DYNAMIC PROGRAMMING

Dynamic Programming

- It is a useful mathematical technique for making a sequence of interrelated decisions.
- Systematic procedure for determining the optimal combination of decisions.
- There is no standard mathematical formulation of “the” Dynamic Programming problem.
- Knowing when to apply dynamic programming depends largely on experience with its general structure.

Prototype example

- Stagecoach problem
  - Fortune seeker wants to go from Missouri (A) to California (J) in the mid-19th century.
  - Journey has 4 stages.
  - Cost is the life insurance of a specific route; lowest cost is equivalent to safest trip.

Costs

- Cost $c_{ij}$ of going from state $i$ to state $j$ is:

<table>
<thead>
<tr>
<th></th>
<th>B</th>
<th>C</th>
<th>D</th>
<th></th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
<th>G</th>
<th>H</th>
<th>I</th>
<th>J</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>2</td>
<td>4</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>7</td>
<td>4</td>
<td>6</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>3</td>
<td>2</td>
<td>4</td>
<td>B</td>
<td>7</td>
<td>4</td>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D</td>
<td>4</td>
<td>1</td>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>6</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- Problem: which route minimizes the total cost of the policy?

Solving the problem

- Note that greedy approach does not work.
  - Solution $A \rightarrow B \rightarrow F \rightarrow I \rightarrow J$ has total cost of 33.
  - However, e.g. $A \rightarrow D \rightarrow F$ is cheaper than $A \rightarrow B \rightarrow F$.
- Other possibility: trial-and-error. Too much effort even for this simple problem.
- Dynamic programming is much more efficient than exhaustive enumeration, especially for large problems.
- Starts from the last stage of the problem, and enlarges it one stage at a time.

Formulation

- Decision variables $x_n$ ($n = 1, 2, 3, 4$) are the immediate destination of stage $n$.
  - Route is $A \rightarrow x_1 \rightarrow x_2 \rightarrow x_3 \rightarrow x_4$, where $x_4 = J$.
- Total cost of the best overall policy for the remaining stages is $f_n(s, x_n)$.
  - Actual state is $s$, ready to start stage $n$, selecting $x_n$ as the immediate destination.
- $x_n^*$ minimizes $f_n(s, x_n)$ and $f_n^*(s, x_n)$ is the minimum value of $f_n(s, x_n)$:
  $$f_n^*(s) = \min_{x_n} f_n(s, x_n) = f_n(s, x_n^*)$$
Formulation

- $f_i^*(s, x_i) = \text{immediate cost (stage n)} + \text{minimum future cost (stages n+1 onward)}$
  
- $c_{sx} + f^*_n(x_n)$

- **Objective:** find $f^*_n(A)$ and the corresponding route.
  - Dynamic programming finds successively $f^*_n(s), f^*_n(s), f^*_n(s)$
  - and finally $f^*_n(A)$.

Stage $n = 3$

- Needs a few calculations. If fortune seeker is in state $F$, he can go to either $H$ or $I$ with costs $c_{FH} = 6$ or $c_{FI} = 3$.
- Choosing $H$, the minimum additional cost is $f^*_n(H) = 3$.
- Total cost is $6 + 3 = 9$.
- Choosing $I$, the total cost is $3 + 4 = 7$.
  - This is smaller, and it is the optimal choice for state $F$.

Stage $n = 2$

- In this case, $f^*_n(s, x_j) = c_{sx} + f^*_n(x_j)$.

- Example for node C:
  - $x_2 = E$: $f_1^*(C, E) = c_{CE} + f_2^*(E) = 3 + 4 = 7$ -- optimal
  - $x_3 = F$: $f_1^*(C, F) = c_{CF} + f_2^*(F) = 2 + 7 = 9$.
  - $x_6 = G$: $f_1^*(C, G) = c_{CG} + f_2^*(G) = 4 + 6 = 10$.

Solution procedure

- When $n = 4$, the route is determined by its current state $s$ ($H$ or $J$) and its final destination $J$.
- Since $f^*_n(s) = f^*_n(s, J) = c_{sJ}$, the solution for $n = 4$:

<table>
<thead>
<tr>
<th>$s$</th>
<th>$f^*_n(s)$</th>
<th>$x_n^*$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$H$</td>
<td>3</td>
<td>$J$</td>
</tr>
<tr>
<td>$I$</td>
<td>4</td>
<td>$J$</td>
</tr>
</tbody>
</table>

Stage $n = 3$

- Similar calculations can be made for the two possible states $s = E$ and $s = G$, resulting in the table for $n = 3$:

<table>
<thead>
<tr>
<th>$s$</th>
<th>$f^*_n(s)$</th>
<th>$x_n^*$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$H$</td>
<td>6</td>
<td>$I$</td>
</tr>
<tr>
<td>$I$</td>
<td>6</td>
<td>$J$</td>
</tr>
</tbody>
</table>

Stage $n = 2$

- Similar calculations can be made for the two possible states $s = B$ and $s = D$, resulting in the table for $n = 2$:

<table>
<thead>
<tr>
<th>$s$</th>
<th>$f^*_n(s)$</th>
<th>$x_n^*$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$B$</td>
<td>11</td>
<td>$E$ or $F$</td>
</tr>
<tr>
<td>$C$</td>
<td>11</td>
<td>$E$ or $F$</td>
</tr>
<tr>
<td>$D$</td>
<td>11</td>
<td>$E$ or $F$</td>
</tr>
</tbody>
</table>
Stage \( n = 1 \)

- Just one possible starting state: \( A \).
  - \( x_0 = B \): \( f_1^s(A, B) = c_{A,B} + f_1^s(B) = 2 + 11 = 13 \).
  - \( x_0 = C \): \( f_1^s(A, C) = c_{A,C} + f_1^s(C) = 4 + 7 = 11 \) ← optimal
  - \( x_0 = D \): \( f_1^s(A, D) = c_{A,D} + f_1^s(D) = 3 + 8 = 11 \) ← optimal

Results in the table:

<table>
<thead>
<tr>
<th>( s )</th>
<th>( f_1^s(x_0, x_1) = c_{s,x_1} + f_1^s(x_1) )</th>
<th>( x_1^* )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( A )</td>
<td>13</td>
<td>11</td>
</tr>
<tr>
<td>( B )</td>
<td>11</td>
<td>11</td>
</tr>
<tr>
<td>( C )</td>
<td>11</td>
<td>11</td>
</tr>
<tr>
<td>( D )</td>
<td>( C ) or ( D )</td>
<td></td>
</tr>
</tbody>
</table>

Optimal solution

- Three optimal solutions, all with \( f_1^s(A) = 11 \):

Characteristics of DP

1. The problem can be divided into stages, with a policy decision required at each stage.
   - Example: 4 stages and life insurance policy to choose.
   - Dynamic programming problems require making a sequence of interrelated decisions.
2. Each stage has a number of states associated with the beginning of each stage.
   - Example: states are the possible territories where the fortune seeker could be located.
   - States are possible conditions in which the system might be.

Characteristics of DP

3. Policy decision transforms the current state to a state associated with the beginning of the next stage.
   - Example: fortune seeker’s decision led him from his current state to the next state on his journey.
   - DP problems can be interpreted in terms of networks: each node correspond to a state.
   - Value assigned to each link is the immediate contribution to the objective function from making that policy decision.
   - In most cases, objective corresponds to finding the shortest or the longest path.

Characteristics of DP

4. The solution procedure finds an optimal policy for the overall problem. Finds a prescription of the optimal policy decision at each stage for each of the possible states.
   - Example: solution procedure constructed a table for each stage, \( n \), that prescribed the optimal decision, \( x_1^* \), for each possible state \( s \).
   - In addition to identifying optimal solutions, DP provides a policy prescription of what to do under every possible circumstance (why a decision is called policy decision). This is useful for sensitivity analysis.

Characteristics of DP

5. Given the current state, an optimal policy for the remaining stages is independent of the policy decisions adopted in previous stages.
   - Optimal immediate decision depends only on current state and not on how it was obtained: this is the principle of optimality for DP.
   - Example: at any state, the insurance policy is independent on how the fortune seeker got there.
   - Knowledge of the current state conveys all information necessary for determining the optimal policy henceforth (Markovian property). Problems lacking this property are not Dynamic Programming Problems.
### Characteristics of DP

6. Solution procedure begins by **finding the optimal policy for the last stage**. Solution is usually trivial.

7. A **recursive relationship** that identifies optimal policy for stage \( n \), given optimal policy for stage \( n+1 \), is available.
   - **Example:** recursive relationship was
     \[
     f'_n(t) = \min \{ c_{nt} + f'_{n+1}(x_{n+1}) \}
     \]
     - Recursive relationship differs somewhat among dynamic programming problems.

### Characteristics of DP

7. (cont.) Recursive relationship:
   - \( f'_n(t) = \max\{f_n(t, x_n), f'_n(t, x'_n)\} \) or \( f'_n(t) = \min\{f_n(t, x_n), f'_n(t, x'_n)\} \)
   - where \( f_n(t, x_n) \) is written in terms of \( s_n, x_n, f'_{n+1}(s_{n+1}) \), and probably some measure of the immediate contribution of \( x_n \) to the objective function.

8. Using recursive relationship, solution procedure starts at the end and moves **backward** stage by stage.
   - Stops when optimal policy starting at initial stage is found.
   - The optimal policy for the entire problem is found.
   - **Example:** the tables for the stages show this procedure.

### Deterministic dynamic programming

**Deterministic problems:** the state at the next stage is completely determined by the state and policy decision at the current stage.

- **Form of the objective function:** minimize or maximize the sum, product, etc. of the contributions from the individual stages.
- **Set of states:** may be discrete or continuous, or a state vector. Decision variables can also be discrete or continuous.

### Example: distributing medical teams

**The World Health Council has five medical teams to allocate to three underdeveloped countries.**

**Measure of performance:** additional person-years of life, i.e., increased life expectancy (in years) times country's population.

<table>
<thead>
<tr>
<th>Medical teams</th>
<th>Thousands of additional person-years of life</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>40</td>
</tr>
<tr>
<td>2</td>
<td>45</td>
</tr>
<tr>
<td>3</td>
<td>50</td>
</tr>
<tr>
<td>4</td>
<td>105</td>
</tr>
<tr>
<td>5</td>
<td>120</td>
</tr>
</tbody>
</table>
Formulation of the problem

- Problem requires three **interrelated decisions**: how many teams to allocate to the three countries (stages).
- \( x_n \) is the number of teams to allocate to stage \( n \).
- What are the states? What changes from one stage to another?
- \( s_n \) = number of medical teams still available for remaining countries (\( n, \ldots, 3 \)).
- Thus: \( s_1 = 5 \), \( s_2 = 5 - x_1 = s_1 - x_1 \), \( s_3 = s_2 - x_2 \).

States to be considered

<table>
<thead>
<tr>
<th>Medical teams</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>45</td>
<td>20</td>
<td>50</td>
<td>50</td>
</tr>
<tr>
<td>2</td>
<td>70</td>
<td>45</td>
<td>70</td>
<td>70</td>
</tr>
<tr>
<td>3</td>
<td>90</td>
<td>75</td>
<td>80</td>
<td>80</td>
</tr>
<tr>
<td>4</td>
<td>105</td>
<td>110</td>
<td>120</td>
<td>120</td>
</tr>
<tr>
<td>5</td>
<td>120</td>
<td>120</td>
<td>120</td>
<td>120</td>
</tr>
</tbody>
</table>

Overall problem

- \( p(x) \): **measure of performance** from allocating \( x_i \) medical teams to country \( i \).
- Maximize \( \sum_{i=1}^{3} p(x_i) \),
- subject to \( \sum_{i=1}^{3} x_i = s_n \),
- and \( x_i \) are nonnegative integers.

Policy

- Recursive relationship relating functions:
  \[ f_n(s_n) = \max_{x_n = 0, 1, \ldots, s_n} \left\{ p_n(x_n) + f_{n+1}(s_n - x_n) \right\} \]
  for \( n = 1, 2 \)

Solution procedure, stage \( n = 3 \)

- For last stage \( n = 3 \), values of \( p(x_i) \) are the last column of table. Here, \( x_1^* = s_1 \) and \( f_1^*(s_1) = p_1(s_1) \).

Stage \( n = 2 \)

- Here, finding \( x_1^* \) requires calculating \( f_1(s_1, x_1) \) for the values of \( x_1 = 0, 1, \ldots, s_1 \). Example for \( s_1 = 2 \):
Stage $n = 2$

- Similar calculations can be made for the other values of $s_n$:

<table>
<thead>
<tr>
<th>$n = 2$</th>
<th>$x_k$</th>
<th>$f(x_k)$</th>
<th>$p(x_k)$</th>
<th>$f(x_k) - p(x_k)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>50</td>
<td>20</td>
<td>0</td>
<td>20</td>
</tr>
<tr>
<td>2</td>
<td>70</td>
<td>45</td>
<td>0</td>
<td>45</td>
</tr>
<tr>
<td>3</td>
<td>80</td>
<td>95</td>
<td>5</td>
<td>90</td>
</tr>
<tr>
<td>4</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>150</td>
<td>125</td>
<td>120</td>
<td>5</td>
</tr>
</tbody>
</table>

Stage $n = 1$

- Only state is the starting state $s_1 = 5$:

<table>
<thead>
<tr>
<th>$n = 1$</th>
<th>$x_k$</th>
<th>$f(x_k)$</th>
<th>$p(x_k)$</th>
<th>$f(x_k) - p(x_k)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>50</td>
<td>20</td>
<td>0</td>
<td>20</td>
</tr>
<tr>
<td>2</td>
<td>70</td>
<td>45</td>
<td>0</td>
<td>45</td>
</tr>
<tr>
<td>3</td>
<td>80</td>
<td>95</td>
<td>5</td>
<td>90</td>
</tr>
<tr>
<td>4</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>150</td>
<td>125</td>
<td>120</td>
<td>5</td>
</tr>
</tbody>
</table>

Optimal policy decision

- Distribution of effort problem

- One kind of resource is allocated to a number of activities. Objective: how to distribute the effort (resource) among the activities most effectively.
- DP involves only one (or few) resources, while LP can deal with thousands of resources.
- The assumptions of LP: proportionality, divisibility and certainty can be violated by DP. Only additivity (or analogous for product of terms) is necessary because of the principle of optimality.
- World Health Council problem violates proportionality and divisibility (WHY?)

Formulation of distribution of effort

Stage $n = \text{activity } n (n = 1, 2, \ldots, N)$.
- $x_n = \text{amount of resource allocated to activity } n$.
- State $s_n = \text{amount of resource still available for allocation to remaining activities (} n, \ldots, N \text{)}$.

- When system starts at stage $n$ in state $s_n$, choice $x_n$ results in the next state at stage $n + 1$ being $s_{n+1} = s_n - x_n$:

Example

- Distributing scientists to research teams
  - 3 teams are solving engineering problem to safely fly people to Mars.
  - 2 extra scientists reduce the probability of failure.
Continuous dynamic programming

- Previous examples had a discrete state variable $s_n$ at each stage.
- They all have been reversible; the solution procedure could have moved backward or forward stage by stage.
- Next example is continuous. As $s_n$ can take any values in certain intervals, the solutions $f_n^*(-s_n)$ and $x_n^*$ must be expressed as functions of $s_n$.
- Stages in the next example will correspond to time periods, so the solution must proceed backwards.

Example: scheduling jobs

- The company Local Job Shop needs to schedule employment jobs due to seasonal fluctuations.
  - Machine operators are difficult to hire and costly to train.
  - Peak season payroll should not be maintained afterwards.
  - Overtime work on a regular basis should be avoided.
- Minimum requirements in near future:

<table>
<thead>
<tr>
<th>Season</th>
<th>Spring</th>
<th>Summer</th>
<th>Autumn</th>
<th>Winter</th>
<th>Spring</th>
</tr>
</thead>
<tbody>
<tr>
<td>Requirements</td>
<td>255</td>
<td>240</td>
<td>240</td>
<td>200</td>
<td>255</td>
</tr>
</tbody>
</table>

Example: scheduling jobs

- Employment above level in the table costs $2,000 per person per season.
- Total cost of changing level of employment from one season to the other is $200 times the square of the difference in employment levels.
- Fractional levels are possible due to part-time employees.

Formulation

- From data, maximum employment should be 255 (spring). It is necessary to find the level of employment for other seasons. Seasons are stages.
- One cycle of four seasons, where stage 1 is summer and stage 4 is spring (known employment).
- $x_n$ = employment level for stage n (n = 1, 2, 3, 4); $x_n$ = 255
- $r_n$ = minimum employment requirement for stage n: $r_1$ = 220, $r_2$ = 240, $r_3$ = 200, $r_4$ = 255. Thus: $r_n \leq x_n \leq 255$

Data

Choose $x_1$, $x_2$, and $x_3$ as to minimize $\sum_{i=1}^{3} \left( 2000(x_i - x_{i-1})^2 + 200(x_i - r_i)^2 \right)$, subject to $r_i \leq x_i \leq 255$, for $i = 1, 2, 3, 4$.

<table>
<thead>
<tr>
<th>n</th>
<th>$x_n$</th>
<th>Minimum $r_n$</th>
<th>Maximum $x_n$</th>
<th>Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>200</td>
<td>220</td>
<td>255</td>
<td>2000($x_1 - r_1$) + 200($x_1 - 255$)</td>
</tr>
<tr>
<td>2</td>
<td>220</td>
<td>240</td>
<td>255</td>
<td>2000($x_2 - r_2$) + 200($x_2 - 255$)</td>
</tr>
<tr>
<td>3</td>
<td>240</td>
<td>240</td>
<td>255</td>
<td>2000($x_3 - r_3$) + 200($x_3 - 255$)</td>
</tr>
<tr>
<td>4</td>
<td>255</td>
<td>255</td>
<td>255</td>
<td>200($x_4 - r_4$)</td>
</tr>
</tbody>
</table>
### Formulation

**Recursive relationship:**

\[
f_3^*(x_i) = \min_{s_i} \left\{ 200(x_i - s_i)^2 + 2000(x_i - r_i) + f_{i+1}^*(x_{i+1}) \right\}
\]

**Basic structure of the problem:**

<table>
<thead>
<tr>
<th>Stage</th>
<th>( s_i )</th>
<th>( f_3^*(s_i) )</th>
<th>( x_i^* )</th>
</tr>
</thead>
</table>
| Stage 1 | \( s_1 \) | \( f_1^*(s_1) \) | \( x_1^* \)
| Stage 2 | \( s_2 \) | \( f_2^*(s_2) \) | \( x_2^* \)
| Stage 3 | \( s_3 \) | \( f_3^*(s_3) \) | \( x_3^* \)
| Stage 4 | \( s_4 \) | \( f_4^*(s_4) \) | \( x_4^* \)

### Solution procedure

**Stage 4:** the solution is known to be \( x_4^* = 255 \).

\[
x_4^* = 255
\]

**Stage 3:** \( 240 \leq s_4 \leq 255 \):

\[
f_3^*(s_i) = \min_{x_i} \left\{ 200(x_i - s_i)^2 + 2000(x_i - 200) + f_4^*(s_4) \right\}
\]

\[
= \min_{x_i} \left\{ 200(x_i - s_i)^2 + 2000(x_i - 200) + 200(255 - x_i)^2 \right\}
\]

**Stage 2:**

Solved in a similar fashion, with

\[
f_i(s_i, x_i) = 200(x_i - s_i)^2 + 2000(x_i - r_i) + f_i^*(x_i)
\]

\[
= 200(x_i - s_i)^2 + 2000(x_i - 240) + 50(250 - x_i)^2 + 50(260 - x_i)^2 + 1000(x_i - 150)
\]

for \( 220 \leq s_i \leq 255 \) (possible values) and \( 240 \leq x_i \leq 255 \) (feasible values).

**Stage 1:**

Solving \( \frac{\partial f_1}{\partial x_1}(s_1, x_1) = 0 \), yields:

\[
x_1 = \frac{25 + 240}{3}
\]
Stage 2

- The solution has to be feasible for $220 \leq s_1 \leq 255$ (i.e., $240 \leq x_1 \leq 255$ for $220 \leq s_1 \leq 255$).

  $x_1^* = \frac{25.5 + 240}{3}$ only feasible for $240 \leq s_1 \leq 255$.

- Need to solve for feasible value of $x_1$ that minimizes $f_1(s_1, x_1)$ when $220 \leq s_1 < 240$.

- For $s_1 < 240$, $\frac{\partial f_1(s_1, x_1)}{\partial x_1} > 0$ for $240 \leq x_1 \leq 255$.

  So $x_1^* = 240$.

Stage 2 and Stage 1

<table>
<thead>
<tr>
<th>$s_1$</th>
<th>$f_1^*(s_1)$</th>
<th>$x_1^*$</th>
</tr>
</thead>
<tbody>
<tr>
<td>220</td>
<td>200(240−s_1)^2+12500</td>
<td>240</td>
</tr>
<tr>
<td>240</td>
<td>200/[(240−s_1)^2+(255−s_1)^2]−1000(255−s_1^2)</td>
<td>(25.5+240)/3</td>
</tr>
</tbody>
</table>

Stage 1: procedure is similar.

- Solution: $s_1^* = 247.5$, $x_1^* = 245$, $x_2^* = 247.5$, $x_4^* = 255$.

- Total cost of $185,000.

Deterministic continuous problem

- Consider the following nonlinear programming problem:

  Maximize $Z = x_1^2 + x_2$, subject to $x_1^2 + x_2 \leq 2$.

  (There are no nonnegativity constraints.)

Use dynamic programming to solve this problem.

Probabilistic dynamic programming

- State at next stage is not completely determined by state and policy decision at current stage.

- There is a probability distribution for determining the next state, see figure.

  - $S$ = number of possible states at stage $n + 1$.
  - system goes to $i$ with probability $p_i$ given state $s_n$ and decision $x_n$ at stage $n$.
  - $C_i$ = contribution of stage $n$ to objective function.

- If figure is expanded to all possible states and decisions at all stages, it is a decision tree.

Basic structure

- Relation between $f_1(s_n, x_n)$ and $f_{n+1}(s_{n+1})$ depends upon form of overall objective function.

**Example:** minimize the expected sum of the contributions from individual stages.

- $f_1(s_n, x_n)$ is the minimum expected sum from stage $n$ onward, given state $s_n$ and policy decision $x_n$ at stage $n$:

  $f_1(s_n, x_n) = \sum_{i=1}^{S} \rho_i [C_i + f_{n+1}(i)]$

  with $f_{n+1}(i) = \min_{x_n} f_1(i, x_n)$.
Example: determining reject allowances

- The Hit-and-Miss Manufacturing Company received an order to supply 1 item of a particular type.
  - Customer requires specified stringent quality requirements.
  - Manufacturer has to produce more than one to achieve one acceptable. Number of extra items is the reject allowance.
  - Probability of acceptable or defective is $\frac{1}{2}$.
  - Number of acceptable items in a lot of size $L$ has a binomial distribution probability of not acceptable items is $(1/2)^L$.
- Setup cost is $300, cost per item is $100. Maximum production runs = 3. Cost of no acceptable item after 3 runs $= 3 \times 600$.

Formulation

- Objective: determine policy regarding lot size $(1 +$ reject allowance) for required production run(s) that minimizes total expected cost.
- Stage $n =$ production run $n$ ($n = 1, 2, 3$), $x_n =$ lot size for stage $n$.
- State $s_n =$ number of acceptable items still needed ($1$ or $0$) at the beginning of stage $n$.
  - At stage $1$, state $s_1 = 1$.

Basic structure of the problem

- Recursive relationship:
  - $f_1^*(1) = \min\{K(x_1) + x_1 \text{ or } 0.5 f_2^*(1)\}$
  - for $n = 1, 2, 3$

Solution procedure

<table>
<thead>
<tr>
<th>$n = 1$</th>
<th>$x_1$</th>
<th>$x_1$</th>
<th>$f_1(x_1) = K(x_1) + x_1 + 1/2 f_2(x_1)$</th>
<th>$f_1^{opt}$</th>
<th>$x_1^{opt}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>10</td>
<td>12</td>
<td>9</td>
<td>8</td>
<td>8.5</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$n = 2$</th>
<th>$x_2$</th>
<th>$f_2(x_2) = K(x_2) + x_2 + 1/2 f_3(x_2)$</th>
<th>$f_2^{opt}$</th>
<th>$x_2^{opt}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>7</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$n = 3$</th>
<th>$x_3$</th>
<th>$f_3(x_3) = K(x_3) + x_3 + 1/2 f_4(x_3)$</th>
<th>$f_3^{opt}$</th>
<th>$x_3^{opt}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>7</td>
<td>7.5</td>
<td>6.75</td>
</tr>
</tbody>
</table>

Probability problem

- An enterprising young statistician believes that she has developed a system for winning a popular Las Vegas game. Her colleagues do not believe that her system works, so they have made a large bet with her that if she starts with three chips, she will not have at least five chips after three plays of the game. Each play of the game involves betting any desired number of available chips and then either winning or losing this number of chips. The statistician believes that her system will give her a probability of $2/3$ of winning a given play of the game.
- Assuming the statistician is correct, use dynamic programming to determine her optimal policy regarding how many chips to bet (if any) at each of the three plays of the game. The decision at each play should take into account the results of earlier plays. The objective is to maximize the probability of winning her bet with her colleagues.