ($)
%           cost of resin,
%           probability of successful upstream transition,
%           probability of no resin decay,
%           probability of resin decay by one step,
%           probability of resin decay by two steps,
%           minimum capacity of resin (\in [0..1])
%           maximum concentration of tPa in production medium (g/L)], cf. Datar, 1993: tPa production in CHO cells
processParameters = [160, 0, 12.8, 2, 24000, 100, 96480, 0.978, 0.05, 0.90, 0.05, 0.65, 33.5];

```

```
% Discount factor not used for average reward criterion
discountFactor = 0.99;

% must be \in {"average","policyIteration","valueIteration","LP"}
solutionMethod = "average";

tic;
[V, policy, averageReward] = MarkovDecisionProcess(discountFactor,processParameters,solutionMethod);
timeParallel = toc;

runTests=false;

if runTests==true
    % Caution for average reward criterion!
    % Expected time: up to 8 hours = 300s*50*2 / 3600s/h
    numberOfRuns = 50;

    % Hypo a: cheaper resin -> higher average reward
    % Hypo b: cheaper resin -> earlier resin exchange
    firstExchange = zeros(numberOfRuns,1);
    for i=1:numberOfRuns
        costOfResin(i) = 192960-192960/(numberOfRuns-1)*(i-1);
    end
    policiesCostOfResin = [];
    averageRewardsCostOfResin = [];
    timingCostOfResin = [];
    valuesCostOfResin = [];
    for i=1:numberOfRuns
        processParameters(7) = costOfResin(i);
        tic;
        [V, policy, averageReward] = MarkovDecisionProcess(discountFactor,processParameters,solutionMethod);
        timeCOR = toc;
        timingCostOfResin = [timingCostOfResin; timeCOR];

        firstExchange(i) = minNumberOfBatchesPurified(policy);

        valuesCostOfResin = [valuesCostOfResin; V(1)];
        averageRewardsCostOfResin = [averageRewardsCostOfResin; averageReward];
        policiesCostOfResin = [policiesCostOfResin; policy];
    end

    %Reset processParameter(7) if both test run back to back
    processParameters(7) = 96480;
```

```
    % Hypo c: steeper decay and lower minimum capacity -> higher average
    % reward
    % Hypo d: steeper decay and lower minimum capacity -> later resin
    % exchange
    for i=1:numberOfRuns
        minimumCapacitiesCap(i) = 1.0-1.0/(numberOfRuns-1)*(i-1);
    end
    policiesMinCap = [];
    averageRewardsMinCap = [];
    timingMinCap = [];
    valuesMinCap = [];
    firstExchangeCap = zeros(numberOfRuns,1);

    for i=1:numberOfRuns
        processParameters(12) = minimumCapacitiesCap(i);
        tic;
        [V, policy, averageReward] = MarkovDecisionProcess(discountFactor,processParameters,solutionMethod);
        timeMinCap = toc;

        timingMinCap = [timingMinCap; timeMinCap];

        firstExchangeCap(i) = minNumberOfBatchesPurified(policy);

        valuesMinCap = [valuesMinCap; V(1)];
        averageRewardsMinCap = [averageRewardsMinCap; averageReward];
        policiesMinCap = [policiesMinCap; policy];
    end
end
```

```
function [V, policiesForAllDSPStates, averageReward] = MarkovDecisionProcess(lambda,processParameters,solutionMethod)
clc
disp("Two parallel USP reactors into one chromatography column")

%Process parameters

V=processParameters(1);
cb=processParameters(2);
cg=processParameters(3);
cp=processParameters(4);
vp=processParameters(5);
% value of growth medium per L: concentration of 1.5mg of TPA in spent
% growth medium times the per gram value of un-purified TPA
vg=vp*1.5e-3;
```

```
fix=processParameters(6);
Cresin=processParameters(7);

p = processParameters(8);

pNoDecay = processParameters(9);
pDecayByOne = processParameters(10);
pDecayByTwo = processParameters(11);

minimumCapacity = processParameters(12);
maxConcentration = processParameters(end);

%sufficiently large number to disincentivize infeasible actions
M=99999999;

%Upstream reactor states
%let "empty"                    be state 1
%let "ready"                    be state 2
%let "growth i"                 be state 3..3+ng
%let "production i"             be state 3+ng+1..3+ng+1+np
%let "upset"                    be state 3+ng+1+np+1

%number of growth and production states, respectively
ng = 3;
np = 12;

e=1; r=2;

growth = r+1:r+1+ng-1;
production = growth(end)+1:growth(end)+1+np-1;

Gp = growth(3);

u = production(end)+1;

Susp=u;

productionProbabilities = zeros(length(production),0);

% Deterioration over the last 1/6 of the production phase
declinePhase=[0.84,0.34];

productionProbabilities(1:length(production)-length(declinePhase)-1)=p;
productionProbabilities(length(production)-length(declinePhase):length(production)-1)=declinePhase(1:length(declinePhase));
productionProbabilities(length(production))=1;

```

```
concentration = setConcentrations(np,maxConcentration);

%Downstream reactor states
%let "resin capacity to purify batch i" be state 1..ncap

%must be > 3
ncap = 12;

Sdsp=ncap;

%USP reactor actions
%let "none" be action 1
%let "add growth medium" be action 2
%let "add production medium" be action 3
%let "harvest" be action 4
%let "prepare" be action 5
%let "harvest and prepare" be action 6
actionsUsp=["none","addgm","addpm","harvest","prep","hprep"];

Ausp=length(actionsUsp);

%DSP reactor actions
%let "none" be action 1
%let "accept" be action 2
%let "exchange resin" be action 3
actionsDsp=["none","accept","exresin"];

Adsp=length(actionsDsp);

%P(current, next, action)
Pusp=zeros(Susp,Susp,Ausp);

%action none = 1
Pusp(e,e,1)=1;
Pusp(r,r,1)=p;
Pusp(r,u,1)=1-p;

%doing nothing during growth -> upset
for i=growth(1):growth(end)
    Pusp(i,u,1)=1;
end

%doing nothing during production -> upset
for i=production(1):production(end)
    Pusp(i,u,1)=1;
end
```

```

Pusp(u,u,1)=1;

%action add growth medium = 2
for i=1:Susp
    % addgm during growth but before last growth state or in ready to start
    % growth
    if ismember(i,[r,growth]) && i<growth(end)
        Pusp(i,i+1,2)=p;
        Pusp(i,u,2)=1-p;
    else
        %infeasible everywhere else
        Pusp(i,u,2)=1;
    end
end



%action add production medium = 3
for i=1:Susp
    % addpm to continue production until second to last production state
    if ismember(i,production(1):production(end)-1)
        %fetch probabilities from probability vector including
        %deterioration towards the end
        ProdStage2=i-3-ng+1;
        probability = productionProbabilities(ProdStage2);

        %row sums must equal 1, state transitions either without problem or
        %transitions to upset
        Pusp(i,i+1,3)=probability;
        Pusp(i,u,3)=1-probability;
    % addpm to convert production-competent growth state into production state
    elseif ismember(i,Gp)
        Pusp(i,production(1),3)=p;
        Pusp(i,u,3)=1-p;
    else
        %infeasible everywhere else
        Pusp(i,u,3)=1;
    end
end

%action harvest/ dump = 4
Pusp(r,e,4)=1;

% dump all growth medium to abort process
for i=growth(1):growth(end)
    Pusp(i,e,4)=1;
end

% harvest production medium is only feasible if a(DSP)=accept and a(USPj)!=4 or 6
for i=production(1):production(end)
```

```
    Pusp(i,e,4)=1;
end

% dump upset reactor's contents to abort process
Pusp(u,e,4)=1;

Pusp(e,e,4)=1;

%action prepare = 5
for i=1:Susp
    %infeasible everywhere except if reactor is empty
    if i==e
        Pusp(e,r,5)=p;
        Pusp(e,u,5)=1-p;
    else
        Pusp(i,u,5)=1;
    end
end

%action harvest/dump and prepare = 6
Pusp(r,r,6)=Pusp(e,r,5);
Pusp(r,u,6)=Pusp(e,u,5);

% dump all growth medium and prepare reactor to re-start process
for i=growth(1):growth(end)
    Pusp(i,r,6)=Pusp(e,r,5);
    Pusp(i,u,6)=Pusp(e,u,5);
end

% harvest production medium is only feasible if a(DSP) = accept and prepare
% reactor to restart process
for i=production(1):production(end)
    %only feasible for USPi if a(DSP)=accept and a(USPj)!=4 or 6
    Pusp(i,r,6)=Pusp(e,r,5);
    Pusp(i,u,6)=Pusp(e,u,5);
end

Pusp(u,r,6)=Pusp(e,r,5);
Pusp(u,u,6)=Pusp(e,u,5);

Pusp(e,r,6)=Pusp(e,r,5);
Pusp(e,u,6)=Pusp(e,u,5);


%P(current, next, action)
Pdsp=zeros(Sdsp,Sdsp,Adsp);

%action none = 1
for i = 1:ncap
    %feasible only if i<ncap, must exchange if ncap is reached
    Pdsp(i,i,1) = 1;
end

```

```
%action accept = 2
%resin is used once for purification and its capacity deteriorates
%stochastically
for i = 1:(ncap-1)
    %feasible only before resin has to be exchanged and if a(USP)=harvest OR harvest+prep
    if i==ncap-1
        %simplifying assumption as discussed in thesis
        Pdsp(i,i,2) = pNoDecay;
        Pdsp(i,i+1,2) = pDecayByOne+pDecayByTwo;
    else
        Pdsp(i,i,2) = pNoDecay;
        Pdsp(i,i+1,2) = pDecayByOne;
        Pdsp(i,i+2,2) = pDecayByTwo;
    end
end
%infeasible
Pdsp(ncap,ncap,2) = 1;


%action exchange resin = 3
%resin is exchanged for a full capacity one
for i = 1:ncap
    Pdsp(i,1,3) = 1;
end

disp("Combining probability matrices...");
PUspTwo = constructSystemProbabilities(Pusp,Pusp);
tic
Pall = constructSystemProbabilities(PUspTwo,Pdsp);
toc

%Test for  non-one row sums in transition probabilities
% rowSumProductUSP = 1;
% for i=1:Susp
%     rowSumProductUSP = rowSumProductUSP.*sum(Pusp(i,:,:));
% end
%
% rowSumProductDSP = 1;
% for i=1:Sdsp
%     rowSumProductDSP = rowSumProductDSP.*sum(Pdsp(i,:,:));
% end
%
% rowSumProductAll = 1;
% for i=1:Susp*Sdsp
%     rowSumProductAll = rowSumProductAll.*sum(Pall(i,:,:));
% end

%Upstream Process Reactor
%R(current, action)
%costs are negative, revenues positive
```

```
Rusp=zeros(Susp,Ausp);

%action none = 1, zero reward
for i=1:Susp
    if ismember(i,growth)
        %opportunity cost of lost growth medium because system moves to upset state
        Rusp(i,1)=-vg*V;
    elseif ismember(i,production)
        %cost of lost batch
        Rusp(i,1)=-vp*V*concentration(i-3-ng+1);
    else
        Rusp(i,1)=0;
    end
end

%action add growth medium = 2
for i=1:Susp
    if ~ismember(i,[r,growth(1):growth(end)-1])
        Rusp(i,2)=-M;
    else
        Rusp(i,2)=-cg*V-fix;
    end
end


%action add production medium = 3
for i=1:Susp
    if ~ismember(i,[Gp,production(1):production(end)-1])
        % addpm is infeasible in last production state and outside of:
        % production phase and production competent growth states
        Rusp(i,3)=-M;
    else
        Rusp(i,3)=-cp*V-fix;
    end
end

%action harvest/ dump = 4
for i=1:Susp
    if ismember(i,growth)
        %dump growth medium: capture value of spent growth medium
        Rusp(i,4)=vg*V-fix;
    elseif ismember(i,[r,u])
        Rusp(i,4)=-fix;
    elseif ismember(i,production)
        %only feasible in USPi if a(DSP)=accept and a(USPj)!= 4 or 6
        %set rewards in system matrix
        Rusp(i,4)=-M;
    else
        Rusp(i,4)=-M;
    end
end

```

```
%action prepare = 5
for i=1:Susp
    if i==e
        %prepare reactor with growth medium and microcarriers
        Rusp(i,5)=-(cg+cb)*V-fix;
    else
        Rusp(i,5)=-M;
    end
end
%action harvest/ dump + prepare = 6
for i=1:Susp
    if ismember(i,growth)
        %dump growth medium: capture value of spent growth medium - cost
        %of preparing reactor afterwards
        Rusp(i,6)=vg*V-(cg+cb)*V-fix;
    elseif ismember(i,[r,u])
        Rusp(i,6)=-fix;
    elseif ismember(i,production)
        %only feasible in USPi if a(DSP)=accept and a(DSPj)= 4 or 6
        %set rewards in system matrix
        Rusp(i,6)=-M;
    else
        Rusp(i,6)=-M;
    end
end
%Downstream Process Reactor
%R(current, action)
Rdsp=zeros(Sdsp,Adsp);
%let do nothing be action 1
for i=1:(Sdsp-1)
    Rdsp(i,1)=0;
end
%infeasible action: have to exchange resin when min capacity is reached
Rdsp(Sdsp,1)=-M;
%let accept be action 2
%rewards are set in system reward matrix due to influence of USP reactor
%state
for i=1:(Sdsp-1)
    Rdsp(i,2)=-M;
end
%infeasible action: have to exchange resin when min capacity is reached
Rdsp(Sdsp,2)=-M;
%let exchange resin be action 3
for i=1:Sdsp
    Rdsp(i,3)=-Cresin;
end
```

```
disp("Combining rewards matrices...");
RUspTwo = constructSystemRewards(Rusp,Rusp);
tic
Rall = constructSystemRewards(RUspTwo,Rdsp);
toc

%for USP i and j
%   let harvest/ dump be action 4
%    let harvest/ dump + prep be action 6
%for DSP:
%   let accept be action 2
%rewards are set in system reward matrix due to influence of USP reactor's
%state on DSP decision's reward
capacity = setCapacities(ncap,minimumCapacity);

disp("Setting system level rewards...");
tic
% set rewards for feasible actions where state interdependencies exist;
% infeasibilities are already modeled above
for Usp1State = 1:Susp
    for Usp2State = 1:Susp
        if (ismember(Usp1State,production) && ~ismember(Usp2State,production))
            % Case 1: USP 1 is in the production phase but USP 2 is not:
            % (4,x,2) and (6,x,2) are feasible where x is a feasible action
            % in USP2,
            % and the DSP is not in the last state
            for DspState = 1:Sdsp-1
                for Usp2Action = 1:Ausp
                    % Position in rewards matrix
                    % determine row index
                    x=(Usp1State-1)*Susp*Sdsp + (Usp2State-1)*Sdsp + DspState;

                    % (4,x,2) is feasible
                    Usp1Action = 4;
                    DspAction = 2;
                    % Position in rewards matrix
                    % determine column index
                    y=(Usp1Action-1)*Ausp*Adsp + (Usp2Action-1)*Adsp + DspAction;


                    % Production phase cycle of USP 1 for
                    % concentration of product
                    Usp1ProdState=Usp1State-3-ng+1;

                    % Set reward for feasible action:
                    % Total reward consists of revenue from
                    % purified product
                    % - fixed costs
```

```
                        % + rewards caused in USP2 in that decision epoch
                        Rall(x,y)=vp*V*concentration(Usp1ProdState)*capacity(DspState)-fix+Rusp(Usp2State,Usp2Action);


                        % (6,x,2) is feasible
                        Usp1Action = 6;
                        DspAction = 2;
                        % Position in rewards matrix
                        % determine column index
                        y=(Usp1Action-1)*Ausp*Adsp + (Usp2Action-1)*Adsp + DspAction;

                        Rall(x,y)=vp*V*concentration(Usp1ProdState)*capacity(DspState)-fix-(cg+cb)*V-fix+Rusp(Usp2State,Usp2Action);
                    end
                end

            elseif (ismember(Usp1State,production) && ismember(Usp2State,production))
            % Case 2: USP 1 is in the production phase and USP 2 is as well:
            % (4,x,2),(6,x,2),(x,4,2),(x,6,2) are feasible, where x is a
            % feasible action in USP 1 and USP 2, resp., which is not 4 or 6,
            % and the DSP is not in the last state
                for DspState = 1:Sdsp-1
                    for Usp1Action = 1:Ausp
                        for Usp2Action = 1:Ausp
                            % Case 2a: harvesting in USP 2 is infeasible while
                            % USP 1 is harvested
                            if (ismember(Usp1Action,[4,6]) && ~ismember(Usp2Action,[4,6]))
                                % Position in rewards matrix
                                % determine row index
                                x=(Usp1State-1)*Susp*Sdsp + (Usp2State-1)*Sdsp + DspState;

                                % (4,x,2) is feasible
                                Usp1Action = 4;
                                DspAction = 2;
                                % Position in rewards matrix
                                % determine column index
                                y=(Usp1Action-1)*Ausp*Adsp + (Usp2Action-1)*Adsp + DspAction;


                                % Production phase cycle of USP 1 for
                                % concentration of product
```

```
                    Usp1ProdState=Usp1State-3-ng+1;

                    % Set reward for feasible action:
                    % Total reward consists of revenue from
                    % purified product
                    % - fixed costs
                    % + rewards caused in USP2 in that decision epoch
                    Rall(x,y)=vp*V*concentration(Usp1ProdState)*capacity(DspState)-fix+Rusp(Usp2State,Usp2Action);


                    % (6,x,2) is feasible
                    Usp1Action = 6;
                    DspAction = 2;
                    % Position in rewards matrix
                    % determine column index
                    y=(Usp1Action-1)*Ausp*Adsp + (Usp2Action-1)*Adsp + DspAction;

                    Rall(x,y)=vp*V*concentration(Usp1ProdState)*capacity(DspState)-fix-(cg+cb)*V-fix+Rusp(Usp2State,Usp2Action);
                % Case 2b: harvesting in USP 1 is infeasible while
                % USP 2 is harvested
                elseif (~ismember(Usp1Action,[4,6]) && ismember(Usp2Action,[4,6]))
                    % Position in rewards matrix
                    % determine row index
                    x=(Usp1State-1)*Susp*Sdsp + (Usp2State-1)*Sdsp + DspState;

                    % (x,4,2) is feasible
                    Usp2Action = 4;
                    DspAction = 2;
                    % Position in rewards matrix
                    % determine column index
                    y=(Usp1Action-1)*Ausp*Adsp + (Usp2Action-1)*Adsp + DspAction;


                    % Production phase cycle of USP 1 for
                    % concentration of product
                    Usp2ProdState=Usp2State-3-ng+1;

                    % Set reward for feasible action:
```

```
                            % Total reward consists of revenue from
                            % purified product
                            % - fixed costs
                            % + rewards caused in USP2 in that decision epoch
                            Rall(x,y)=vp*V*concentration(Usp2ProdState)*capacity(DspState)-fix+Rusp(Usp1State,Usp1Action);


                            % (x,6,2) is feasible
                            Usp2Action = 6;
                            DspAction = 2;
                            % Position in rewards matrix
                            % determine column index
                            y=(Usp1Action-1)*Ausp*Adsp + (Usp2Action-1)*Adsp + DspAction;

                            Rall(x,y)=vp*V*concentration(Usp2ProdState)*capacity(DspState)-fix-(cg+cb)*V-fix+Rusp(Usp1State,Usp1Action);
                        end
                    end
                end
            end
        elseif (~ismember(Usp1State,production) && ismember(Usp2State,production))
            % Case 3: USP 2 is in the production phase but USP 1 is not:
            % (x,4,2) and (x,6,2) are feasible where x is a feasible action
            % in USP1,
            % and the DSP is not in the last state
            for DspState = 1:Sdsp-1
                for Usp1Action = 1:Ausp
                    % Position in rewards matrix
                    % determine row index
                    x=(Usp1State-1)*Susp*Sdsp + (Usp2State-1)*Sdsp + DspState;

                    % (4,x,2) is feasible
                    Usp2Action = 4;
                    DspAction = 2;
                    % Position in rewards matrix
                    % determine column index
                    y=(Usp1Action-1)*Ausp*Adsp + (Usp2Action-1)*Adsp + DspAction;


                    % Production phase cycle of USP 1 for
                    % concentration of product
                    Usp2ProdState=Usp2State-3-ng+1;

                    % Set reward for feasible action:
```

```
                    % Total reward consists of revenue from
                    % purified product
                    % - fixed costs
                    % + rewards caused in USP2 in that decision epoch
                    Rall(x,y)=vp*V*concentration(Usp2ProdState)*capacity(DspState)-fix+Rusp(Usp1State,Usp1Action);


                    % (x,6,2) is feasible
                    Usp2Action = 6;
                    DspAction = 2;
                    % Position in rewards matrix
                    % determine column index
                    y=(Usp1Action-1)*Ausp*Adsp + (Usp2Action-1)*Adsp + DspAction;

                    Rall(x,y)=vp*V*concentration(Usp2ProdState)*capacity(DspState)-fix-(cg+cb)*V-fix+Rusp(Usp1State,Usp1Action);
                end
            end

        end
    end
end
toc



%Check model
mdp_check(Pall,Rall);

%Set discount rate
discount = lambda;

averageReward = 0;


disp("Solving MDP...");
tic
%Solve MDP
if solutionMethod=="average"
    [policy, average_reward] = mdp_relative_value_iteration(Pall,Rall,0.1,10);
    averageReward = average_reward;
elseif solutionMethod=="policyIteration"
    [V, policy] = mdp_policy_iteration(Pall, Rall, discount);
elseif solutionMethod=="valueIteration"
    [policy] = mdp_value_iteration(Pall, Rall, discount);
elseif solutionMethod=="LP"
    [V, policy] = mdp_LP(Pall, Rall, discount);
else
```

```
    disp("Error: No viable solution method defined!");
end
toc

% Generate a readable policy table for both USP reactors for one given DSP
% state

policiesForAllDSPStates = [];
%Make policy legible
readableSolution = constructReadableSolution(policy);
for DspStateForPolicy = 1:Sdsp
    policyMatrixReadable = constructPolicyMatrix(readableSolution,Susp,Sdsp,DspStateForPolicy);
    policiesForAllDSPStates = [policiesForAllDSPStates; policyMatrixReadable];
end
disp("Done! Have a look at the readable policy matrix.")
```

```
function readableSolution = constructReadableSolution(policy)

Ausp = 6;
Adsp = 3;

lookupTable = strings(1,Ausp*Ausp*Adsp);

%Construct lookup table
for Usp1Action = 1:Ausp
    for Usp2Action = 1:Ausp
        for DspAction = 1:Adsp
            indexNumber = (Usp1Action-1)*Ausp*Adsp + (Usp2Action-1)*Adsp + DspAction;
            lookupTable(indexNumber) = strcat(string(Usp1Action),string(Usp2Action),string(DspAction));
        end
    end
end

readableSolution = [];

for index=1:length(policy)
    readableSolution = [readableSolution; lookupTable(policy(index))];
end
```

```
function outputMatrix = constructPolicyMatrix(policy,Susp,Sdsp,DspState)

outputMatrix = strings(Susp,Susp);
if DspState == 0
    % Set to 0 to indicate that policy for only one reactor is relevant
    for Usp1State = 1:Susp
```

```
            for DspState = 1:Sdsp
              Usp2State = 1;
                policyIndex = (Usp1State-1)*Susp*Sdsp + (Usp2State-1)*Sdsp + DspState;
                outputMatrix(Usp1State,DspState) = policy(policyIndex);
            end
        end
else
        for Usp1State = 1:Susp
            for Usp2State = 1:Susp
                policyIndex = (Usp1State-1)*Susp*Sdsp + (Usp2State-1)*Sdsp + DspState;
                outputMatrix(Usp1State,Usp2State) = policy(policyIndex);
            end
        end
End
```

```
function MinimumNumberOfBatchesBeforeExchange = minNumberOfBatchesPurified(policy)

% Takes a matrix of dimensions 12*18 x 18 holding the optimal policies for
% the 2:1 case study for all 12 DSP states after each other
% Outputs the number of batches which were purified before the a
% resin exchange is prescribed for any USP states


%Only find strings for which DSP action = 3
allExchanges = find(~cellfun(@isempty,regexp(policy,'^[0-9]+\s?3$')));

% find row index of all entries
DSPStateDuringExchange = mod(allExchanges,18*12); % for final DSP state always zero

DSPStateDuringExchange = DSPStateDuringExchange ./ 18;

% sort ascendingly
DSPStateDuringExchangeSorted = sort(DSPStateDuringExchange);

% first entries may be zero, because row index of final DSP
% state (ncap) is zero: 12 mod 18*12 = 0
% if only zeros exist, resin exchange takes place in ncap-1
i = 1;
while DSPStateDuringExchangeSorted(i)==0
    if i == length(DSPStateDuringExchangeSorted)
        % step out if there are only zeros in the array ->
        break;
    end
```

```
    i=i+1;
end

% Number of batches purified before resin exchange is the index of the DSP
% state in which exchange takes place
MinimumNumberOfBatchesBeforeExchange = floor(DSPStateDuringExchangeSorted(i));
```

```
function reward = getReward(Rall,Usp1State,Usp2State,DspState,Usp1Action,Usp2Action,DspAction,Susp,Sdsp,Ausp,Adsp)

    x=(Usp1State-1)*Susp*Sdsp + (Usp2State-1)*Sdsp + DspState;
    y=(Usp1Action-1)*Ausp*Adsp + (Usp2Action-1)*Adsp + DspAction;
     reward = Rall(x,y);
```

Appendix 10: Total expected discounted reward maximizing policy for the operation of one TPA production reactor and one chromatography step

	100%	100%	100%	100%	95%	90%	85%	80%	75%	70%	65%	0%
e						**51**						**53**
r												
g_1										**61**		
g_2					**21**							**23**
g_3												
g_4												
g_5												
g_6												
g_7												
g_8												
p_1						**31**						
p_2												
…												**33**
p_{28}												
p_{29}												
p_{30}												
p_{31}					**62**							
…												
p_{36}												**13**
u						**61**						**63**

Appendix 11: Average expected reward maximizing policy for the operation of one TPA production reactor and one chromatography step, with 36 hour-decision epochs

	100%	100%	100%	100%	95%	90%	85%	80%	75%	70%	65%	0%
e						**51**						**53**
r												
g_1						**21**						**23**
g_2												
g_3												
…						**31**						
p_8												
p_9										**33**		
p_{10}												
p_{11}					**62**							
p_{12}												**13**
u						**61**						**63**

Appendix 12: 1:1 case sensitivity analysis data: Resin material cost on average reward and earliness of first exchange

USP:DSI	Cost of resin exchange	Minimum capacity allowed	Probability of	Probability of	Probability of	Probability of	Average reward	Total disc. Rewa	Lateness of first exchange	Figure x	Time to
1:1		1.11E-16					1175.329089		205		
1:1		0.020408163					1175.79962		222		
1:1		0.040816327					1176.544042		227		
1:1		0.06122449					1176.639694		195		
1:1		0.081632653					1176.864961		227		
1:1		0.102040816					1176.197569		227		
1:1		0.12244898					1176.012587		260		
1:1		0.142857143					1177.247156		274		
1:1		0.163265306					1177.459708		270		
1:1		0.183673469					1178.537032		237		
1:1		0.204081633					1180.702866		237		
1:1		0.224489796					1182.810365		251		
1:1		0.244897959					1184.840017		239		
1:1		0.265306122					1186.824152		248		
1:1		0.285714286					1189.257588		248		
1:1		0.306122449					1191.269273		255		
1:1		0.326530612					1193.607792		264		
1:1		0.346938776					1196.215067		266		
1:1		0.367346939					1198.938969		236		
1:1		0.387755102					1202.001763		242		
1:1		0.408163265					1204.851429		256		
1:1		0.428571429					1206.446398		274		
1:1		0.448979592					1209.521597		300		
1:1		0.469387755					1211.847936		314		
1:1		0.489795918					1212.830886		321		
1:1		0.510204082					1216.925168		303		
1:1		0.530612245					1220.992238		296		
1:1		0.551020408					1224.136405		311		
1:1		0.571428571					1230.586921		320		
1:1		0.591836735					1235.666639		321		
1:1		0.612244898					1241.024221		321		
1:1		0.632653061					1244.618755		321		
1:1		0.653061224					1251.374746		344		
1:1		0.673469388					1255.477073		368		
1:1		0.693877551					1261.195743		368		
1:1		0.714285714					1269.734404		387		
1:1		0.734693878					1276.216583		393		
1:1		0.755102041					1285.411144		415		
1:1		0.775510204					1292.920965		415		
1:1		0.795918367					1300.327278		415		
1:1		0.816326531					1307.705413		415		
1:1		0.836734694					1308.020391		462		
1:1		0.857142857					1303.265634		509		
1:1		0.87755102					1302.28161		518		
1:1		0.897959184					1302.809816		518		
1:1		0.918367347					1297.878868		518		
1:1		0.93877551					1298.052363		518		
1:1		0.959183673					1301.199811		518		
1:1		0.979591837					1305.779133		518		
1:1		1					1314.276893		518		

Appendix 13: 1:1 case sensitivity analysis data: Minimum resin capacity on average reward and earliness of first exchange

USP:DSI	Cost of resin exchange	Minimum capacity allowed	Probability of	Probability of	Probability of	Probability of	Average reward	Total disc. Rewa	Lateness of first exchange
1:1		1.11E-16					1175.329089		205
1:1		0.020408163					1175.79962		222
1:1		0.040816327					1176.544042		227
1:1		0.06122449					1176.639694		195
1:1		0.081632653					1176.864961		227
1:1		0.102040816					1176.197569		227
1:1		0.12244898					1176.012587		260
1:1		0.142857143					1177.247156		274
1:1		0.163265306					1177.459708		270
1:1		0.183673469					1178.537032		237
1:1		0.204081633					1180.702866		237
1:1		0.224489796					1182.810365		251
1:1		0.244897959					1184.840017		239
1:1		0.265306122					1186.824152		248
1:1		0.285714286					1189.257588		248
1:1		0.306122449					1191.269273		255
1:1		0.326530612					1193.607792		264
1:1		0.346938776					1196.215067		266
1:1		0.367346939					1198.938969		236
1:1		0.387755102					1202.001763		242
1:1		0.408163265					1204.851429		256
1:1		0.428571429					1206.446398		274
1:1		0.448979592					1209.521597		300
1:1		0.469387755					1211.847936		314
1:1		0.489795918					1212.830886		321
1:1		0.510204082					1216.925168		303
1:1		0.530612245					1220.992238		296
1:1		0.551020408					1224.136405		311
1:1		0.571428571					1230.586921		320
1:1		0.591836735					1235.666639		321
1:1		0.612244898					1241.024221		321
1:1		0.632653061					1244.618755		321
1:1		0.653061224					1251.374746		344
1:1		0.673469388					1255.477073		368
1:1		0.693877551					1261.195743		368
1:1		0.714285714					1269.734404		387
1:1		0.734693878					1276.216583		393
1:1		0.755102041					1285.411144		415
1:1		0.775510204					1292.920965		415
1:1		0.795918367					1300.327278		415
1:1		0.816326531					1307.705413		415
1:1		0.836734694					1308.020391		462
1:1		0.857142857					1303.265634		509
1:1		0.87755102					1302.28161		518
1:1		0.897959184					1302.809816		518
1:1		0.918367347					1297.878868		518
1:1		0.93877551					1298.052363		518
1:1		0.959183673					1301.199811		518
1:1		0.979591837					1305.779133		518
1:1		1					1314.276893		518

Appendix 14: 1:1 and 2:1 case miscellaneous sensitivity analysis data

USP:DSP	Cost of resin exchange	Minimum capacity allowed	Probability of	Probability of	Probability of	Probability of	Average reward	Total disc. Rewa	Lateness of first exchange	Figure x	Time to
1:1											
1:1			0.6				0	-2.93978E-11			
1:1			0.7				126.6352698	9418.053633			
1:1			0.8				309.5945555	27370.97143			
1:1			0.9				512.3744645	47291.94035			
1:1			0.99				1150.580042	104771.8477			
1:1			0.993				1272.495425				
1:1			0.995				1346.92611	126032.7268			
1:1			0.9975				1347.614584				
1:1			0.99775				1323.44168				
1:1			0.998				1289.085155	139411.3159			
1:1			0.999				973.0200819	144015.0209			
1:1			0.9					47291.94035			
1:1			0.95					57896.31373			
1:1			0.993					104771.8477			
1:1			1					148664.5009			
1:1				0.15	0.85	0	1326.995662	95173.68786			
1:1				0.1	0.9	0	1301.475933	101710.6245			
1:1				0.05	0.95	0	1272.411591	117948.3462			
1:1				0.01	0.99	0	1239.044184	117589.337			
1:1				0	1	0	1232.332033	117498.9031			
1:1				0.05	0.9	0.05	1272.495425	117394.7423			
1:1				0.025	0.95	0.025	1255.23136	117446.4988			
2:1	192960						-50794.43301				315.76245
2:1	165394.2857						-37437.61876				317.02087
2:1	137828.5714						-30485.93662				271.10169
2:1	110262.8571						-30485.93662				288.38027
2:1	82697.14286						-13219.94537				305.9227
2:1	55131.42857						-818.5414904				289.10565
2:1	27565.71429						656.047204				313.15566
2:1	0						937.2394146				268.06971

Appendix 15: 2:1 case sensitivity analysis data: Resin material cost on average reward and earliness of first exchange

USP:DSP	Cost of resin exchange	Minimum capacity allowed	Probability of	Probability of	Probability of	Probability of	Average reward	Total disc. Rewa	Lateness of first exchange	Figure x	Time to
2:1	192960						-50794.43301		10		204.80187
2:1	189022.0408						-47543.48186		10		121.34982
2:1	185084.0816						-44292.53071		10		125.5229
2:1	181146.1224						-41845.21103		10		120.76234
2:1	177208.1633						-41845.21103		10		124.03694
2:1	173270.2041						-41845.21103		10		125.51783
2:1	169332.2449						-40452.96694		10		120.98936
2:1	165394.2857						-37437.61876		10		122.01293
2:1	161456.3265						-34422.27057		10		123.42319
2:1	157518.3673						-32857.70429		10		121.98657
2:1	153580.4082						-31922.75911		10		116.02192
2:1	149642.449						-30987.81393		9		127.93956
2:1	145704.4898						-30485.93662		9		121.14548
2:1	141766.5306						-30485.93662		9		124.23252
2:1	137828.5714						-30485.93662		9		111.42547
2:1	133890.6122						-30485.93662		9		127.76812
2:1	129952.6531						-30485.93662		9		127.56259
2:1	126014.6939						-30485.93662		9		128.90198
2:1	122076.7347						-30485.93662		9		175.85862
2:1	118138.7755						-30485.93662		9		126.06733
2:1	114200.8163						-30485.93662		9		168.00204
2:1	110262.8571						-30485.93662		9		122.23682
2:1	106324.898						-30485.93662		9		121.9555
2:1	102386.9388						-28677.38368		9		122.7595
2:1	98448.97959						-25662.0355		9		166.07275
2:1	94511.02041						-22580.64526		9		128.01889
2:1	90573.06122						-19497.46715		9		118.22047
2:1	86635.10204						-16372.47921		9		119.18525
2:1	82697.14286						-13219.94537		9		123.31668
2:1	78759.18367						-9870.275479		9		177.22811
2:1	74821.22449						-6424.362235		9		119.76268
2:1	70883.26531						-4263.06195		9		118.64677
2:1	66945.30612						-3401.931835		8		117.82048
2:1	63007.34694						-2540.80172		8		121.94963
2:1	59069.38776						-1679.671605		8		117.68345
2:1	55131.42857						-818.5414904		8		122.03409
2:1	51193.46939						42.58862451		7		144.42245
2:1	47255.5102						287.8039774		7		122.17765
2:1	43317.55102						349.0461058		7		122.4064
2:1	39379.59184						514.4037001		6		120.26236
2:1	35441.63265						561.3114073		6		119.60135
2:1	31503.67347						624.8123992		6		120.89883
2:1	27565.71429						656.047204		5		118.30766
2:1	23627.7551						688.3133911		5		122.54681
2:1	19689.79592						751.8143831		5		122.28837
2:1	15751.83673						751.8143831		4		163.51537
2:1	11813.87755						815.315375		4		130.10747
2:1	7875.918367						829.3774265		4		122.22986
2:1	3937.959184						878.816367		3		123.23131
2:1	0						937.2394146		2		121.93243

Appendix 16: 2:1 case sensitivity analysis data: Minimum resin capacity on average reward and earliness of first exchange

USP:DSI	Cost of resin exchange	Minimum capacity allowed	Probability of	Probability of	Probability of	Probability of	Average reward	Total disc. Rewa	Lateness of first exchange	Figure x	Time to
2:1		1					-42319.96475		11		126.28342
2:1		0.979591837					-41688.34348		10		127.19446
2:1		0.959183673					-40879.53339		10		118.81572
2:1		0.93877551					-40010.85096		10		154.84439
2:1		0.918367347					-39142.16853		10		118.78379
2:1		0.897959184					-38273.35611		10		120.59154
2:1		0.87755102					-37404.48301		9		123.68914
2:1		0.857142857					-36535.60991		9		120.61067
2:1		0.836734694					-35666.73681		9		119.03511
2:1		0.816326531					-34797.86371		9		136.96801
2:1		0.795918367					-34064.77432		9		119.04385
2:1		0.775510204					-33565.65575		9		122.92038
2:1		0.755102041					-32191.24442		9		178.52046
2:1		0.734693878					-30611.17134		9		120.32223
2:1		0.714285714					-29566.07291		9		120.89078
2:1		0.693877551					-27504.34241		9		128.48948
2:1		0.673469388					-25932.74755		9		119.57396
2:1		0.653061224					-24358.38821		9		133.38122
2:1		0.632653061					-22784.02887		9		119.75503
2:1		0.612244898					-21209.66953		9		120.89477
2:1		0.591836735					-19635.3102		9		122.08184
2:1		0.571428571					-18060.95086		9		122.64823
2:1		0.551020408					-16486.59152		9		185.05343
2:1		0.530612245					-14912.23218		8		119.6111
2:1		0.510204082					-13337.87285		8		121.52442
2:1		0.489795918					-11769.27015		8		118.93197
2:1		0.469387755					-10219.56362		8		121.40322
2:1		0.448979592					-8669.857085		8		119.54922
2:1		0.428571429					-7114.488311		8		123.544
2:1		0.408163265					-5552.04386		7		124.58239
2:1		0.387755102					-3988.529333		7		120.58155
2:1		0.367346939					-3228.560943		7		120.90337
2:1		0.346938776					-2910.122872		7		122.83529
2:1		0.326530612					-2591.684802		7		120.547
2:1		0.306122449					-2343.648935		7		119.54733
2:1		0.285714286					-2246.208687		6		120.42769
2:1		0.265306122					-2141.337204		6		132.57063
2:1		0.244897959					-1316.319183		6		124.40129
2:1		0.224489796					-965.3998225		6		118.64601
2:1		0.204081633					-537.1699927		6		126.9427
2:1		0.183673469					-108.9401628		6		179.86169
2:1		0.163265306					93.04506506		6		127.33254
2:1		0.142857143					130.0714215		6		123.7029
2:1		0.12244898					188.5212115		6		122.67386
2:1		0.102040816					260.4774909		6		123.31961
2:1		0.081632653					294.2290652		5		118.83308
2:1		0.06122449					294.2290652		5		125.24967
2:1		0.040816327					294.2290652		5		122.50014
2:1		0.020408163					294.2290652		5		120.57225
2:1		1.11E-16					294.2290652		5		120.15547

Appendix 17: Average reward maximizing policy for the operation of one TPA production reactor and one chromatography step under deterministic performance decay

<table>
<tr><th></th><th>100%</th><th>100%</th><th>100%</th><th>100%</th><th>95%</th><th>90%</th><th>85%</th><th>80%</th><th>75%</th><th>70%</th><th>65%</th><th>0%</th></tr>
<tr><td>e</td><td colspan="11">51</td><td>53</td></tr>
<tr><td>r</td><td colspan="11" rowspan="5">21</td><td rowspan="5">23</td></tr>
<tr><td>g_1</td></tr>
<tr><td>…</td></tr>
<tr><td>g_5</td></tr>
<tr><td>g_6</td><td colspan="11" rowspan="6">31</td><td rowspan="14">33</td></tr>
<tr><td>g_7</td></tr>
<tr><td>g_8</td></tr>
<tr><td>p_1</td></tr>
<tr><td>…</td></tr>
<tr><td>p_{21}</td></tr>
<tr><td>p_{22}</td><td colspan="8">31</td><td colspan="3">33</td></tr>
<tr><td>p_{23}</td><td colspan="11" rowspan="4">31</td></tr>
<tr><td>…</td></tr>
<tr><td>p_{28}</td></tr>
<tr><td>p_{29}</td></tr>
<tr><td>p_{30}</td><td colspan="7">62</td><td colspan="4">33</td></tr>
<tr><td>p_{31}</td><td colspan="9">62</td><td colspan="2">33</td></tr>
<tr><td>…</td><td colspan="11" rowspan="2">62</td></tr>
<tr><td>p_{36}</td><td>13</td></tr>
<tr><td>u</td><td colspan="11">61</td><td>63</td></tr>
</table>

Appendix 18: Fermentation titer maximizing policy $\pi_{MaxTiter}$

<table>
<tr><th></th><th>100%</th><th>100%</th><th>100%</th><th>100%</th><th>95%</th><th>90%</th><th>85%</th><th>80%</th><th>75%</th><th>70%</th><th>65%</th><th>0%</th></tr>
<tr><td>e</td><td></td><td colspan="10">51</td><td>53</td></tr>
<tr><td>r
g_1
…
g_5</td><td></td><td colspan="10">21</td><td>23</td></tr>
<tr><td>g_6
g_7
g_8
p_1
…
p_{35}</td><td rowspan="3"></td><td colspan="10">31</td><td>33</td></tr>
<tr><td>p_{36}</td><td colspan="10">62</td><td>13</td></tr>
<tr><td>u</td><td colspan="10">61</td><td>63</td></tr>
</table>

Appendix 19: Average reward maximizing policy for the operation of two parallel bioreactors into a single chromatography column, which is at full resin capacity before the third batch

<table>
<tr><th></th><th>e</th><th>r</th><th>g_1</th><th>g_2</th><th>g_3</th><th>p_1</th><th>…</th><th>p_8</th><th>p_9</th><th>p_{10}</th><th>p_{11}</th><th>p_{12}</th><th>u</th></tr>
<tr><td>e</td><td>551</td><td colspan="3">521</td><td colspan="5">531</td><td colspan="3">562</td><td>561</td></tr>
<tr><td>r
g_1
g_2</td><td>251</td><td colspan="3">221</td><td colspan="5">231</td><td colspan="3">262</td><td>261</td></tr>
<tr><td>g_3</td><td rowspan="5">351</td><td colspan="3" rowspan="5">321</td><td colspan="5" rowspan="5">331</td><td colspan="3" rowspan="6">362
Continue production in reactor one, harvest reactor two</td><td rowspan="4">361</td></tr>
<tr><td>p_1</td></tr>
<tr><td>…</td></tr>
<tr><td>p_8</td></tr>
<tr><td>p_9</td><td rowspan="4">662</td></tr>
<tr><td>p_{10}</td><td rowspan="3">652</td><td colspan="3" rowspan="3">622</td><td colspan="5" rowspan="3">632</td></tr>
<tr><td>p_{11}</td><td colspan="2" rowspan="2"></td><td></td></tr>
<tr><td>p_{12}</td><td>612</td></tr>
<tr><td>u</td><td>651</td><td colspan="3">621</td><td colspan="4">631</td><td colspan="4">662</td><td>661</td></tr>
</table>

Appendix 20: Average reward maximizing policy for the operation of two parallel bioreactors into a single chromatography column, after the fourth to ninth batches

	e	r	g_1	g_2	g_3	p_1	…	p_8	p_9	p_{10}	p_{11}	p_{12}	u
e	551		521				531				562		561
r													
g_1	251		221				231				262		261
g_2													
g_3													
p_1											362		361
…	351		321				331						
p_8											Continue production in reactor one, harvest reactor two		
p_9													
p_{10}													662
p_{11}	652		622				632						
p_{12}												162	
u	651		621				631				662		661

Appendix 21: Average reward maximizing policy for the operation of two parallel bioreactors into a single chromatography column, before the tenth batch

	e	r	g_1	g_2	g_3	p_1	p_2	p_3	…	p_7	p_8	p_9	p_{10}	p_{11}	p_{12}	u
e	551		521						531					562		561
r																
g_1	251		221						231					262		261
g_2																
g_3																
p_1														362		
p_2																
p_3	351		321			331			331							361
…																
p_7																
p_8												Continue production, exchange resin				
p_9										333						
p_{10}																
p_{11}	652		622							632						662
p_{12}															162	
u	651		621						631					662		661

Appendix 22: Average reward maximizing policy policy for the operation of two parallel bio-reactors into a single chromatography column with 65 % remaining resin capacity

	e	r	g_1	g_2	g_3	p_1	p_2	p_3	p_4	…	p_7	p_8	p_9	p_{10}	p_{11}	p_{12}	u
e	551		521						531					533	562		561
r			221						231								
g_1	251		261	621	631									233	262		261
g_2																	
g_3																	
p_1			361						331								
p_2																	
p_3	351																361
p_4															362		
…			321											333			
p_7																	
p_8																	
p_9	353		323														363
p_{10}																362	
p_{11}	652		622						632						331		662
p_{12}																162	
u	651		621						631					633	662		661

Appendix 23: Average reward maximizing policy for the operation of two parallel bioreactors into a single chromatography column, which is completely depleted

	e	r	g_1	g_2	g_3	p_1	…	p_{11}	p_{12}	u
e	553		523				533		513	563
r										
g_1	253		223				233		213	263
g_2										
g_3										
p_1	353		323				333		313	363
…										
p_{11}										
p_{12}	153		123				133			163
u	653		623				633		613	663