ECE 573 – Power System Operations and Control

15. Dynamic – Programming – Based Unit Commitment Solution Schemes

George Gross

Department of Electrical and Computer Engineering

University of Illinois at Urbana-Champaign
UNIT COMMITMENT SOLUTION METHODS

- Heuristics
- Dynamic programming
- Mixed integer nonlinear programming
- Lagrangian relaxation
HEURISTIC APPROACHES

- Heuristic approaches use simple, practical schemes to provide priority-list-based methods to construct UC schedules
  - Priority lists give the order to commit and decommit units
  - Some, but not all, of the constraints of the generation system, are taken into account
HEURISTIC APPROACHES

- Typical approaches are based on the order of units according to increasing $AFLC$ values, where

\[
AFLC = \text{average full load costs} = \frac{\text{costs at full load}}{\text{full load}}
\]

- The units are arranged in the order

\[
AFLC_1 \leq AFLC_2 \leq \ldots \leq AFLC_n
\]
HEURISTIC APPROACHES

- In the competitive environment, costs are private information and such approaches are based on prices as costs need not be known.

- In the relations above, we replace costs everywhere by prices and the units are in order of increasing average full load prices.
BASIC LOGIC OF A PRIORITY – LIST – BASED SCHEME

start

shortage of capacity?
  yes
  select the next lowest and available AFLC unit
  if unit minimum uptime constraint is satisfied when the load decreases, start up the selected unit
  no

excess of capacity?
  yes
  select the next highest AFLC unit committed
  if unit minimum downtime constraint is satisfied when the load increases, shut down the selected unit
  no

stop
THE BASIC CONCEPT: UNIT 6 OF THE COMMITMENT LIST

load demand

committed capacity schedule

capacity of unit 6

daily load forecast

interval length must exceed $T^d_6$

testing for shutdown of unit 6

hour of the day

2 a.m.  8 a.m.
ADVANTAGES OF HEURISTIC APPROACHES

- Are practical tools to construct schedules
- Are computationally fast
- Allow the incorporation of minimum up/minimum down time constraints
- Require few calculations and so the large-scale nature of the UC problem is not an issue
BUT, HEURISTIC APPROACHES . . .

- Cannot provide optimal solutions
- Cannot determine sensitivity information
- Cannot accommodate the consideration of many constraints that may entail significant impacts
- Are rather inflexible
DYNAMIC PROGRAMMING OR

*DP* APPROACH

- *DP* is a highly **useful tool** to make a sequence of interrelated decisions: for a multi-stage problem, *DP* separates the decision into the constituent stages of the problem and then recombinest them to compute the solution.

- *DP* is a **systematic approach** to sequential decision making with the decisions optimized at
DYNAMIC PROGRAMMING OR $DP$ APPROACH

each stage rather than all at once for the entire multi-stage system: this is a salient computational aspect of the approach

- The critical requirement is the separability of the problem into non-overlapping stages

- $DP$ is based on the repeated, sequential application of the principle of optimality
DP TERMINOLOGY

- **stage** $n$: the point (in time, space, etc.) at which we make a decision for each of the $N$ stages

- **state** $s_n$: the description of a possible configuration of the system in the stage $n$

- **decision** $d_n$: the policy decision at the stage $n$ that transforms the state $s_n (s_{n-1})$ into the state $s_{n+1} (s_n)$ at the next stage $n + 1 (n)$ under forward (backward) recursion
DP TERMINOLOGY

- **transition function**: is the function that determines the relationship between states \( s_n(s_{n-1}), s_{n+1}(s_n) \) and the decision \( d_n \)

- **return function** \( r_n(s_n,d_n) \): the function that reflects the contribution emanating from the decision \( d_n \) made in state \( s_n \) to the accumulated total returns
**DP TERMINOLOGY**

- **optimal decision** $d^*_n$: is the decision at stage $n$, that yields the optimum accumulated returns for the state $s_n$.

- **optimal return** $f^*_n(s_n)$: is the optimal value of the accumulated total returns for the state $s_n$ in stage $n$ corresponding to the decision $d^*_n$.

- **node**: a pair $(n, s_n)$
DP TERMINOLOGY

- DP may proceed via either
  - backward recursion: recursive computation proceeding from an end state to the initial state; or
  - forward recursion: the recursive computation proceeding from the initial state to a feasible end state
The initial stage is the starting point of the problem solution; the final stage contains the ending point(s) of the problem solution; we sequentially construct the solution trajectory through the nodes \((1, s_1^*), (2, s_2^*), \ldots (N, s_N^*)\).
Given the input state at stage $n$, it is possible to determine $f^*_n(s_n)$ from $f^*_{n-1}(s_{n-1})$ by the following recursion relation:

$$f^*_n(s_n) = \underset{d_n}{\text{opt}} \left\{ r_n(s_n,d_n) + f^*_{n-1}(s_{n-1}) \right\} = r_n(s_n,d^*_n) + f^*_{n-1}(s_{n-1})$$

$$f^*_0(s_0) = 0$$
STRUCTURE OF THE BACKWARD RECURSION DP SOLUTION PROCESS

**Stage n**
- $S_n$

**Stage n - 1**
- $S_{n-1}$

*backward recursion*

- $d_n$
- $r_n$
The optimal path from a starting node to a final node has the property that, for any intermediate node, the path must be the optimal path from the starting node to that intermediate node.
ROAD TRIP EXAMPLE

- A poor student is traveling from NY to LA
- To minimize costs, the student plans to sleep at friends’ houses each night in cities along the trip
- Based on past experience he can reach
  - Columbus, Nashville or Louisville after 1 day
  - Kansas City, Omaha or Dallas after 2 days
  - San Antonio or Denver after 3 days
  - LA after 4 days
## DECISION CHOICES

<table>
<thead>
<tr>
<th>day of travel</th>
<th>cities that are possible to reach</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Columbus, Nashville, Louisville</td>
</tr>
<tr>
<td>2</td>
<td>Kansas City, Omaha, Dallas</td>
</tr>
<tr>
<td>3</td>
<td>San Antonio, Denver</td>
</tr>
<tr>
<td>4</td>
<td>LA</td>
</tr>
</tbody>
</table>
We make the following definitions for this problem to be able to cast into the DP framework:

- **the stages** of the problem are the days
- **a state** \( i \) is a reachable city in a stage \( n \)
- **\( c_{ij} \)** is the distance between the directly linked states \( i \) and \( j \) in two successive stages
- **\( f^*_n(i) \)** is the distance of the shortest path from state \( i \) in stage \( n \) to LA
PROBLEM PRELIMINARIES

The objective is to determine \( f_5^*(i) \) using backward regression.

The recursion formula is:

\[
f_n^*(i) = \min_j \{ f_{n-1}(j) + c_{ij} \}
\]

with

\[
f_1^*(i) = 0 \quad \forall i
\]
RECURSION FORMULA

\[ f_n^*(i) = \min_j \left\{ f_{n-1}^*(j) + c_{ij} \right\} \]
BACKWARD REGRESSION FOR ROAD TRIP

solution direction
RECURSION CALCULATIONS

- Stage 1: no computation is required since state 10 is the only possible state and so

\[ f_1^*(10) = 0 \]

- Stage 2: by inspection

\[ f_2^*(8) = 1,030 \quad f_2^*(9) = 1,390 \]
Stage 3: evaluation of three alternatives

\[ f_3^*(5) = \min \left\{ \begin{array}{c} 610 + 1,030 \\ 1,640 \end{array}, \begin{array}{c} 790 + 1,390 \\ 2,180 \end{array} \right\} = 1,640 \]

\[ f_3^*(6) = \min \left\{ \begin{array}{c} 540 + 1,030 \\ 1,570 \end{array}, \begin{array}{c} 940 + 1,390 \\ 2,330 \end{array} \right\} = 1,570 \]

\[ f_3^*(7) = \min \left\{ \begin{array}{c} 790 + 1,030 \\ 1,820 \end{array}, \begin{array}{c} 270 + 1,350 \\ 1,660 \end{array} \right\} = 1,660 \]
Stage 4: evaluation of three alternatives

\[ f^*_4(2) = \min \left\{ \begin{array}{c} \frac{680 + 1,640}{2,320} \\ \frac{790 + 1,570}{2,360} \\ \frac{1,050 + 1,660}{2,710} \end{array} \right\} = 2,320 \]

\[ f^*_4(3) = \min \left\{ \begin{array}{c} \frac{580 + 1,640}{2,220} \\ \frac{760 + 1,570}{2,330} \\ \frac{660 + 1,660}{2,320} \end{array} \right\} = 2,220 \]

\[ f^*_4(4) = \min \left\{ \begin{array}{c} \frac{510 + 1,640}{2,150} \\ \frac{700 + 1,570}{2,270} \\ \frac{830 + 1,660}{2,490} \end{array} \right\} = 2,150 \]
Stage 5: single computation of three alternatives

\[ f_5^*(1) = \min \left\{ \left( \frac{550 + 2,320}{2,870} \right), \left( \frac{900 + 2,220}{3120} \right), \left( \frac{770 + 2,150}{2,920} \right) \right\} = 2,870 \]

Solution results are:

- the length of the shortest path is 2870 mi
- the trajectory of cities is 1 → 2 → 5 → 8 → 10
- every other trajectory with a larger length is suboptimal
We consider the development of a transport network from north slope of Alaska to one of 6 possible shipping points in the U.S.

The network must meet the problem feasibility requirements

- 7 pumping stations from a north slope ground storage plant to a shipping port
- Use of only those paths that are physically and environmentally feasible
OIL TRANSPORT TECHNOLOGY

intermediate region

oil storage

substations

final destinations
Objective: determine a feasible pumping configuration that minimizes the total construction costs of branches of network of the feasible pumping configuration.

\[
\text{total costs} = \sum \text{construction costs of branches of network of the feasible pumping configuration}
\]
Possible approaches to solving such a problem include:

- **direct enumeration**: exhaustive evaluation of all possible paths, which is too costly since there are more than 100 possible paths for this small size problem.

- **myopic decision rule**: at each node, pick as the next node the one reachable by the cheapest path (in case of ties the pick is arbitrary); we show a possible path.
but such a path is not unique and is not guaranteed to be optimal

- serial dynamic programming (DP): we need to construct the problem solution by defining the stages, states and the optimal decisions.
We define an intermediate stage to represent each pumping region and so each such stage corresponds to the set of vertical nodes in regions I, II, ..., VII.

We also define a stage of final destinations and the initial stage for oil storage.

We use *backwards recursion*: we start from any final destination and work *backwards* to the oil storage stage.
DP SOLUTION

- We define a state $s_k$ as a specific pumping station, a final destination or the oil storage facility.

- A decision refers to the selection of the branch from each state $s_k$.

- At each intermediate stage there are at most three choices for a decision $d_k$:

  $L \leftrightarrow$ left  \hspace{1cm} $F \leftrightarrow$ forward  \hspace{1cm} $R \leftrightarrow$ right
**DP SOLUTION**

Stage $k$

- $S_k$
- $L$
- $S'_{k-1}$
- $F$
- $R$
- $S''_{k-1}$

Stage $k-1$

- $S_{k-1}$

*backward recursion*
We further define for a decision $d_n$ in the state $s_n$:

- The return function $\{r_n(s_n, d_n) + f_{n-1}^*(s_{n-1})\}$: which evaluates the accumulated costs associated with the decision $d_n$ that serves to transition from state taken in state $s_n$ to $s_{n-1}$.

- The optimal return function
  
  $f_n^*(s_n) = \min_{d_n} \{r_n(s_n, d_n) + f_{n-1}^*(s_{n-1})\}$

which expresses the least-cost path from $s_{n-1}$ to a feasible final destination.
Note that the optimal decision $d_n^*$ ensures that if $s_n$ lies on the least-cost path, the portion of the least-cost path from $s_{n-1}$ to a feasible final destination is already identified and its costs $f_n^*(s_n)$ are known; this information is also useful whenever a suboptimal path through $s_n$ is needed.
**DP PROBLEM PRELIMINARIES**

- The Principle of Optimality ensures that there is no path from $s_n$ to a feasible final destination that is less costly than $f^*_n(s_n)$.

- We use *backward recursion* to solve the problem starting at the terminal region (stage 0) and proceeding backwards to the oil storage facility (stage 8).
**DP SOLUTION:**

**STAGE 1 ↔ REGION VII**

<table>
<thead>
<tr>
<th>$s_1$</th>
<th>$d_1$</th>
<th>$d^*_1$</th>
<th>$f^*_1(s_1)$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$R$</td>
<td>$L$</td>
<td>$F$</td>
</tr>
<tr>
<td>$A$</td>
<td>7</td>
<td></td>
<td>$R$ 7</td>
</tr>
<tr>
<td>$B$</td>
<td>6</td>
<td>3</td>
<td>$F$ 3</td>
</tr>
<tr>
<td>$C$</td>
<td>7</td>
<td>5</td>
<td>$L$ 5</td>
</tr>
<tr>
<td>$D$</td>
<td>6</td>
<td>5</td>
<td>$F$ 3</td>
</tr>
<tr>
<td>$E$</td>
<td>7</td>
<td>8</td>
<td>$F$ 5</td>
</tr>
<tr>
<td>$F$</td>
<td>4</td>
<td>2</td>
<td>$L$ 2</td>
</tr>
</tbody>
</table>

Values of $\{r_2(s_2, d_2) + f^*_1(s_1)\}$

Optimal decision return
**DP SOLUTION:**

**STAGE 2 ↔ REGION VI**

<table>
<thead>
<tr>
<th>$s_2$</th>
<th>$d_2$</th>
<th>$d^*_2$</th>
<th>$f^*_2(s_2)$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$R$</td>
<td></td>
<td>$R$</td>
</tr>
<tr>
<td>$A$</td>
<td>10</td>
<td>12</td>
<td>10</td>
</tr>
<tr>
<td>$B$</td>
<td>9</td>
<td>12</td>
<td>7</td>
</tr>
<tr>
<td>$C$</td>
<td>5</td>
<td>6</td>
<td>7</td>
</tr>
<tr>
<td>$D$</td>
<td>8</td>
<td>7</td>
<td>6</td>
</tr>
<tr>
<td>$E$</td>
<td>7</td>
<td>6</td>
<td>6</td>
</tr>
</tbody>
</table>

**optimal decision**

Values of $r_1(d_1, s_1)$ to a feasible terminal state from $s_2$ to a final destination.

Cumulative costs in proceeding from $s_2$ to a final destination.
OIL TRANSPORT : \textit{STAGE 2}

\[ f_2^*(s_2) = \min_{d_2} \left( r_2(s_2, d_2) + f_1^*(s_1) \right) \]

a function of only \( s_1 \)

\[ \downarrow \]

for a given \( d_2, s_1 \) is set
### DP SOLUTION:

**STAGE 3 ⇔ REGION V**

\[
f_3^*(s_3) = \min_{d_3} \left\{ r_3(s_3, d_3) + f_2^*(s_2) \right\}
\]

<table>
<thead>
<tr>
<th>( s_3 )</th>
<th>( d_3 )</th>
<th>( d_3^* )</th>
<th>( f_3^*(s_3) )</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>R</td>
<td>L</td>
<td>F</td>
</tr>
<tr>
<td>A</td>
<td>14</td>
<td>16</td>
<td>R</td>
</tr>
<tr>
<td>B</td>
<td>14</td>
<td>17</td>
<td>15</td>
</tr>
<tr>
<td>C</td>
<td>10</td>
<td>5</td>
<td>13</td>
</tr>
<tr>
<td>D</td>
<td>9</td>
<td>12</td>
<td>9</td>
</tr>
<tr>
<td>E</td>
<td>12</td>
<td>15</td>
<td>L</td>
</tr>
</tbody>
</table>

Values of \( \{ r_3(s_3, d_3) + f_2^*(s_2) \} \)
**DP SOLUTION:**

**STAGE 4 ↔ REGION IV**

\[
f_4^*(s_4) = \min_{d_4} \{r_4(s_4,d_4) + f_3^*(s_3)\}
\]

<table>
<thead>
<tr>
<th>(s_4)</th>
<th>(d_4)</th>
<th>(d_4^*)</th>
<th>(f_4^*(s_4))</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>R</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>L</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>F</td>
<td></td>
<td></td>
</tr>
<tr>
<td>B</td>
<td>17</td>
<td>18</td>
<td>23</td>
</tr>
<tr>
<td>C</td>
<td>15</td>
<td>22</td>
<td>16</td>
</tr>
<tr>
<td>D</td>
<td>18</td>
<td>17</td>
<td>16</td>
</tr>
<tr>
<td>E</td>
<td>16</td>
<td>21</td>
<td></td>
</tr>
</tbody>
</table>

Values of \(\{r_4(s_4,d_4) + f_3^*(s_3)\}\)
DP SOLUTION:
STAGE 5 ↔ REGION III

\[ f^*_5(s_5) = \min_{d_5} \left\{ r^*_5(s_5, d_5) + f^*_4(s_4) \right\} \]

<table>
<thead>
<tr>
<th>( s_5 )</th>
<th>( d_5 )</th>
<th>( d^*_5 )</th>
<th>( f^*_5(s_5) )</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>R</td>
<td>L</td>
<td>F</td>
</tr>
<tr>
<td>A</td>
<td>19</td>
<td></td>
<td></td>
</tr>
<tr>
<td>B</td>
<td>18</td>
<td>18</td>
<td>R,F</td>
</tr>
<tr>
<td>C</td>
<td>24</td>
<td>23</td>
<td>17</td>
</tr>
<tr>
<td>D</td>
<td>20</td>
<td>19</td>
<td>25</td>
</tr>
<tr>
<td>E</td>
<td>21</td>
<td>17</td>
<td>F</td>
</tr>
<tr>
<td>F</td>
<td>20</td>
<td></td>
<td>L</td>
</tr>
</tbody>
</table>
**DP SOLUTION:**

**STAGE 6 ↔ REGION II**

\[ f_6^*(s_6) = \min_{d_6} \{ r_6(s_6, d_6) + f_5^*(s_5) \} \]

<table>
<thead>
<tr>
<th>( s_6 )</th>
<th>( d_6 )</th>
<th>( r )</th>
<th>( L )</th>
<th>( F )</th>
<th>( d^*_6 )</th>
<th>( f_6^*(s_6) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>25</td>
<td>24</td>
<td>( F )</td>
<td>24</td>
<td></td>
<td></td>
</tr>
<tr>
<td>B</td>
<td>21</td>
<td>25</td>
<td>24</td>
<td>( R )</td>
<td>21</td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>28</td>
<td>21</td>
<td>23</td>
<td>( L )</td>
<td>21</td>
<td></td>
</tr>
<tr>
<td>D</td>
<td>27</td>
<td>26</td>
<td>29</td>
<td>( L )</td>
<td>26</td>
<td></td>
</tr>
<tr>
<td>E</td>
<td>26</td>
<td>23</td>
<td>22</td>
<td>( F )</td>
<td>22</td>
<td></td>
</tr>
<tr>
<td>F</td>
<td>18</td>
<td>23</td>
<td>( L )</td>
<td>18</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Values of \( r_6(s_6, d_6) + f_5^*(s_5) \) for each combination of \( s_6 \) and \( d_6 \).
**DP SOLUTION:**

**STAGE 7 ↔ REGION I**

\[ f^*_7(s_7) = \min_{d_7} \left\{ r_7(s_7, d_7) + f^*_6(s_6) \right\} \]

<table>
<thead>
<tr>
<th>( s_7 )</th>
<th>( d^*_7 )</th>
<th>( f^*_7(s_7) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>R</td>
<td>27</td>
</tr>
<tr>
<td>B</td>
<td>R,F</td>
<td>26</td>
</tr>
<tr>
<td>C</td>
<td>L</td>
<td>25</td>
</tr>
<tr>
<td>D</td>
<td>R</td>
<td>25</td>
</tr>
<tr>
<td>E</td>
<td>R</td>
<td>27</td>
</tr>
</tbody>
</table>
The optimal trajectory

- For the last stage 8

<table>
<thead>
<tr>
<th>$s_8$ ( F_8(s_8) )</th>
<th>( d_8 )</th>
<th>( d_8^* )</th>
<th>( f_8^*(s_8) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>33</td>
<td>30</td>
<td>30</td>
<td>B,E</td>
</tr>
<tr>
<td>32</td>
<td>33</td>
<td>B,E</td>
<td>30</td>
</tr>
<tr>
<td>33</td>
<td>30</td>
<td>30</td>
<td>B,E</td>
</tr>
</tbody>
</table>

\[
f_8^*(s_8) = \min \{27 + 6, 26 + 4, 25 + 7, 25 + 8, 27 + 3\}
= 30
\]

- To find trajectory we retrace forwards through the stages 7, 6, \ldots, 1 and obtain
OIL TRANSPORT PROBLEM
SOLUTION

[Diagram showing a network of substations connected to oil storage and final destinations with numerical values on the edges.]

oil storage       substations

ECE 573 © 2001-2015 George Gross, University of Illinois at Urbana-Champaign, All Rights Reserved.
Multiple optimal solutions are obtained: this is a very desirable outcome of having multiple solutions at no additional expense.

The problem was logically decomposed into stages with the amount of computation in each stage being a function of the number of the states in the stage.
All sub-optimal paths are immediately available

- we may calculate the least cost path to any suboptimal shipping point other than $D$

- we can also determine the suboptimal path if the construction of a previously feasible path cannot be undertaken
We got to the node IV-C but we are unable to obtain the required permits to build from that node to node V-D. From the stage 3 calculations, we know the optimal completion costs from any station in region V. The least cost path from IV-C to the final destination $D$ is then:

IV - C → V - C → VI - D → VII - D → final destination $D$

with a new optimal cost of 31.
SMALL SYSTEM EXAMPLE: LOAD DEMAND AND GENERATION SUPPLY

<table>
<thead>
<tr>
<th>$k$</th>
<th>$d(k)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>200</td>
</tr>
<tr>
<td>1</td>
<td>300</td>
</tr>
<tr>
<td>2</td>
<td>200</td>
</tr>
<tr>
<td>3</td>
<td>200</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>unit</th>
<th>capacity (MW)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>300</td>
</tr>
<tr>
<td>2</td>
<td>200</td>
</tr>
</tbody>
</table>

no other unit constraints are considered
SMALL SYSTEM EXAMPLE
DEFINITIONS

- **state**: a configuration of the generation system in a given hour $k$

- **state space**: the collection of all possible states in all the hours of the problem horizon

- **stage**: the phase of the problem associated with the system in the hour $k$
Definition of states:

\[ \alpha \iff (0,0) \quad \gamma \iff (1,0) \]
\[ \beta \iff (0,1) \quad \delta \iff (1,1) \]

Initial state: \((1,0)\iff\text{unit 1 is up and unit 2 is down}\)

Decision at any state is to start-up or shut-down unit 1 or unit 2 or both or to keep them as in the previous stage.

We use forward recursion to solve the DP problem.
STATE SPACE

stages

0

1

2

3

initial configuration

\begin{align*}
\gamma & \quad \delta \\
\beta & \quad \delta \\
\alpha & \quad \delta
\end{align*}
FEASIBLE STATE SPACE

assumption: we assume for the sake of simplicity that the problem terminal state is $\gamma$
TRANSITION COSTS  \( c_{ij} \)

- The transition costs from a state in stage \( n \) to a state in stage \( n + 1 \) includes both the operating costs and the incurred start-up costs, if any.

- The transition costs from state \( \gamma \) to state \( \delta \) are

\[
r_1 \left( \gamma, \gamma \rightarrow \delta \right) = f_1^F \left[ u_1(1), p_1(1) \right] + f_2^S \left[ u_2(1), p_2(1) \right] + f_2^F \left[ u_2(1), p_2(1) \right]
\]

\( \text{operating costs of unit 1} \)
\( \text{start-up costs of unit 2} \)
\( \text{operating costs of unit 2} \)
COSTS OF FEASIBLE TRANSITIONS

forward recursion
The general recursion is

$$f^*_n(s_n) = \min_{d_{n-1}} \{r_{n-1}(s_{n-1},d_{n-1}) + f^*_{n-1}(s_{n-1})\}$$

Stage 1 has two possible states with the costs

given by

$$f^*_1(\gamma) = 3 \quad f^*_1(\delta) = 7$$
FORWARD RECURSION CALCULATIONS

Stage 2 has three possible states

\[ f_2^*(\beta) = \min \left\{ 6 + f_1^*(\gamma), 8 + f_1^*(\delta) \right\} = 9 \]

\[ f_2^*(\gamma) = \min \left\{ 9 + f_1^*(\gamma), 8 + f_1^*(\delta) \right\} = 12 \]

\[ f_2^*(\delta) = \min \left\{ 11 + f_1^*(\gamma), 8 + f_1^*(\delta) \right\} = 12 \]

Stage 3 has an assumed state \( \gamma \)

\[ f_3^*(\gamma) = \min \left\{ 1 + f_2^*(\delta), 4 + f_2^*(\gamma), 2 + f_2^*(\beta) \right\} = 11 \]
Optimal path is given by

\[ \gamma_{\text{stage } 0} \rightarrow \gamma_{\text{stage } 1} \rightarrow \beta_{\text{stage } 2} \rightarrow \gamma_{\text{stage } 3} \]

Optimal costs \( = f^*_3(\gamma) = 11 \)
GENERAL CASE: THE STATE SPACE

- For a system with \( N \) units, the unconstrained state space is the set of all possible combinations of the \( N \) units.

\[
\begin{array}{cccccccc}
\text{unit} & 1 & 2 & 3 & \ldots & N \\
1 & 1 & 1 & \ldots & 1 \\
0 & 1 & 1 & \ldots & 1 \\
1 & 0 & 1 & \ldots & 1 \\
\ldots & \ldots & \ldots & \ldots & \ldots \\
\end{array}
\]

\[
2^N \cdot H \text{ states} \]

- With minimum up and minimum down times considered for a set of identical units, there are

\[
\left( T^u + T^d \right)^N \cdot H \text{ states}
\]
CURSE OF DIMENSIONALITY
ILLUSTRATION

- Example: $T_i^u = T_i^d = 6, N = 12 \Rightarrow 12^{12}$ states in each hour

- Load requirements eliminate most of the states

- Minimum up/minimum down time requirements eliminate most of the transition arcs

- Example: suppose 99.99% of the states are eliminated; a hundred million states are still left!
ADVANTAGES OF \textit{DP} – BASED APPROACHES

- Provide means for the systematic enumeration of possible states and state transitions
- Yield optimum solutions
- Are appropriate for handling discrete variables, nonlinear cost functions and multiple and highly complex constraints
DISADVANTAGES OF $DP$ - BASED APPROACHES

- “Curse of dimensionality” leads to computational intractability since execution time increases exponentially with the number of units.

- No sensitivity information is provided but suboptimal solutions can be readily identified.
DP – BASED APPROACHES

- For computational tractability, approximations and heuristics become necessary to reduce the problem dimensionality so as to allow its solution.

- We examine how heuristics may be used to do this effectively.

- We provide a few practical examples of DP – based approaches.
DP METHODS AND HEURISTICS

- Basic approach: introduction of heuristic rules to limit the size of the state space
- DP sequential combinations ($DP - SC$) approach uses a combination of DP with priority list notions

Procedure consists of
- definition of the priority list
- reduction of the size of the state space using the heuristic based on the priority list
- application of DP to the reduced size state space
EXAMPLE: \( DP – SC \)

- A system with \( N = 4 \) units which are ordered with

  1 (4) being the most economic (most expensive)

- Whenever we use a priority heuristic, there is the implicit implication that **we cannot commit a more expensive unit before a cheaper unit is committed**
### EXAMPLE: $DP - SC$

<table>
<thead>
<tr>
<th>Unit 1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 1 1 1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1 0 1 1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0 0 1 1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1 1 0 1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0 1 0 1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1 0 0 1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0 0 0 1</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

16 states

<table>
<thead>
<tr>
<th>Unit 1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 1 1 1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1 1 1 0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1 1 0 0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1 0 0 0</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

5 states
DP METHODS AND HEURISTICS

- Truncated combination ($DP \otimes TC$): approach uses $DP$ limited to a pre-defined search range or combination window.

- Procedure consists of the steps:
  - Definition of the priority list.
  - Definition of position in the priority list and range of the combination ‘window’, i.e., the number of units in the ‘window’.
  - Application of $DP$ to the reduced size state space.
EXAMPLE: $DP - TC$

- Example: $N = 4$, with set position to 3 with the range set to 2 and units 1 & 2 scheduled by priority list.
- First 2 d State space is reduced as follows:

<table>
<thead>
<tr>
<th>unit</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>X</td>
<td>Y</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>X</td>
<td>Y</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>X</td>
<td>Y</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>X</td>
<td>Y</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

$XY$ represents priority list scheduling

Priority list combination window

$4$ states
DP METHODS AND HEURISTICS

- **DP sequential truncated combinations**
  
  \((DP – STC)\): approach uses a combination of the **DP – SC** and the **DP – TC** schemes

- Procedure consists of the following steps

  - definition of the *priority list*
  - reduction of the size of the state space for the specified sequential combination input
  - definition of the *position and range* of the combination window
  - application of **DP** to the reduced size state space
EXAMPLE: $DP - STC$ SCHEME

- $N = 4$

- The specified $SC$ input consists of the first two units: the possible states for units 1 and 2 are:

```
1 1 1 0 0 0
```

3 states

- The position of the combination window is 3 with a range of 2; possible states for the combination window units 3 and 4 are:

```
1 1 0 1 1 0 0 0
```

4 states
EXAMPLE: $DP \rightarrow STC$

- The resulting state space is the (Cartesian) product of the two state spaces above:

<p>| | | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**12 states**
**EXAMPLE: 27 – UNIT SYSTEM**

Comparative Results for Three Schemes

<table>
<thead>
<tr>
<th>scheme</th>
<th>total system costs $</th>
<th>CPU seconds</th>
</tr>
</thead>
<tbody>
<tr>
<td>priority list</td>
<td>995,232</td>
<td>18</td>
</tr>
<tr>
<td>DP – SC</td>
<td>951,699</td>
<td>37</td>
</tr>
<tr>
<td>DP – TC *</td>
<td>949,781</td>
<td>268</td>
</tr>
</tbody>
</table>

* with a search range of 10 units (1,024 combinations)
EXAMPLE: 27 – UNIT SYSTEM

The graph shows the relationship between the size of the search range and total system costs ($1,000). The x-axis represents the size of the search range, while the y-axis represents total system costs. The graph indicates that as the size of the search range increases, the total system costs also increase. The CPU seconds are shown on the right y-axis, and they increase proportionally with the size of the search range.
EXAMPLE: *DP METHODS AND HEURISTICS*

- System with 22 cycling units and 30 peaking units
- Comparative results run on a Cyber 173/8

<table>
<thead>
<tr>
<th>method</th>
<th>total system costs $</th>
<th>CPU seconds</th>
</tr>
</thead>
<tbody>
<tr>
<td>priority list</td>
<td>413,820</td>
<td>13</td>
</tr>
<tr>
<td><em>DP – SC</em> (6)</td>
<td>381,170</td>
<td>29</td>
</tr>
<tr>
<td><em>DP – STC</em> (6)</td>
<td>380,462</td>
<td>114</td>
</tr>
<tr>
<td><em>DP – TC</em> (6)</td>
<td>372,262</td>
<td>982</td>
</tr>
</tbody>
</table>
DP METHODS AND HEURISTICS: ADVANTAGES

- Numerical tractability is possible because of the reduced state space and size.

- Search for the optimum within a ‘good’ region of the state space.

- Availability of suboptimal solutions.
DP METHODS AND HEURISTICS: DISADVANTAGES

- Difficult to establish effective search range
- Optimum need *not* be in the reduced state space
- Lack of sensitivity information