

Stockinger, Quirin

Article

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

Junior Management Science (JUMS)

Provided in Cooperation with:

Junior Management Science e. V.

Suggested Citation: Stockinger, Quirin (2020) : Stochastic Optimization of Bioreactor Control Policies Using a Markov Decision Process Model, Junior Management Science (JUMS), ISSN 2942-1861, Junior Management Science e. V., Planegg, Vol. 5, Iss. 1, pp. 50-80, <https://doi.org/10.5282/jums/v5i1pp50-80>

This Version is available at:

<https://hdl.handle.net/10419/294923>

Standard-Nutzungsbedingungen:

Die Dokumente auf EconStor dürfen zu eigenen wissenschaftlichen Zwecken und zum Privatgebrauch gespeichert und kopiert werden.

Sie dürfen die Dokumente nicht für öffentliche oder kommerzielle Zwecke vervielfältigen, öffentlich ausstellen, öffentlich zugänglich machen, vertreiben oder anderweitig nutzen.

Sofern die Verfasser die Dokumente unter Open-Content-Lizenzen (insbesondere CC-Lizenzen) zur Verfügung gestellt haben sollten, gelten abweichend von diesen Nutzungsbedingungen die in der dort genannten Lizenz gewährten Nutzungsrechte.

Terms of use:

Documents in EconStor may be saved and copied for your personal and scholarly purposes.

You are not to copy documents for public or commercial purposes, to exhibit the documents publicly, to make them publicly available on the internet, or to distribute or otherwise use the documents in public.

If the documents have been made available under an Open Content Licence (especially Creative Commons Licences), you may exercise further usage rights as specified in the indicated licence.



<https://creativecommons.org/licenses/by/4.0/>



Online-Appendix zu

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

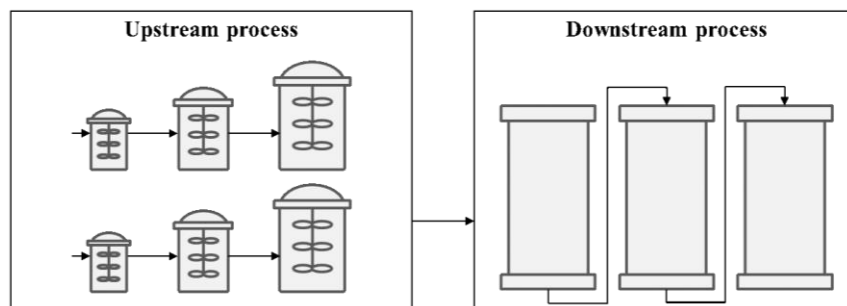
Quirin Stockinger

Technische Universität München

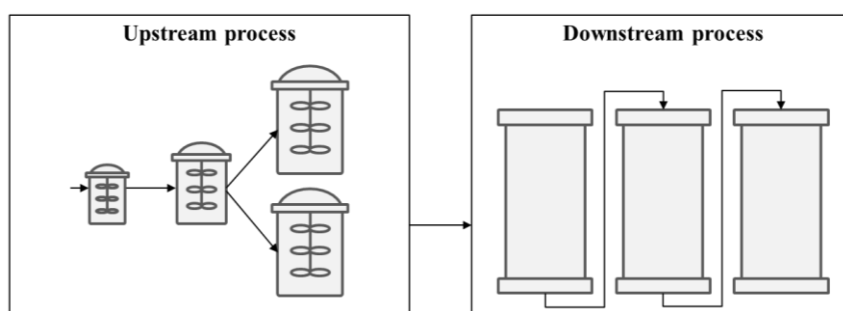
Junior Management Science 5(1) (2020) 50-80

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

Refer- ence⁸	System- level policies	USP & DSP de- cisions	Produc- tion un- cer- tainty	Resin perfor- mance decay	DSP un- certain- ties	Finan- cial trade-off
Schmidt (1996)	✓		✓			✓
Liu et al. (2014)		✓	✓	✓ ⁹		✓
Liu et al. (2016)		✓	✓		✓	✓
Martagan et al. (2016)	✓		✓			✓
Martagan et al. (2018)	✓		✓		✓	✓
Martagan et al. (Ac- cepted/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

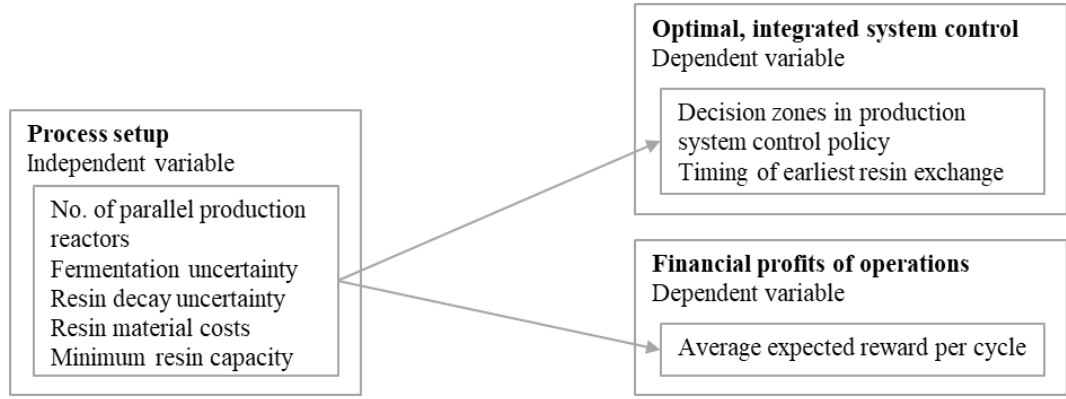
Determine:

- 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 %
$p(cap_m, cap_{m+\{0,1,2\}}, 2)$	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

```

1.  %=====
    %=====
2.  %
3.  % Run the MDP case study with the parameters set be-
    % low.
4.  %
5.  %=====
    %=====
6.
7.
8.  % Set process parameters
9.  % Syntax: [reactor volume,
10. %         cost of microcarriers ($/L),
11. %         cost of growth medium ($/L),
12. %         cost of production medium ($/L),
13. %         value of TPA ($/g),
14. %         fixed costs of operations ($)
15. %         cost of resin,
16. %         probability of successful upstream tran-
    % sition,
17. %         probability of no resin decay,
18. %         probability of resin decay by one step,
19. %         probability of resin decay by two steps,
20. %         minimum capacity of resin (\in [0..1])
21. %         maximum concentration of tPa in produc-
    % tion medium (mg/L)], cf Datar, 1993: tPa production
    % in CHO cells
22. processParameters = [160, 0, 12.8, 2, 24000, 100,
    96480, 0.993, 0.05, 0.9, 0.05, 0.65, 33.5];
23.
24.
25. % set MDP parameters
26. discountFactor = 0.99;
27.
28. % must be \in {"average","policyIteration","valueIt-
    % eration","LP"}
29. solutionMethod = "average";
30.
31.
32. tic;
33. [V, policy, averageReward] = MarkovDecisionPro-
    % cess(discountFactor,processParameters,solution-
    % Method);
34. time=toc;
35.
36.
37. runTests=false;
38.
39. if runTests==true
40.     numberOfRuns = 50;
41.
42.     % Hypo a: cheaper resin -> earlier first ex-
    % change
43.     % Hypo b: cheaper resin -> higher average reward
44.     firstExchange = zeros(numberOfRuns);

```

```

45.         firstExchangeDuringProduction = ze-
ros(length(firstExchange));
46.         timingCostOfResin = zeros(length(firstEx-
change));
47.         for i=1:length(firstExchange)
48.             costOfResin(i) = 192960-
192960/(length(firstExchange)-1)*(i-1);
49.         end
50.         policiesCostOfResin = [];
51.         averageRewardsCostOfResin = [];
52.         for i=1:length(firstExchange)
53.             processParameters(7) = costOfResin(i);
54.             tic;
55.             [policy, averageReward] = MarkovDecisionPro-
cess(discountFactor,processParameters,solution-
Method);
56.             timingCostOfResin(i) = toc;
57.             firstExchange(i) = find(~cellfun(@is-
empty,regexp(policy,'\w+3')),1);
58.
59.             averageRewardsCostOfResin = [averageRe-
wardsCostOfResin; averageReward];
60.             policiesCostOfResin = [policiesCostOfResin;
policy];
61.         end
62.
63.         x=costOfResin.';
64.
65.         %Reset processParameter(7) if both test run back
to back
66.         processParameters(7) = 96480;
67.
68.
69.         % Hypo c: steeper decay and lower minimum capac-
ity -> earlier first exchange
70.         % Hypo d: steeper decay and lower minimum capac-
ity -> higher average
71.         % reward
72.         firstExchangeCap = zeros(numberOfRuns);
73.         firstExchangeDuringProductionCap = ze-
ros(length(firstExchangeCap));
74.         timingMinCap = zeros(length(firstExchangeCap));
75.
76.         for i=1:length(firstExchangeCap)
77.             minimumCapacitiesCap(i) = 1.0-
1.0/(length(firstExchangeCap)-1)*(i-1);
78.         end
79.
80.         averageRewardsMinCap = [];
81.         policiesMinimumCapacity = [];
82.         for i=1:length(firstExchangeCap)
83.             processParameters(12) = minimumCapacities-
Cap(i);
84.             tic;
85.             [policy, averageReward] = MarkovDecisionPro-
cess(discountFactor,processParameters,solution-
Method);
86.             timingCostOfResin(i) = toc;

```



```

87.
88.         firstExchangeCap(i) = find(~cellfun(@isempty, regexp(policy, '\w+3')),1);
89.
90.         averageRewardsMinCap = [averageRewardsMinCap; averageReward];
91.         policiesMinimumCapacity = [policiesMinimumCapacity; policy];
92.     end
93. end

1.  function [V, policyMatrixReadable, averageReward] =
    MarkovDecisionProcess(lambda, processParameters, solutionMethod)
2.  clc
3.  disp("One USP reactor into one chromatography reactor")
4.  %Process parameters
5.  V=processParameters(1);
6.  cb=processParameters(2);
7.  cg=processParameters(3);
8.  cp=processParameters(4);
9.  vp=processParameters(5);
10. % value of growth medium per L: concentration of
    1.5mg of TPA in spent
11. % growth medium times the per gram value of unpurified TPA
12. vg=vp*1.5e-3;
13. fix=processParameters(6);
14. Cresin=processParameters(7);
15. p = processParameters(8);
16. pNoDecay = processParameters(9);
17. pDecayByOne = processParameters(10);
18. pDecayByTwo = processParameters(11);
19. minimumCapacity = processParameters(12);
20. %sufficiently large number to disincentivize infeasible actions
21. M=99999999;
22. %Upstream reactor states
23. %let "empty" be state
    1
24. %let "ready" be state
    2
25. %let "growth i" be state
    3..3+ng
26. %let "production j" be state
    3+ng+1..3+ng+1+np
27. %let "upset" be state
    3+ng+1+np+1
28. %number of growth and production states, respectively
29. ng = 8;
30. np = 36;
31. e=1; r=2;
32. growth = r+1:r+1+ng-1;
33. production = growth(end)+1:growth(end)+1+np-1;
34. Gp = growth(6):growth(8);

```

```

35. u = production(end)+1;
36. Susp=u;
37. productionProbabilities = zeros(length(production),0);
38. declinePhase=[0.84,0.67,0.5,0.34,0.17,0.05];
39. productionProbabilities(1:length(production)-length(declinePhase)-1)=p;
40. productionProbabilities(length(production)-length(declinePhase):length(production)-1)=declinePhase(1:length(declinePhase));
41. productionProbabilities(length(production))=1;
42. concentration = setConcentrations(np,processParameters(end));
43. %Downstream reactor states
44. %let "resin capacity in cycle i" be state
    1..ncap
45. %must be > 3
46. ncap = 12;
47. Sdsp=ncap;
48. %USP reactor actions
49. %let "none" be action 1
50. %let "add growth medium" be action 2
51. %let "add production medium" be action 3
52. %let "harvest" be action 4
53. %let "prepare" be action 5
54. %let "harvest and prepare" be action 6
55. actionsUsp=["none","addgm","addpm","harvest","prep","hprep"];
56. Ausp=length(actionsUsp);
57. %DSP reactor actions
58. %let "none" be action 1
59. %let "accept" be action 2
60. %let "exchange resin" be action 3
61. actionsDsp=["none","accept","exresin"];
62. Adsp=length(actionsDsp);
63. %P(current, next, action)
64. Pusp=zeros(Susp,Susp,Ausp);
65. %action none = 1
66. Pusp(e,e,1)=1;
67. Pusp(r,r,1)=p;
68. Pusp(r,u,1)=1-p;
69. %doing nothing during growth -> upset
70. for i=growth(1):growth(end)
71.     Pusp(i,u,1)=1;
72. end
73. %doing nothing during production -> upset
74. for i=production(1):production(end)
75.     Pusp(i,u,1)=1;
76. end

```

```

77. Pusp(u,u,1)=1;
78. %action add growth medium = 2
79. for i=1:Susp
80.     % addgm during growth but before last growth
    state or in ready to start
81.     % growth
82.     if ismember(i,[r,growth(1):growth(end)-1])
83.         Pusp(i,i+1,2)=p;
84.         Pusp(i,u,2)=1-p;
85.     else
86.         %infeasible everywhere else
87.         Pusp(i,u,2)=1;
88.     end
89. end
90. %action add production medium = 3
91. for i=1:Susp
92.     % addpm to continue production until second to
    last pro-duction state
93.     if ismember(i,production(1):production(end)-1)
94.         %fetch probabilities from probability vector
    including
95.         %deterioration towards the end
96.         ProdStage2=i-3-ng+1;
97.         probability = productionProbabilities(Prod-
    Stage2);
98.
99.         %row sums must equal 1, state transitions
    either with-out problem or
100.        %transitions to upset
101.        Pusp(i,i+1,3)=probability;
102.        Pusp(i,u,3)=1-probability;
103.        % addpm to convert production-competent growth
    state into production state
104.        elseif ismember(i,Gp)
105.            Pusp(i,production(1),3)=p;
106.            Pusp(i,u,3)=1-p;
107.        else
108.            %infeasible everywhere else
109.            Pusp(i,u,3)=1;
110.        end
111. end
112. %action harvest/ dump = 4
113. Pusp(r,e,4)=1;
114. % dump all growth medium to abort process
115. for i=growth(1):growth(end)
116.     Pusp(i,e,4)=1;
117. end
118. % harvest production medium is only feasible if
    a(DSP)=accept and a(USPj)!=4 or 6
119. for i=production(1):production(end)
120.     Pusp(i,e,4)=1;
121. end
122. % dump upset reactor's contents to abort process
123. Pusp(u,e,4)=1;
124. Pusp(e,e,4)=1;
125. %action prepare = 5
126. for i=1:Susp

```

```

127.      %infeasible everywhere except if reactor is
      empty
128.      if i==e
129.          Pusp(e,r,5)=p;
130.          Pusp(e,u,5)=1-p;
131.      else
132.          Pusp(i,u,5)=1;
133.      end
134. end
135. %action harvest/dump and prepare = 6
136. Pusp(r,r,6)=Pusp(e,r,5);
137. Pusp(r,u,6)=Pusp(e,u,5);
138. % dump all growth medium and prepare reactor to re-
      start pro-cess
139. for i=growth(1):growth(end)
140.     Pusp(i,r,6)=Pusp(e,r,5);
141.     Pusp(i,u,6)=Pusp(e,u,5);
142. end
143. % harvest production medium is only feasible if
      a(DSP) = ac-cept and prepare
144. % reactor to restart process
145. for i=production(1):production(end)
146.     %only feasible for USPi if a(DSP)=accept and
      a(USPj)!=4 or 6
147.     Pusp(i,r,6)=Pusp(e,r,5);
148.     Pusp(i,u,6)=Pusp(e,u,5);
149. end
150. Pusp(u,r,6)=Pusp(e,r,5);
151. Pusp(u,u,6)=Pusp(e,u,5);
152. Pusp(e,r,6)=Pusp(e,r,5);
153. Pusp(e,u,6)=Pusp(e,u,5);
154. %P(current, next, action)
155. Pdsp=zeros(Sdsp,Sdsp,Adsp);
156. %action none = 1
157. for i = 1:ncap
158.     %feasible only if i<ncap, must exchange if ncap
      is reached
159.     Pdsp(i,i,1) = 1;
160. end
161. %action accept = 2
162. %resin is used once for purification and its capac-
      ity deterio-rates
163. %stochastically
164. for i = 1:(ncap-1)
165.     %feasible only before resin has to be exchanged
      and if a(USP)=harvest OR harvest+prep
166.     if i==ncap-1
167.         %simplifying assumption as discussed in the-
      sis
168.         Pdsp(i,i,2) = pNoDecay;
169.         Pdsp(i,i+1,2) = pDecayByOne+pDecayByTwo;
170.     else
171.         Pdsp(i,i,2) = pNoDecay;
172.         Pdsp(i,i+1,2) = pDecayByOne;
173.         Pdsp(i,i+2,2) = pDecayByTwo;
174.     end
175. end
176. %infeasible

```

```

177. Pdsp(ncap,ncap,2) = 1;
178. %action exchange resin = 3
179. %resin is exchanged for a full capacity one
180. for i = 1:ncap
181.     Pdsp(i,1,3) = 1;
182. end
183. Pall = constructSystemProbabilities(Pusp,Pdsp);
184. %Test for non-one row sums in transition probabilities
185. % rowSumProductUSP = 1;
186. % for i=1:Susp
187. %     rowSumProductUSP = row-
SumProductUSP.*sum(Pusp(i,:,:));
188. % end
189. %
190. % rowSumProductDSP = 1;
191. % for i=1:Sdsp
192. %     rowSumProductDSP = row-
SumProductDSP.*sum(Pdsp(i,:,:));
193. % end
194. %
195. % rowSumProductAll = 1;
196. % for i=1:Susp*Sdsp
197. %     rowSumProductAll = rowSumProduc-
tAll.*sum(Pall(i,:,:));
198. % end
199. %Upstream Process Reactor
200. %R(current, action)
201. %Maximize rewards (costs are negative, rewards posi-
tive)
202. Rusp=zeros(Susp,Ausp);
203. Rdsp=zeros(Sdsp,Adsp);
204. %action none = 1
205. for i=1:Susp
206.     if ismember(i,growth)
207.         %opportunity cost of lost growth medium be-
cause system moves to upset state
208.         Rusp(i,1)=-vg*V;
209.     elseif ismember(i,production)
210.         %opportunity cost of lost batch
211.         Rusp(i,1)=-vp*V*concentration(i-3-ng+1);
212.     else
213.         Rusp(i,1)=0;
214.     end
215. end
216. %action add growth medium = 2
217. for i=1:Susp
218.     if ~ismember(i,[r,growth(1):growth(end)-1])
219.         Rusp(i,2)=-M;
220.     else
221.         Rusp(i,2)=-cg*V-fix;
222.     end
223. end
224. %action add production medium = 3
225. for i=1:Susp
226.     if ~ismember(i,[Gp,production(1):produc-
tion(end)-1])
227.         Rusp(i,3)=-M;

```

```

228.         else
229.             Rusp(i,3)=-cp*V-fix;
230.         end
231.     end
232. %action harvest/ dump = 4
233. for i=1:Susp
234.     if ismember(i,growth)
235.         %dump growth medium: capture value of spent
growth medium
236.         Rusp(i,4)=vg*V-fix;
237.     elseif ismember(i,[r,u])
238.         Rusp(i,4)=-fix;
239.     elseif ismember(i,production)
240.         %only feasible if a(DSP)=accept
241.         %set rewards in system matrix
242.         Rusp(i,4)=-M;
243.     else
244.         Rusp(i,4)=-M;
245.     end
246. end
247. %action prepare = 5
248. for i=1:Susp
249.     if i==e
250.         Rusp(i,5)=-(cg+cb)*V-fix;
251.     else
252.         Rusp(i,5)=-M;
253.     end
254. end
255. %action harvest/ dump + prepare = 6
256. for i=1:Susp
257.     if ismember(i,growth)
258.         %dump growth medium: capture value of spent
growth medium - cost
259.         %of preparing
260.         Rusp(i,6)=vg*V-(cg+cb)*V-fix;
261.     elseif ismember(i,[r,u])
262.         Rusp(i,6)=-fix;
263.     elseif ismember(i,production)
264.         %only feasible if a(DSP)=accept
265.         %set rewards in system matrix
266.         Rusp(i,6)=-M;
267.     else
268.         Rusp(i,6)=-M;
269.     end
270. end
271. %Downstream Process Reactor
272. %R(current, action)
273. %let do nothing be action 1
274. for i=1:(Sdsp-1)
275.     Rdsp(i,1)=0;
276. end
277. %infeasible action: have to exchange resin when min
capacity is reached
278. Rdsp(Sdsp,1)=-M;
279. %let accept be action 2
280. %rewards are set in system reward matrix due to in-
fluence of USP reactor
281. %state

```

```

282. for i=1:(Sdsp-1)
283.     Rdsp(i,2)=-M;
284. end
285. %infeasible action: have to exchange resin when min
    capacity is reached
286. Rdsp(Sdsp,2)=-M;
287. %let exchange resin be action 3
288. for i=1:Sdsp
289.     Rdsp(i,3)=-Cresin;
290. end
291. %Rtotal=zeros(Susp*Sdsp,Ausp*Adsp);
292. Rall=constructSystemRewards(Rusp,Rdsp);
293. %USP: let harvest/ dump be action 4
294. %USP: let harvest/ dump + prep be action 6
295. %DSP: let accept be action 2
296. %rewards are set in system reward matrix due to in-
    fluence of USP reactor
297. %state on DSP decision's reward
298. capacity = setCapacities(ncap,minimumCapacity);
299. for UspStage=1:Susp
300.     %during the production phase of the USP, har-
        vesting is feasible ...
301.         if ismember(UspStage,production)
302.             %... only if DSP capacity has not passed its
                minimum viable capacity
303.                 for DspStage=1:ncap-1
304.                     %Position in rewards matrix
305.                     x=(UspStage-1)*Sdsp+DspStage;
306.
307.                     ProdStage=UspStage-3-ng+1;
308.
309.                     %(4,2) and (6,2) are feasible
310.                     a2=2;
311.                     a1=4;
312.                     y=(a1-1)*Adsp+a2;
313.                     Rall(x,y)=vp*V*concentration(Prod-
                        Stage)*capacity(DspStage)-fix;
314.                     a1=6;
315.                     y=(a1-1)*Adsp+a2;
316.                     Rall(x,y)=vp*V*concentration(Prod-
                        Stage)*capacity(DspStage)-fix-(cg+cb)*V-fix;
317.                 end
318.             end
319.         end
320. %Check model
321. mdp_check(Pall,Rall);
322. %Set discount rate
323. discount = lambda;
324. averageReward = 0;
325. %Solve MDP
326. if solutionMethod=="average"
327.     [policy, average_reward] = mdp_rela-
        tive_value_iteration(Pall,Rall);
328.     averageReward = average_reward;
329. elseif solutionMethod=="policyIteration"
330.     [V, policy] = mdp_policy_iteration(Pall, Rall,
        discount);
331. elseif solutionMethod=="valueIteration"

```

```

332.     [policy] = mdp_value_iteration(Pall, Rall, dis-
        count);
333. elseif solutionMethod=="LP"
334.     [V, policy] = mdp_LP(Pall, Rall, discount);
335.     disp("here");
336. else
337.     disp("No viable solution method defined!");
338. end
339. %Make policy legible
340. readableSolution = constructReadableSolution(pol-
        icy);
341. policyMatrixReadable = constructPolicyMa-trix(reada-
        bleSolution,Susp,Sdsp);
342. % Titre maximizing policy
343. % Uncomment code sections
344. % Assumptions:
345. % - Converts all Gp into production states
346. % - Continues production until last production state
347. % - exchanges resin only in state ncap
348. % PolicyMaxTitre = zeros(Susp*Sdsp,1);
349. %
350. % Set breakpoint here and paste policy data from Ex-
        cel workbook into the
351. % PolicyMaxTitre variable; resume
352. %
353. % readableSolutionMaxTitre = constructReadableSolu-
        tion(PolicyMaxTitre);
354. % policyMatrixReadable = constructPolicyMatrix(read-
        ableSolutionMaxTitre,Susp,Sdsp);
355. %
356. % ValueMaxTitre = mdp_eval_policy_itera-
        tive(Pall,Rall,discount,PolicyMaxTitre);
357. %
358. % policyMatrixReadable = constructPolicyMatrix(read-
        ableSolutionMaxTitre,Susp,Sdsp);
359. disp("Done! Have a look at the readable policy ma-
        trix.")

1.  function vector = setConcentrations(production-
        Stages,maxConcentration)
2.
3.  vector = zeros(1,productionStages);
4.
5.  %simplified linear relationship between number of
        production stages and concentration
6.  % 0 at start of pl
7.  % max concentration in p_np
8.  for index=1:productionStages
9.      vector(index)=(maxConcentration/(production-
        Stages-1)*(index-1))/1000;
10. end

1.  function Ptotal = constructSystemProbabili-
        ties(P1,P2)
2.
3.      sa = size(P1);
4.      sb = size(P2);

```



```

5.
6.         Ptotal=zeros(
7.         ros(sa(1)*sb(1),sa(2)*sb(2),sa(3)*sb(3));
8.         %multiply transition probability P(j,i) in first
9.         %second matrix
10.        for i = 1:sa(1)
11.            for j = 1:sa(2)
12.                %position in systems rewards matrix
13.                x1 = (i-1)*sb(1)+1;
14.                x2 = x1+sb(1)-1;
15.                y1 = (j-1)*sb(2)+1;
16.                y2 = y1+sb(2)-1;
17.
18.                for a1 = 1:sa(3)
19.                    for a2=1:sb(3)
20.                        Ptotal(x1:x2,y1:y2,(a1-
21.                        1)*sb(3)+a2) = P1(i,j,a1)*P2(:,a2);
22.                    end
23.                end
24.            end
25.        end

1. function R = constructSystemRewards(R1,R2)
2.
3.         sa=size(R1);
4.         sb=size(R2);
5.
6.         for s1=1:sa(1)
7.             for s2=1:sb(1)
8.                 for a1=1:sa(2)
9.                     for a2=1:sb(2)
10.                        %Position in system reward ma-
11.                        trix
12.                        x = (s1-1)*sb(1)+s2;
13.                        y = (a1-1)*sb(2)+a2;
14.                        R(x,y)=R1(s1,a1)+R2(s2,a2);
15.                    end
16.                end
17.            end
18.        end

1. function vector = setCapacities(capacityStages,min-
2. Capacity)
3.
4.         vector = zeros(1,capacityStages);
5.
6.         for index=1:4
7.             vector(index)=1;
8.         end
9.
10.        if minCapacity==0.65 && capacityStages==12
11.            % Liu et al (2014) case

```

```

11.         for index=5:capacityStages-1
12.             vector(index)=vector(index-1)-0.05;
13.         end
14.     else
15.         % otherwise linear decay to minimum capacity
16.         for index=5:capacityStages-1
17.             vector(index)=1-(1-minCapacity)/(capacityStages-1-3)*(index-3);
18.         end
19.     end

1.  function readableSolution = constructReadableSolution(policy)
2.
3.  Ausp = 6;
4.  Adsp = 3;
5.
6.  lookupTable = strings(1,Ausp*Adsp);
7.
8.  %Construct lookup list
9.  for action1=1:Ausp
10.     for action2=1:Adsp
11.         indexNumber = (action1-1)*Adsp+action2;
12.         lookupTable(indexNumber) = strcat(string(action1),string(action2));
13.     end
14. end
15.
16.
17. readableSolution = [];
18.
19. for index=1:length(policy)
20.
21.     readableSolution = [readableSolution; lookupTable(policy(index))];
22. end

1.  function outputMatrix = constructPolicyMatrix(policy,Susp,Sdsp)
2.
3.  outputMatrix = strings(Susp,Sdsp);
4.
5.  for UspState = 1:Susp
6.     for DspState = 1:Sdsp
7.         policyIndex = (UspState-1)*Sdsp + DspState;
8.         outputMatrix(UspState,DspState) = policy(policyIndex);
9.     end
10. 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**

```

1.  function [V, policyMatrixReadable, averageReward] =
    MarkovDecisionProcess(lambda, processParameters, solutionMethod)
2.  clc
3.  disp("One USP reactor into one chromatography reactor")
4.
5.  %Process parameters
6.
7.  V=processParameters(1);
8.  cb=processParameters(2);
9.  cg=processParameters(3);
10. cp=processParameters(4);
11. vp=processParameters(5);
12. % value of growth medium per L: concentration of
    1.5mg of TPA in spent
13. % growth medium times the per gram value of un-purified TPA
14. vg=vp*1.5e-3;
15. fix=processParameters(6);
16. Cresin=processParameters(7);
17.
18. p = processParameters(8);
19.
20. pNoDecay = processParameters(9);
21. pDecayByOne = processParameters(10);
22. pDecayByTwo = processParameters(11);
23.
24. minimumCapacity = processParameters(12);
25.
26. %sufficiently large number to disincentivize infeasible actions
27. M=99999999;
28.
29. %Upstream reactor states
30. %let "empty" be state
    1
31. %let "ready" be state
    2
32. %let "growth i" be state
    3..3+ng
33. %let "production j" be state
    3+ng+1..3+ng+1+np
34. %let "upset" be state
    3+ng+1+np+1
35.
36. %number of growth and production states, respectively
37. ng = 3;
38. np = 12;
39.
40. e=1; r=2;
41.
42. growth = r+1:r+1+ng-1;
43. production = growth(end)+1:growth(end)+1+np-1;

```

```

44.
45. Gp = growth(3);
46.
47. u = production(end)+1;
48.
49. Susp=u;
50.
51. productionProbabilities = zeros(length(production),0);
52.
53. declinePhase=[0.84,0.34];
54.
55. productionProbabilities(1:length(production)-length(declinePhase)-1)=p;
56. productionProbabilities(length(production)-length(declinePhase):length(production)-1)=declinePhase(1:length(declinePhase));
57. productionProbabilities(length(production))=1;
58.
59. concentration = setConcentrations(np,processParameters(end));
60. ...

```

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

```

1.  %=====
2.  %
3.  % Run the 2:1 TPA case study with the parameters set
4.  % below.
5.  %=====
6.
7.
8.  % Set process parameters
9.  % Syntax: [reactor volume,
10. %         cost of microcarriers ($/L),
11. %         cost of growth medium ($/L),
12. %         cost of production medium ($/L),
13. %         value of TPA ($/g),
14. %         fixed costs of operations ($)
15. %         cost of resin,
16. %         probability of successful upstream transition,
17. %         probability of no resin decay,
18. %         probability of resin decay by one step,
19. %         probability of resin decay by two steps,
20. %         minimum capacity of resin (\in [0..1])
21. %         maximum concentration of tPa in production medium (g/L)], cf. Datar, 1993: tPa production in CHO cells
22. processParameters = [160, 0, 12.8, 2, 24000, 100,
23. 96480, 0.978, 0.05, 0.90, 0.05, 0.65, 33.5];

```

```

24. % Discount factor not used for average reward crite-
    rion
25. discountFactor = 0.99;
26.
27. % must be \in {"average","policyIteration","valueIt-
    eration","LP"}
28. solutionMethod = "average";
29.
30. tic;
31. [V, policy, averageReward] = MarkovDecisionPro-
    cess(discountFactor,processParameters,solution-
    Method);
32. timeParallel = toc;
33.
34. runTests=false;
35.
36. if runTests==true
37.     % Caution for average reward criterion!
38.     % Expected time: up to 8 hours = 300s*50*2 /
3600s/h
39.     numberOfRuns = 50;
40.
41.     % Hypo a: cheaper resin -> higher average reward
42.     % Hypo b: cheaper resin -> earlier resin ex-
change
43.     firstExchange = zeros(numberOfRuns,1);
44.     for i=1:numberOfRuns
45.         costOfResin(i) = 192960-192960/(number-
OfRuns-1)*(i-1);
46.     end
47.     policiesCostOfResin = [];
48.     averageRewardsCostOfResin = [];
49.     timingCostOfResin = [];
50.     valuesCostOfResin = [];
51.     for i=1:numberOfRuns
52.         processParameters(7) = costOfResin(i);
53.         tic;
54.         [V, policy, averageReward] = MarkovDeci-
sionProcess(discountFactor,processParameters,solu-
tionMethod);
55.         timeCOR = toc;
56.         timingCostOfResin = [timingCostOfResin;
timeCOR];
57.
58.         firstExchange(i) = minNumberOfBatchesPuri-
fied(policy);
59.
60.         valuesCostOfResin = [valuesCostOfResin;
V(1)];
61.         averageRewardsCostOfResin = [averageRe-
wardsCostOfResin; averageReward];
62.         policiesCostOfResin = [policiesCostOfResin;
policy];
63.     end
64.
65.     %Reset processParameter(7) if both test run back
to back
66.     processParameters(7) = 96480;

```

```

67.
68.     % Hypo c: steeper decay and lower minimum capac-
        ity -> higher average
69.     % reward
70.     % Hypo d: steeper decay and lower minimum capac-
        ity -> later resin
71.     % exchange
72.     for i=1:numberOfRuns
73.         minimumCapacitiesCap(i) = 1.0-1.0/(number-
OfRuns-1)*(i-1);
74.     end
75.     policiesMinCap = [];
76.     averageRewardsMinCap = [];
77.     timingMinCap = [];
78.     valuesMinCap = [];
79.     firstExchangeCap = zeros(numberOfRuns,1);
80.
81.     for i=1:numberOfRuns
82.         processParameters(12) = minimumCapacities-
Cap(i);
83.         tic;
84.         [V, policy, averageReward] = MarkovDeci-
sionProcess(discountFactor,processParameters,solu-
tionMethod);
85.         timeMinCap = toc;
86.
87.         timingMinCap = [timingMinCap; timeMinCap];
88.
89.         firstExchangeCap(i) = minNumberOfBatch-
esPurified(policy);
90.
91.         valuesMinCap = [valuesMinCap; V(1)];
92.         averageRewardsMinCap = [averageRewardsMin-
Cap; averageReward];
93.         policiesMinCap = [policiesMinCap; policy];
94.     end
95. end

1.  function [V, policiesForAllDSPStates, averageReward]
    = MarkovDecisionProcess(lambda,processParameters,sol-
utionMethod)
2.  clc
3.  disp("Two parallel USP reactors into one chromatog-
raphy column")
4.
5.  %Process parameters
6.
7.  V=processParameters(1);
8.  cb=processParameters(2);
9.  cg=processParameters(3);
10. cp=processParameters(4);
11. vp=processParameters(5);
12. % value of growth medium per L: concentration of
    1.5mg of TPA in spent
13. % growth medium times the per gram value of un-puri-
    fied TPA
14. vg=vp*1.5e-3;

```

```

15. fix=processParameters(6);
16. Cresin=processParameters(7);
17.
18. p = processParameters(8);
19.
20. pNoDecay = processParameters(9);
21. pDecayByOne = processParameters(10);
22. pDecayByTwo = processParameters(11);
23.
24. minimumCapacity = processParameters(12);
25. maxConcentration = processParameters(end);
26.
27. %sufficiently large number to disincentivize infea-
    sible actions
28. M=99999999;
29.
30. %Upstream reactor states
31. %let "empty"                                be state
    1
32. %let "ready"                                be state
    2
33. %let "growth i"                            be state
    3..3+ng
34. %let "production i"                        be state
    3+ng+1..3+ng+1+np
35. %let "upset"                                be state
    3+ng+1+np+1
36.
37. %number of growth and production states, respec-
    tively
38. ng = 3;
39. np = 12;
40.
41. e=1; r=2;
42.
43. growth = r+1:r+1+ng-1;
44. production = growth(end)+1:growth(end)+1+np-1;
45.
46. Gp = growth(3);
47.
48. u = production(end)+1;
49.
50. Susp=u;
51.
52. productionProbabilities = zeros(length(produc-
    tion),0);
53.
54. % Deterioration over the last 1/6 of the production
    phase
55. declinePhase=[0.84,0.34];
56.
57. productionProbabilities(1:length(production)-
    length(declinePhase)-1)=p;
58. productionProbabilities(length(production)-
    length(declinePhase):length(production)-1)=de-
    clinePhase(1:length(declinePhase));
59. productionProbabilities(length(production))=1;
60.

```

```

61. concentration = setConcentrations(np,maxConcentra-
    tion);
62.
63. %Downstream reactor states
64. %let "resin capacity to purify batch i"
    be state 1..ncap
65.
66. %must be > 3
67. ncap = 12;
68.
69. Sdsp=ncap;
70.
71. %USP reactor actions
72. %let "none" be ac-
    tion 1
73. %let "add growth medium" be ac-
    tion 2
74. %let "add production medium" be ac-
    tion 3
75. %let "harvest" be ac-
    tion 4
76. %let "prepare" be ac-
    tion 5
77. %let "harvest and prepare" be ac-
    tion 6
78. actionsUsp=["none","addgm","addpm","har-
    vest","prep","hprep"];
79.
80. Ausp=length(actionsUsp);
81.
82. %DSP reactor actions
83. %let "none" be ac-
    tion 1
84. %let "accept" be ac-
    tion 2
85. %let "exchange resin" be ac-
    tion 3
86. actionsDsp=["none","accept","exresin"];
87.
88. Adsp=length(actionsDsp);
89.
90. %P(current, next, action)
91. Pusp=zeros(Susp,Susp,Ausp);
92.
93. %action none = 1
94. Pusp(e,e,1)=1;
95. Pusp(r,r,1)=p;
96. Pusp(r,u,1)=1-p;
97.
98. %doing nothing during growth -> upset
99. for i=growth(1):growth(end)
100.     Pusp(i,u,1)=1;
101. end
102.
103. %doing nothing during production -> upset
104. for i=production(1):production(end)
105.     Pusp(i,u,1)=1;
106. end

```



```

107.
108. Pusp(u,u,1)=1;
109.
110. %action add growth medium = 2
111. for i=1:Susp
112.     % addgm during growth but before last growth
    state or in ready to start
113.     % growth
114.     if ismember(i,[r,growth]) && i<growth(end)
115.         Pusp(i,i+1,2)=p;
116.         Pusp(i,u,2)=1-p;
117.     else
118.         %infeasible everywhere else
119.         Pusp(i,u,2)=1;
120.     end
121. end
122.
123.
124.
125. %action add production medium = 3
126. for i=1:Susp
127.     % addpm to continue production until second to
    last production state
128.     if ismember(i,production(1):production(end)-1)
129.         %fetch probabilities from probability vector
    including
130.         %deterioration towards the end
131.         ProdStage2=i-3-ng+1;
132.         probability = productionProbabilities(Prod-
    Stage2);
133.
134.         %row sums must equal 1, state transitions
    either without problem or
135.         %transitions to upset
136.         Pusp(i,i+1,3)=probability;
137.         Pusp(i,u,3)=1-probability;
138.         % addpm to convert production-competent growth
    state into production state
139.         elseif ismember(i,Gp)
140.             Pusp(i,production(1),3)=p;
141.             Pusp(i,u,3)=1-p;
142.         else
143.             %infeasible everywhere else
144.             Pusp(i,u,3)=1;
145.         end
146.     end
147.
148. %action harvest/ dump = 4
149. Pusp(r,e,4)=1;
150.
151. % dump all growth medium to abort process
152. for i=growth(1):growth(end)
153.     Pusp(i,e,4)=1;
154. end
155.
156. % harvest production medium is only feasible if
    a(DSP)=accept and a(USPj)!=4 or 6
157. for i=production(1):production(end)

```

```

158.     Pusp(i,e,4)=1;
159. end
160.
161. % dump upset reactor's contents to abort process
162. Pusp(u,e,4)=1;
163.
164. Pusp(e,e,4)=1;
165.
166. %action prepare = 5
167. for i=1:Susp
168.     %infeasible everywhere except if reactor is
    empty
169.     if i==e
170.         Pusp(e,r,5)=p;
171.         Pusp(e,u,5)=1-p;
172.     else
173.         Pusp(i,u,5)=1;
174.     end
175. end
176.
177. %action harvest/dump and prepare = 6
178. Pusp(r,r,6)=Pusp(e,r,5);
179. Pusp(r,u,6)=Pusp(e,u,5);
180.
181. % dump all growth medium and prepare reactor to re-
    start process
182. for i=growth(1):growth(end)
183.     Pusp(i,r,6)=Pusp(e,r,5);
184.     Pusp(i,u,6)=Pusp(e,u,5);
185. end
186.
187. % harvest production medium is only feasible if
    a(DSP) = accept and prepare
188. % reactor to restart process
189. for i=production(1):production(end)
190.     %only feasible for USPi if a(DSP)=accept and
        a(USPj)!=4 or 6
191.     Pusp(i,r,6)=Pusp(e,r,5);
192.     Pusp(i,u,6)=Pusp(e,u,5);
193. end
194.
195. Pusp(u,r,6)=Pusp(e,r,5);
196. Pusp(u,u,6)=Pusp(e,u,5);
197.
198. Pusp(e,r,6)=Pusp(e,r,5);
199. Pusp(e,u,6)=Pusp(e,u,5);
200.
201.
202. %P(current, next, action)
203. Pdsp=zeros(Sdsp,Sdsp,Adsp);
204.
205. %action none = 1
206. for i = 1:ncap
207.     %feasible only if i<ncap, must exchange if ncap
        is reached
208.     Pdsp(i,i,1) = 1;
209. end
210.

```

```

211.
212. %action accept = 2
213. %resin is used once for purification and its capacity deteriorates
214. %stochastically
215. for i = 1:(ncap-1)
216.     %feasible only before resin has to be exchanged
    and if a(USP)=harvest OR harvest+prep
217.     if i==ncap-1
218.         %simplifying assumption as discussed in the-
        sis
219.         Pdsp(i,i,2) = pNoDecay;
220.         Pdsp(i,i+1,2) = pDecayByOne+pDecayByTwo;
221.     else
222.         Pdsp(i,i,2) = pNoDecay;
223.         Pdsp(i,i+1,2) = pDecayByOne;
224.         Pdsp(i,i+2,2) = pDecayByTwo;
225.     end
226. end
227. %infeasible
228. Pdsp(ncap,ncap,2) = 1;
229.
230.
231. %action exchange resin = 3
232. %resin is exchanged for a full capacity one
233. for i = 1:ncap
234.     Pdsp(i,1,3) = 1;
235. end
236.
237. disp("Combining probability matrices...");
238. PUSpTwo = constructSystemProbabilities(Pusp,Pusp);
239. tic
240. Pall = constructSystemProbabilities(PUSpTwo,Pdsp);
241. toc
242.
243. %Test for non-one row sums in transition probabilities
244. % rowSumProductUSP = 1;
245. % for i=1:Susp
246. %     rowSumProductUSP = row-
        SumProductUSP.*sum(Pusp(i,:,:));
247. % end
248. %
249. % rowSumProductDSP = 1;
250. % for i=1:Sdsp
251. %     rowSumProductDSP = row-
        SumProductDSP.*sum(Pdsp(i,:,:));
252. % end
253. %
254. % rowSumProductAll = 1;
255. % for i=1:Susp*Sdsp
256. %     rowSumProductAll = rowSumProduc-
        tAll.*sum(Pall(i,:,:));
257. % end
258.
259. %Upstream Process Reactor
260. %R(current, action)
261. %costs are negative, revenues positive

```

```

262. Rusp=zeros(Susp,Ausp);
263.
264. %action none = 1, zero reward
265. for i=1:Susp
266.     if ismember(i,growth)
267.         %opportunity cost of lost growth medium be-
cause system moves to upset state
268.         Rusp(i,1)=-vg*V;
269.     elseif ismember(i,production)
270.         %cost of lost batch
271.         Rusp(i,1)=-vp*V*concentration(i-3-ng+1);
272.     else
273.         Rusp(i,1)=0;
274.     end
275. end
276.
277. %action add growth medium = 2
278. for i=1:Susp
279.     if ~ismember(i,[r,growth(1):growth(end)-1])
280.         Rusp(i,2)=-M;
281.     else
282.         Rusp(i,2)=-cg*V-fix;
283.     end
284. end
285.
286.
287. %action add production medium = 3
288. for i=1:Susp
289.     if ~ismember(i,[Gp,production(1):produc-
tion(end)-1])
290.         % addpm is infeasible in last production
state and outside of:
291.         % production phase and production competent
growth states
292.         Rusp(i,3)=-M;
293.     else
294.         Rusp(i,3)=-cp*V-fix;
295.     end
296. end
297.
298. %action harvest/ dump = 4
299. for i=1:Susp
300.     if ismember(i,growth)
301.         %dump growth medium: capture value of spent
growth medium
302.         Rusp(i,4)=vg*V-fix;
303.     elseif ismember(i,[r,u])
304.         Rusp(i,4)=-fix;
305.     elseif ismember(i,production)
306.         %only feasible in USPi if a(DSP)=accept and
a(USPj) != 4 or 6
307.         %set rewards in system matrix
308.         Rusp(i,4)=-M;
309.     else
310.         Rusp(i,4)=-M;
311.     end
312. end
313.

```

```

314. %action prepare = 5
315. for i=1:Susp
316.     if i==e
317.         %prepare reactor with growth medium and mi-
            crocarriers
318.         Rusp(i,5)=- (cg+cb)*V-fix;
319.     else
320.         Rusp(i,5)=-M;
321.     end
322. end
323.
324. %action harvest/ dump + prepare = 6
325. for i=1:Susp
326.     if ismember(i,growth)
327.         %dump growth medium: capture value of spent
            growth medium - cost
328.         %of preparing reactor afterwards
329.         Rusp(i,6)=vg*V- (cg+cb)*V-fix;
330.     elseif ismember(i,[r,u])
331.         Rusp(i,6)=-fix;
332.     elseif ismember(i,production)
333.         %only feasible in USPi if a(DSP)=accept and
            a(DSPj)= 4 or 6
334.         %set rewards in system matrix
335.         Rusp(i,6)=-M;
336.     else
337.         Rusp(i,6)=-M;
338.     end
339. end
340.
341. %Downstream Process Reactor
342. %R(current, action)
343. Rdsp=zeros(Sdsp,Adsp);
344. %let do nothing be action 1
345. for i=1:(Sdsp-1)
346.     Rdsp(i,1)=0;
347. end
348. %infeasible action: have to exchange resin when min
            capacity is reached
349. Rdsp(Sdsp,1)=-M;
350.
351.
352. %let accept be action 2
353. %rewards are set in system reward matrix due to in-
            fluence of USP reactor
354. %state
355. for i=1:(Sdsp-1)
356.     Rdsp(i,2)=-M;
357. end
358. %infeasible action: have to exchange resin when min
            capacity is reached
359. Rdsp(Sdsp,2)=-M;
360.
361. %let exchange resin be action 3
362. for i=1:Sdsp
363.     Rdsp(i,3)=-Cresin;
364. end
365.

```

```

366. disp("Combining rewards matrices...");
367. RUsptwo = constructSystemRewards(Rusp,Rusp);
368. tic
369. Rall = constructSystemRewards(RUsptwo,Rdsp);
370. toc
371.
372. %for USP i and j
373. %    let harvest/ dump be action 4
374. %    let harvest/ dump + prep be action 6
375. %for DSP:
376. %    let accept be action 2
377. %rewards are set in system reward matrix due to in-
    fluence of USP reactor's
378. %state on DSP decision's reward
379. capacity = setCapacities(ncap,minimumCapacity);
380.
381. disp("Setting system level rewards...");
382. tic
383. % set rewards for feasible actions where state in-
    terdependencies exist;
384. % infeasibilities are already modeled above
385. for Usp1State = 1:Susp
386.     for Usp2State = 1:Susp
387.         if (ismember(Usp1State,production) &&
            ~ismember(Usp2State,production))
388.             % Case 1: USP 1 is in the production
            phase but USP 2 is not:
389.             % (4,x,2) and (6,x,2) are feasible where
            x is a feasible action
390.             % in USP2,
391.             % and the DSP is not in the last state
392.             for DspState = 1:Sdsp-1
393.                 for Usp2Action = 1:Ausp
394.                     % Position in rewards matrix
395.                     % determine row index
396.                     x=(Usp1State-1)*Susp*Sdsp +
                        (Usp2State-1)*Sdsp + DspState;
397.
398.                     % (4,x,2) is feasible
399.                     Usp1Action = 4;
400.                     DspAction = 2;
401.                     % Position in rewards matrix
402.                     % determine column index
403.                     y=(Usp1Action-1)*Ausp*Adsp +
                        (Usp2Action-1)*Adsp + DspAction;
404.
405.
406.                     % Production phase cycle of USP
1 for
407.                     % concentration of product
408.                     Usp1ProdState=Usp1State-3-ng+1;
409.
410.                     % Set reward for feasible ac-
tion:
411.                     % Total reward consists of reve-
nue from
412.                     % purified product
413.                     % - fixed costs

```



```

454.                                     Usp1ProdState=Usp1State-
      3-ng+1;
455.
456.                                     % Set reward for feasi-
      ble action:
457.                                     % Total reward consists
      of revenue from
458.                                     % purified product
459.                                     % - fixed costs
460.                                     % + rewards caused in
      USP2 in that decision epoch
461.                                     Rall(x,y)=vp*V*concen-
      tration(Usp1ProdState)*capacity(DspState)-
      fix+Rusp(Usp2State,Usp2Action);
462.
463.
464.                                     % (6,x,2) is feasible
465.                                     Usp1Action = 6;
466.                                     DspAction = 2;
467.                                     % Position in rewards
      matrix
468.                                     % determine column index
469.                                     y=(Usp1Action-
      1)*Ausp*Adsp + (Usp2Action-1)*Adsp + DspAction;
470.
471.                                     Rall(x,y)=vp*V*concen-
      tration(Usp1ProdState)*capacity(DspState)-fix-
      (cg+cb)*V-fix+Rusp(Usp2State,Usp2Action);
472.                                     % Case 2b: harvesting in USP
      1 is infeasible while
473.                                     % USP 2 is harvested
474.                                     elseif (~ismember(Usp1Ac-
      tion,[4,6]) && ismember(Usp2Action,[4,6]))
475.                                     % Position in rewards
      matrix
476.                                     % determine row index
477.                                     x=(Usp1State-
      1)*Susp*Sdsp + (Usp2State-1)*Sdsp + DspState;
478.
479.                                     % (x,4,2) is feasible
480.                                     Usp2Action = 4;
481.                                     DspAction = 2;
482.                                     % Position in rewards
      matrix
483.                                     % determine column index
484.                                     y=(Usp1Action-
      1)*Ausp*Adsp + (Usp2Action-1)*Adsp + DspAction;
485.
486.
487.                                     % Production phase cycle
      of USP 1 for
488.                                     % concentration of prod-
      uct
489.                                     Usp2ProdState=Usp2State-
      3-ng+1;
490.
491.                                     % Set reward for feasi-
      ble action:

```



```

492.                                     % Total reward consists
      of revenue from
493.                                     % purified product
494.                                     % - fixed costs
495.                                     % + rewards caused in
      USP2 in that decision epoch
496.                                     Rall(x,y)=vp*V*concen-
      tration(Us2ProdState)*capacity(DspState)-
      fix+Rusp(Us1State,Us1Action);
497.
498.
499.                                     % (x,6,2) is feasible
500.                                     Usp2Action = 6;
501.                                     DspAction = 2;
502.                                     % Position in rewards
      matrix
503.                                     % determine column index
504.                                     y=(Us1Action-
      1)*Ausp*Adsp + (Usp2Action-1)*Adsp + DspAction;
505.
506.                                     Rall(x,y)=vp*V*concen-
      tration(Us2ProdState)*capacity(DspState)-fix-
      (cg+cb)*V-fix+Rusp(Us1State,Us1Action);
507.                                     end
508.                                     end
509.                                     end
510.                                     end
511.                                     elseif (~ismember(Us1State,production) &&
      ismember(Us2State,production))
512.                                     % Case 3: USP 2 is in the production
      phase but USP 1 is not:
513.                                     % (x,4,2) and (x,6,2) are feasible where
      x is a feasible action
514.                                     % in USP1,
515.                                     % and the DSP is not in the last state
516.                                     for DspState = 1:Sdsp-1
517.                                         for Us1Action = 1:Ausp
518.                                             % Position in rewards matrix
519.                                             % determine row index
520.                                             x=(Us1State-1)*Susp*Sdsp +
      (Us2State-1)*Sdsp + DspState;
521.
522.                                             % (4,x,2) is feasible
523.                                             Usp2Action = 4;
524.                                             DspAction = 2;
525.                                             % Position in rewards matrix
526.                                             % determine column index
527.                                             y=(Us1Action-1)*Ausp*Adsp +
      (Usp2Action-1)*Adsp + DspAction;
528.
529.
530.                                     % Production phase cycle of USP
      1 for
531.                                     % concentration of product
532.                                     Usp2ProdState=Usp2State-3-ng+1;
533.
534.                                     % Set reward for feasible ac-
      tion:

```

```

535.                                     % Total reward consists of reve-
nue from
536.                                     % purified product
537.                                     % - fixed costs
538.                                     % + rewards caused in USP2 in
that decision epoch
539.                                     Rall(x,y)=vp*V*concentra-
tion(Us2ProdState)*capacity(DspState)-
fix+Rusp(Us1State,Us1Action);
540.
541.
542.                                     % (x,6,2) is feasible
543.                                     Us2Action = 6;
544.                                     DspAction = 2;
545.                                     % Position in rewards matrix
546.                                     % determine column index
547.                                     y=(Us1Action-1)*Ausp*Adsp +
(Us2Action-1)*Adsp + DspAction;
548.
549.                                     Rall(x,y)=vp*V*concentra-
tion(Us2ProdState)*capacity(DspState)-fix-
(cg+cb)*V-fix+Rusp(Us1State,Us1Action);
550.                                     end
551.                                     end
552.
553.                                     end
554.                                     end
555. end
556. toc
557.
558.
559.
560. %Check model
561. mdp_check(Pall,Rall);
562.
563. %Set discount rate
564. discount = lambda;
565.
566. averageReward = 0;
567.
568.
569. disp("Solving MDP...");
570. tic
571. %Solve MDP
572. if solutionMethod=="average"
573.     [policy, average_reward] = mdp_rela-
tive_value_iteration(Pall,Rall,0.1,10);
574.     averageReward = average_reward;
575. elseif solutionMethod=="policyIteration"
576.     [V, policy] = mdp_policy_iteration(Pall, Rall,
discount);
577. elseif solutionMethod=="valueIteration"
578.     [policy] = mdp_value_iteration(Pall, Rall, dis-
count);
579. elseif solutionMethod=="LP"
580.     [V, policy] = mdp_LP(Pall, Rall, discount);
581. else

```

```

582.     disp("Error: No viable solution method de-
        fined!");
583. end
584. toc
585.
586. % Generate a readable policy table for both USP re-
        actors for one given DSP
587. % state
588.
589. policiesForAllDSPStates = [];
590. %Make policy legible
591. readableSolution = constructReadableSolution(policy);
592. for DspStateForPolicy = 1:Sdsp
593.     policyMatrixReadable = constructPolicyMa-
        trix(readableSolution,Susp,Sdsp,DspStateForPolicy);
594.     policiesForAllDSPStates = [poli-
        ciesForAllDSPStates; policyMatrixReadable];
595. end
596. disp("Done! Have a look at the readable policy ma-
        trix.")

1.  function readableSolution = constructReadableSol-
        ution(policy)
2.
3.  Ausp = 6;
4.  Adsp = 3;
5.
6.  lookupTable = strings(1,Ausp*Ausp*Adsp);
7.
8.  %Construct lookup table
9.  for Usp1Action = 1:Ausp
10.     for Usp2Action = 1:Ausp
11.         for DspAction = 1:Adsp
12.             indexNumber = (Usp1Action-1)*Ausp*Adsp +
                (Usp2Action-1)*Adsp + DspAction;
13.             lookupTable(indexNumber) =
                strcat(string(Usp1Action),string(Usp2Ac-
                tion),string(DspAction));
14.         end
15.     end
16. end
17.
18. readableSolution = [];
19.
20. for index=1:length(policy)
21.     readableSolution = [readableSolution; lookupTa-
        ble(policy(index))];
22. end

1.  function outputMatrix = constructPolicyMatrix(pol-
        icity,Susp,Sdsp,DspState)
2.
3.  outputMatrix = strings(Susp,Susp);
4.  if DspState == 0
5.     % Set to 0 to indicate that policy for only one
        reactor is relevant
6.     for Usp1State = 1:Susp

```

```

7.         for DspState = 1:Sdsp
8.             Usp2State = 1;
9.             policyIndex = (Usp1State-1)*Susp*Sdsp +
(Usp2State-1)*Sdsp + DspState;
10.          outputMatrix(Usp1State,DspState) = pol-
icy(policyIndex);
11.        end
12.    end
13. else
14.     for Usp1State = 1:Susp
15.         for Usp2State = 1:Susp
16.             policyIndex = (Usp1State-1)*Susp*Sdsp +
(Usp2State-1)*Sdsp + DspState;
17.             outputMatrix(Usp1State,Usp2State) = pol-
icy(policyIndex);
18.         end
19.     end
20. End

1.  function MinimumNumberOfBatchesBeforeExchange = min-
NumberOfBatchesPurified(policy)
2.
3.  % Takes a matrix of dimensions 12*18 x 18 holding
the optimal policies for
4.  % the 2:1 case study for all 12 DSP states after
each other
5.  % Outputs the number of batches which were purified
before the a
6.  % resin exchange is prescribed for any USP states
7.
8.
9.  %Only find strings for which DSP action = 3
10. allExchanges = find(~cellfun(@isempty,regexp(pol-
icy,'^[0-9]+\s?3$')));
11.
12.  % find row index of all entries
13. DSPStateDuringExchange = mod(allExchanges,18*12); %
for final DSP state always zero
14.
15. DSPStateDuringExchange = DSPStateDuringExchange ./
18;
16.
17. % sort ascendingly
18. DSPStateDuringExchangeSorted = sort(DSPStateDur-
ingExchange);
19.
20. % first entries may be zero, because row index of
final DSP
21. % state (ncap) is zero: 12 mod 18*12 = 0
22. % if only zeros exist, resin exchange takes place in
ncap-1
23. i = 1;
24. while DSPStateDuringExchangeSorted(i)==0
25.     if i == length(DSPStateDuringExchangeSorted)
26.         % step out if there are only zeros in the
array ->
27.         break;
28.     end

```

```
29.         i=i+1;
30.     end
31.
32.     % Number of batches purified before resin exchange
    is the index of the DSP
33.     % state in which exchange takes place
34.     MinimumNumberOfBatchesBeforeExchange =
        floor(DSPStateDuringExchangeSorted(i));

1.     function reward =
        getReward(Rall,Usp1State,Usp2State,DspState,Usp1Ac-
        tion,Usp2Action,DspAction,Susp,Sdsp,Ausp,Adsp)
2.
3.         x=(Usp1State-1)*Susp*Sdsp + (Usp2State-1)*Sdsp +
        DspState;
4.         y=(Usp1Action-1)*Ausp*Adsp + (Usp2Action-1)*Adsp
        + DspAction;
5.         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	21											23
g ₁												
g ₂												
g ₃												
g ₄												
g ₅												
g ₆	31											33
g ₇												
g ₈												
p ₁												
p ₂												
...												
p ₂₈												
p ₂₉												
p ₃₀	62											
p ₃₁												
...												
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												
g₁	21											23
g₂												
g₃												
...	31											
p₈												
p₉								33				
p₁₀												
p₁₁	62											
p₁₂												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 3	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: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											
1:1			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			0.9975				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: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	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	127.70812
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	121.95555
2:1	102386.9388						-28677.36368			9	122.7595
2:1	98448.97959						-25562.0355			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						688.313911			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-D	Cost of resin exchange	Minimum capacity allowed	Probability of	Probability of	Probability of	Probability of	Average reward	Total disc. Rewd	Lateness of first exchange	Figure 3	Time to
		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		0.877555102					-37494.46301		9		123.68914
2:1		0.857142857					-36535.60901		9		120.61067
2:1		0.836734694					-35666.72681		9		119.0351
2:1		0.816326531					-34797.86371		9		136.96901
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					-27594.34241		9		128.49948
2:1		0.673469389					-25932.74755		9		119.57386
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		119.61141
2:1		0.510204082					-13337.87285		8		121.52442
2:1		0.489795918					-11769.27015		8		118.93197
2:1		0.469387756					-10219.56302		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					-2246.206887		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		179.96169
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		0.091632653					294.2290652		5		118.83380
2:1		0.061232449					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		120.15545

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

	100%	100%	100%	100%	95%	90%	85%	80%	75%	70%	65%	0%
e	51											53
r	21											23
g ₁												
...												
g ₅												
g ₆	31											
g ₇												
g ₈												
p ₁												
...												
p ₂₁												
p ₂₂												
p ₂₃												
...												
p ₂₈												
p ₂₉	62											33
p ₃₀												
p ₃₁												
...												
p ₃₆												13
u	61											63

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

	100%	100%	100%	100%	95%	90%	85%	80%	75%	70%	65%	0%
e		51										53
r		21										23
g₁												
...												
g₅												
g₆		31										33
g₇												
g₈												
p₁												
...												
p₃₅												
p₃₆		62										13
u		61										63

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

	e	r	g₁	g₂	g₃	p₁	...	p₈	p₉	p₁₀	p₁₁	p₁₂	u
e	551	521	531						562				561
r			231						262				261
g₁	251	221											
g₂													
g₃													
p₁			331						362				361
...	351	321											
p₈													
p₉													
p₁₀			632						Continue production in reactor one, harvest reactor two				662
p₁₁	652	622											
p₁₂													
u	651	621	631						662				661

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	251	221	231				262				261		
g ₁													
g ₂													
g ₃													
p ₁	351	321	331				362				361		
...													
p ₈													
p ₉													
p ₁₀	652	622	632				662				662		
p ₁₁													
p ₁₂													
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	251	221	231										262		261	
g ₁																
g ₂																
g ₃	351	321	331				331				362		361			
p ₁																
p ₂																
p ₃																
...																
p ₇																
p ₈																
p ₉	652	622	333										Continue production, exchange resin		662	
p ₁₀																
p ₁₁																
p ₁₂																
u	651	621	631										662		661	

Appendix 22: Average reward maximizing policy for the operation of two parallel bioreactors 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			361					331									
p ₁																	
p ₂																	
p ₃	351																
p ₄																	
...				321									333		362		361
p ₇																	
p ₈																	
p ₉																	
p ₁₀	353		323														363
p ₁₁															331	362	
p ₁₂	652		622					632								162	662
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