Design and Analysis of Algorithm

Lecture-14:
Dynamic Programming

Contents

Dynamic Programming

Definition

Dynamic Programming is an algorithm design technique for optimization problems: often minimizing or maximizing.

Like divide and conquer, Dynamic Programming solves problems by combining solutions to sub-problems.

Unlike divide and conquer, sub-problems are not independent.

Dynamic Programming

The term Dynamic Programming comes from Control Theory, not computer science. Programming refers to the use of tables (arrays) to construct a solution.

In dynamic programming we usually reduce time by increasing the amount of space

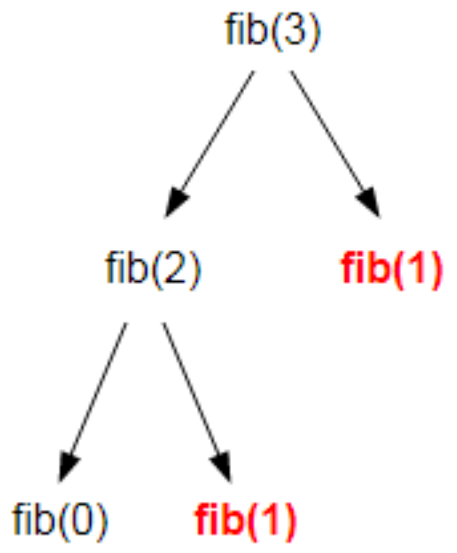
The table is then used for finding the optimal solution to larger problems.

Time is saved since each sub-problem is solved only once.

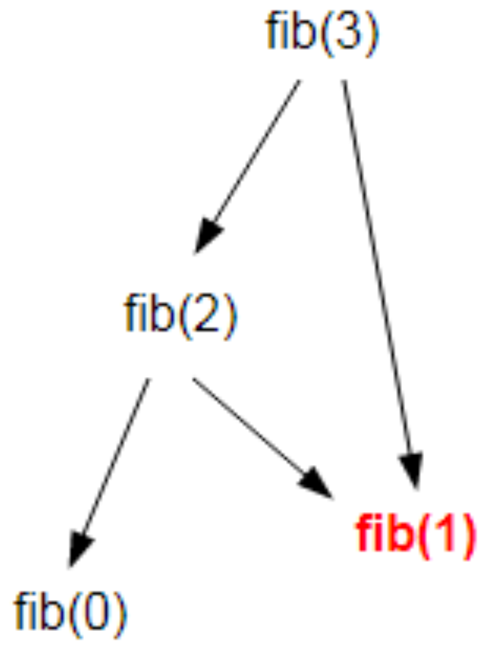
Dynamic Programming is mainly used when solutions of same sub-problems are needed again and again.

Divide and Conquer Vs Dynamic

Divide and Conquer approach



Dynamic Programming approach



Divide and Conquer Vs Dynamic

Divide and Conquer approach

Fibonacci Series

```
{
  if ( n <= 1 )
      return n;
    return fib(n-1) + fib(n-2);
}
```

Finding factorial of $n \geq 1$

$$f(n) = n \times f(n-1)$$

Dynamic Programming approach

```
Fib(n)
 { mem[0] = 0
   mem[1] = 1
  for i = 2...n
        mem[i] = mem[i-2] + mem[i-1]
   return mem[n]
}
```

```
for $i = 1{:}n\{$
        $f(i) = i * f(i-1)$
}
```

Elaborate Dynamic Programing

0-1 Knapsack

Given weights and values of n items, put these items in a knapsack of capacity W to get the maximum total value in the knapsack.

Given that n = 4 W = 5

Elements

1: $(w_1 = 2, benefit = 3)$
2: $(w_2 = 3, benefit = 4)$
3: $(w_3 = 4, benefit = 5)$
4: $(w_4 = 5, benefit = 6)$

0	0	1	2	3	4	5
1	0	0	3	3	3	3
2	0					
3	0					
4	0					
5	0					

Assembly Line Scheduling

Definition

An automobile chassis enters each assembly line, has parts added to it at a number of stations, and a finished auto exits at the end of the line.

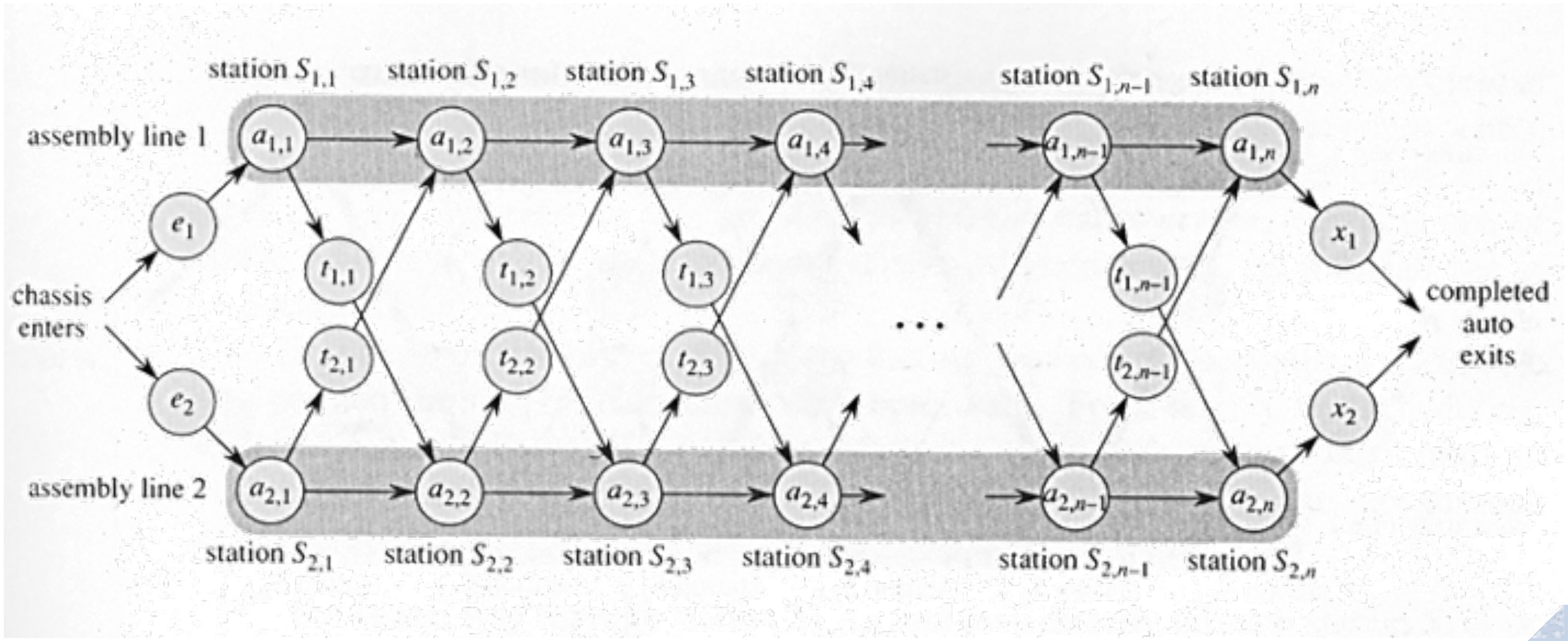
Consider that each assembly line has n stations, numbered $j = 1, 2, \ldots, n$.

We denote the j^{th} station on line i (where i is 1 or 2) by $S_{i,j}$.

The j^{th} station on line 1 , $S(1, j)$ performs the same function as the j^{th} station on line 2 , $S(2, j)$.

Details on sub-stations

The stations were built at different times and with different technologies, however, so that the time required at each station varies, even between stations at the same position on the two different lines. We denote the assembly time required at station S_{ij} by a_{ij}.



Time to enter a station is denoted by e. Time to exit from the assembly is given by x

Details of the problem

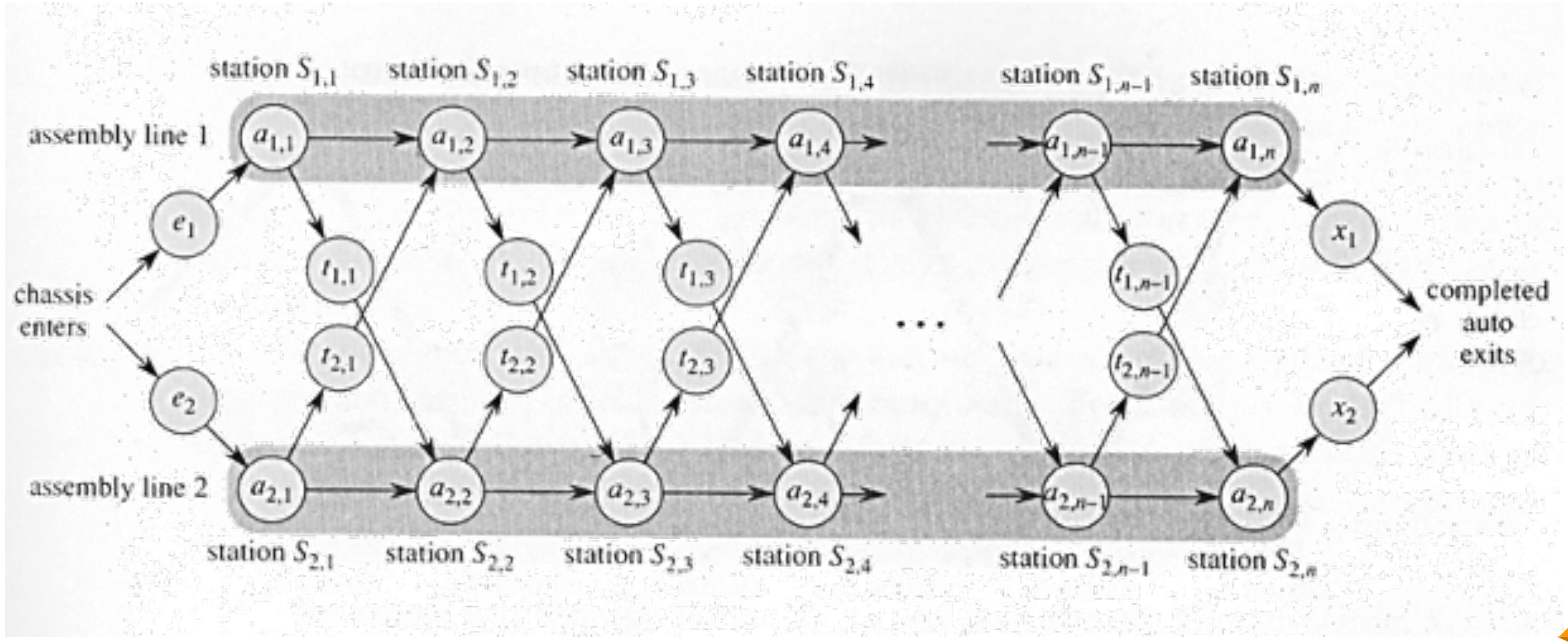
Normally, once a chassis enters an assembly line, it passes through that line only. The time to go from one station to the next within the same assembly line is negligible.

Occasionally a special rush order comes in, and the customer wants the automobile to be manufactured as quickly as possible.

For the rush orders, the chassis still passes through the n stations in order, but the factory manager may switch the partially-completed auto from one assembly line to the other after any station.

Fastest way

- ✓ The fast way through station $S_{1,j}$ is either
  - the fastest way through Station $S_{1,j-1}$ and then directly through station $S_{1,j}$, or
  - the fastest way through station $S_{2,j-1}$, a transfer from line 2 to line 1, and then through station $S_{1,j}$.
- ✓ Using symmetric reasoning, the fastest way through station $S_{2,j}$ is either
  - the fastest way through station $S_{2,j-1}$ and then directly through Station $S_{2,j}$, or
  - the fastest way through station $S_{1,j-1}$, a transfer from line 1 to line 2, and then through Station $S_{2,j}$.

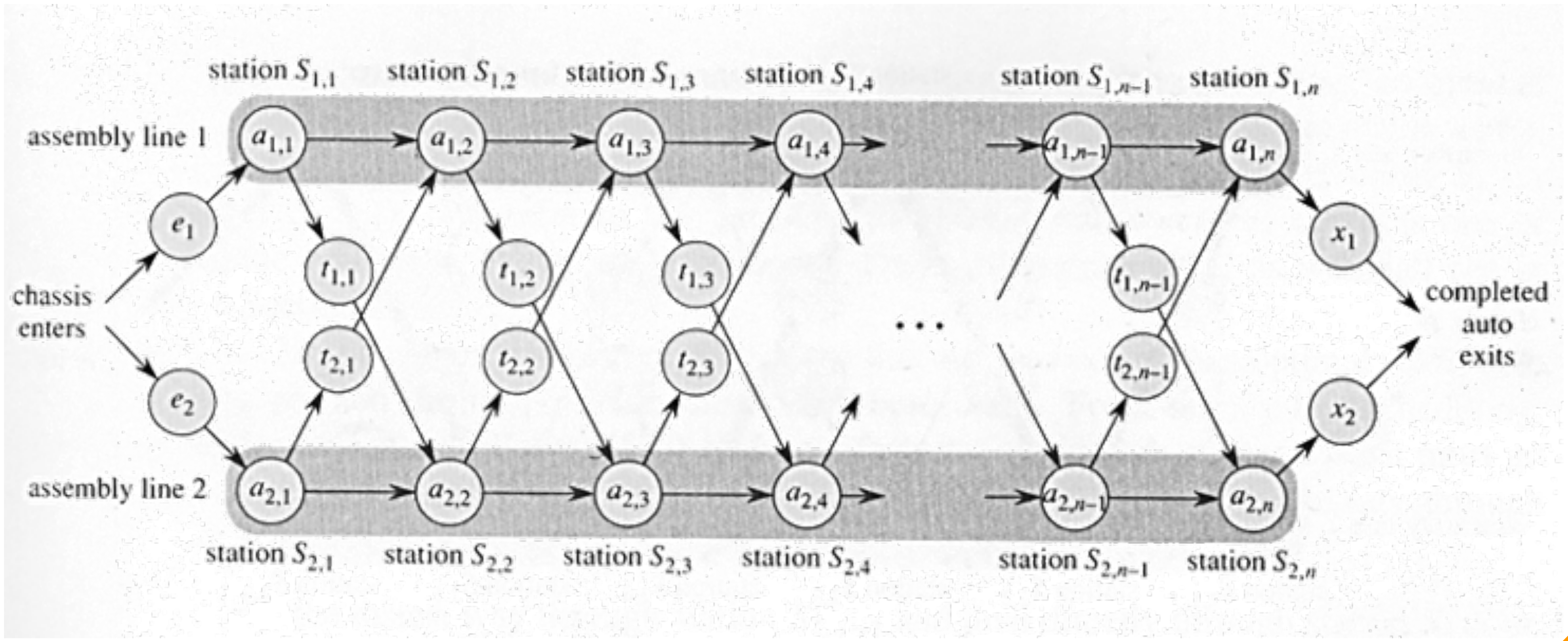


Fastest Way

$$f^* \quad \min(f_1[n] \quad x_1, f_2[n] \quad x_2)$$

$$f_1[j] \quad \begin{array}{ll} e_1 \quad a_{1,1} & \text{if'} \ j \quad 1, \\ \min(f_1[j \quad 1] \quad a_{1,j}, f_2[j \quad 1] \quad t_{2,j \ 1} \quad a_{1,j}) & \text{if'} \ j \quad 2 \end{array}$$

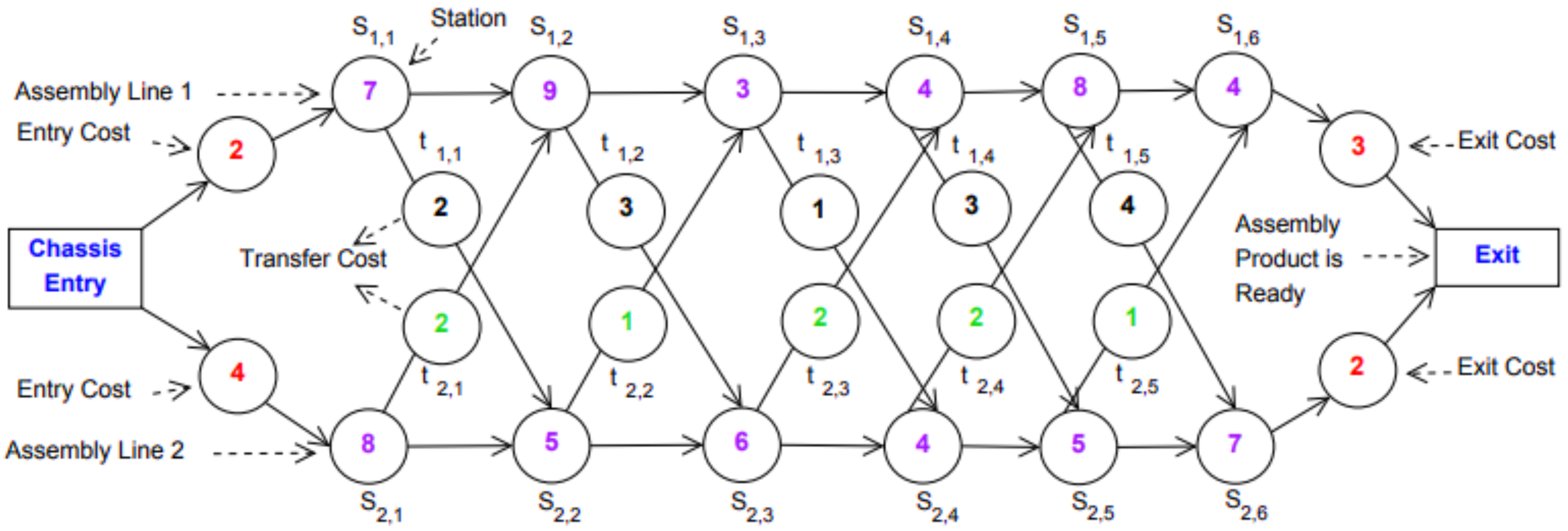
$$f_2[j] \quad \begin{array}{ll} e_2 \quad a_{2,1} & \text{if'} \ j \quad 1, \\ \min(f_2[j \quad 1] \quad a_{2,j}, f_1[j \quad 1] \quad t_{1,j \ 1} \quad a_{2,j}) & \text{if'} \ j \quad 2 \end{array}$$



Algorithm

```
FASTEST-WAY($a, t, e, x, n$)
1  $f_1[1] \leftarrow e_1 + a_{1,1}$
2  $f_2[1] \leftarrow e_2 + a_{2,1}$
3  for $j \leftarrow 2$ to $n$
4        do if $f_1[j\text{-}1] + a_{1,j} \le f_2[j\text{-}1] + t_{2,j\text{-}1} + a_{1,j}$
5              then $f_1[j] \leftarrow f_1[j\text{-}1] + a_{1,j}$
6                    $l_1[j] \leftarrow 1$
7              else $f_1[j] \leftarrow f_2[j\text{-}1] + t_{2,j\text{-}1} + a_{1,j}$
8                    $l_1[j] \leftarrow 2$
9           if $f_2[j\text{-}1] + a_{2,j} \le f_1[j\text{-}1] + t_{1,j\text{-}1} + a_{2,j}$
10             then $f_2[j] \leftarrow f_2[j-1] + a_{2,j}$
11                   $l2[j] \leftarrow 2$
12             else $f_2[j] \leftarrow f_1[j-1] + t_{1,j\text{-}1} + a_{2,j}$
13                   $l_2[j] \leftarrow 1$
14 if $f_1[n] + x_1 \le f_2[n] + x_2$
15    then $f^* = f_1[n] + x_1$
16         $l^* = 1$
17    else $f^* = f_2[n] + x_2$
18         $l^* = 2$
```

Example

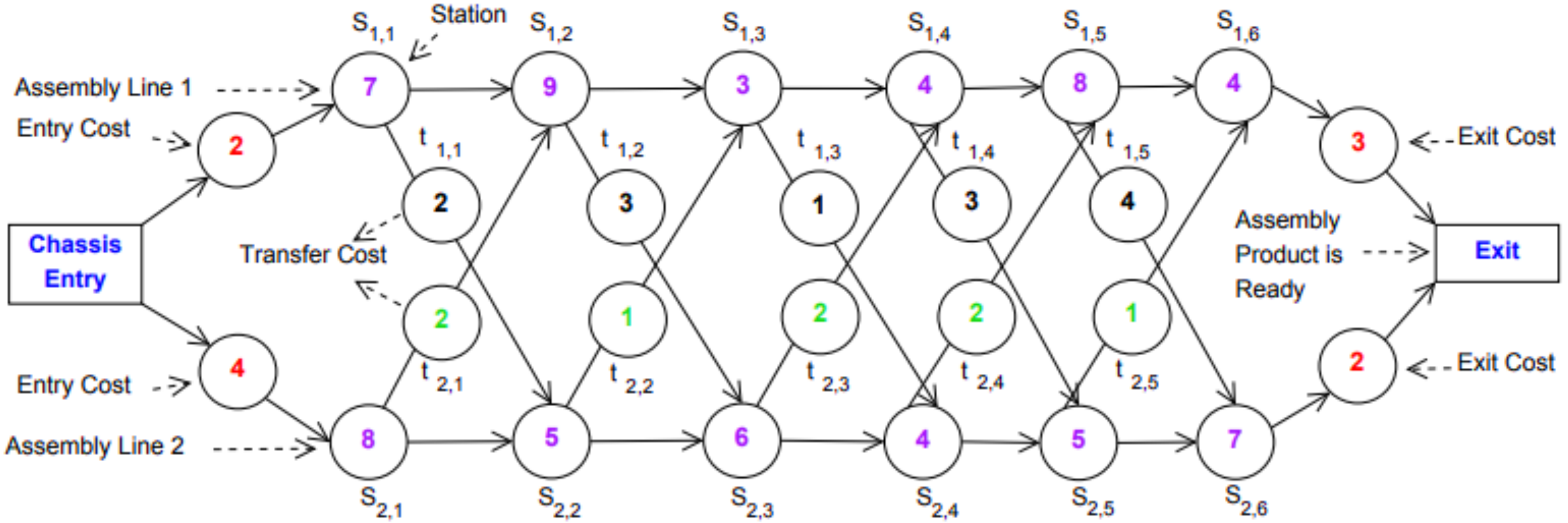


F_1	F_2
1	2
1	1

$$f_1[1] = e_1 + a_{1,1} = 2 + 7 = 9 \text{ and start } = S_{1,1}$$
$$f_2[1] = e_2 + a_{2,1} = 4 + 8 = 12 \text{ and start } = S_{2,1}$$

$$f_1[2] = \min\{f_1[1] + a_{1,2}, f_2[1] + t_{2,1} + a_{1,2}\} = \min\{9 + 9, 12 + 2 + 9\} = 18,\ l_1[2] = 1$$
$$f_2[2] = \min\{f_2[1] + a_{2,2}, f_1[1] + t_{1,1} + a_{2,2}\} = \min\{12 + 5, 9 + 2 + 5\} = 16,\ l_2[2] = 1$$

Example

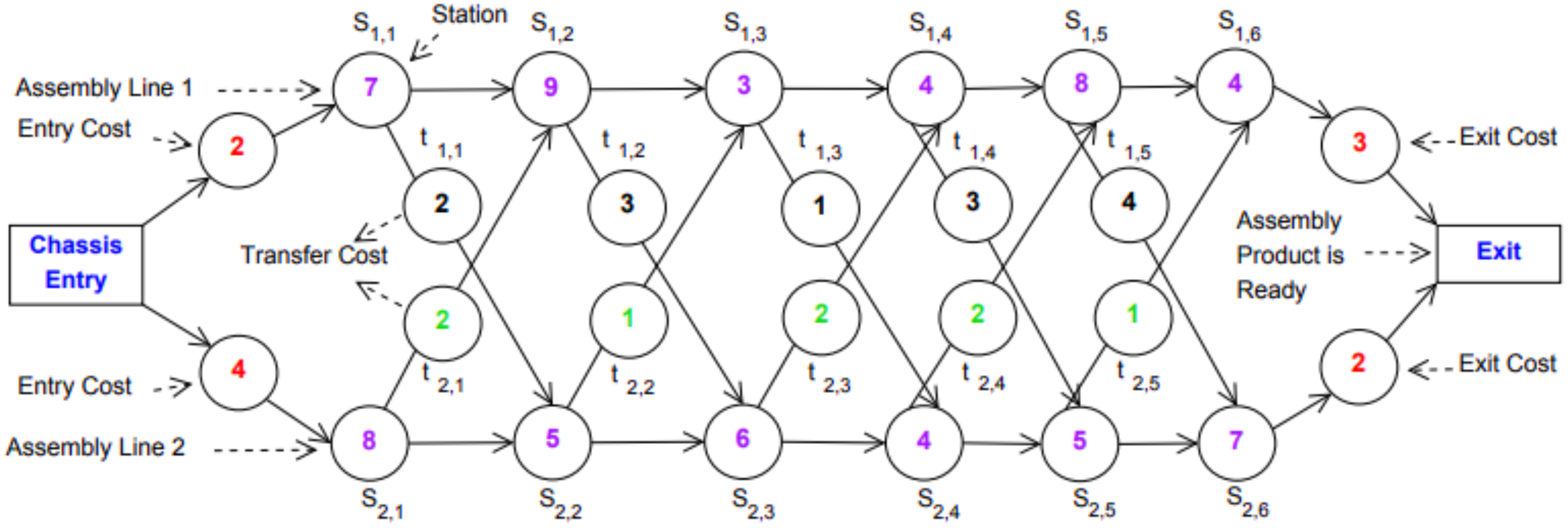


F_1	F_2
1	2
1	1
2	2
1	1

$$f_1[3] = \min\{f_1[2] + a_{1,3}, f_2[2] + t_{2,2} + a_{1,3}\} = \min\{18 + 3, 16 + 1 + 3\} = 20,\ l_1[3] = 2$$
$$f_2[3] = \min\{f_2[2] + a_{2,3}, f_1[2] + t_{1,2} + a_{2,3}\} = \min\{16 + 6, 18 + 3 + 6\} = 22,\ l_2[3] = 2$$

$$f_1[4] = \min\{f_1[3] + a_{1,4}, f_2[3] + t_{2,3} + a_{1,4}\} = \min\{20 + 4, 22 + 2 + 4\} = 24,\ l_1[4] = 1$$
$$f_2[4] = \min\{f_2[3] + a_{2,4}, f_1[3] + t_{1,3} + a_{2,4}\} = \min\{22 + 4, 20 + 1 + 4\} = 25,\ l_2[4] = 1$$

Example



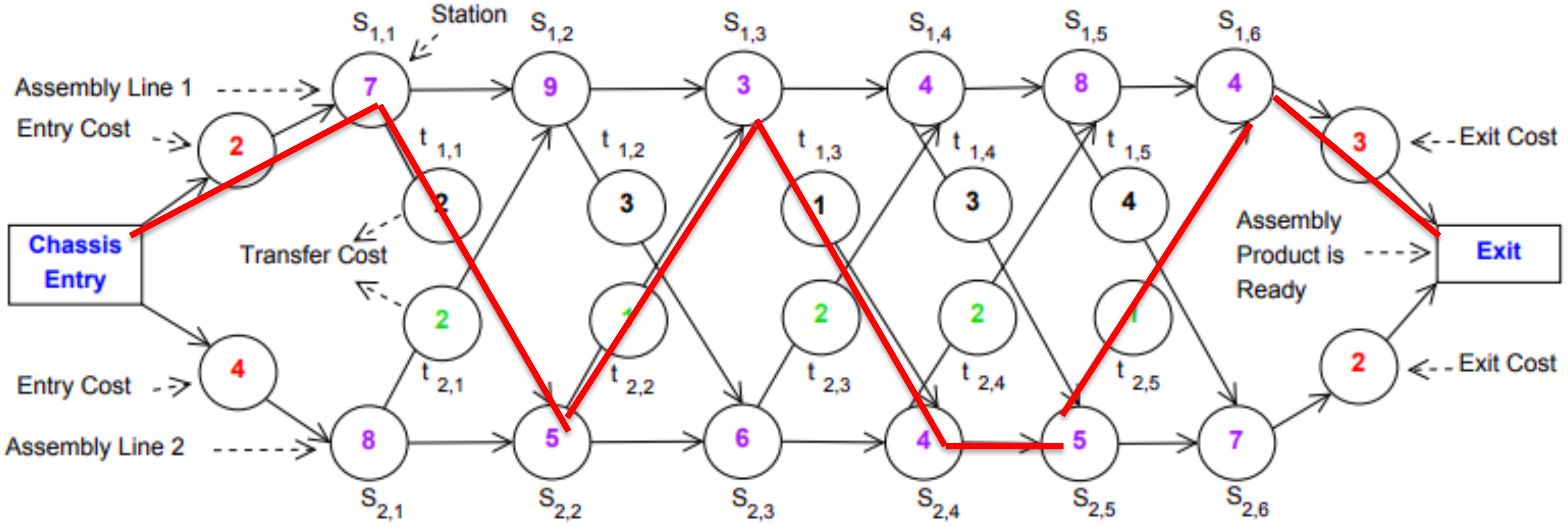
F_1	F_2
1	2
1	1
2	2
1	1
1	2
2	2

$$f_1[5] = \min\{f_1[4] + a_{1,5}, f_2[4] + t_{2,4} + a_{1,5}\} = \min\{24 + 8, 25 + 2 + 8\} = 32,\ l_1[5] = 1$$
$$f_2[5] = \min\{f_2[4] + a_{2,5}, f_1[4] + t_{1,4} + a_{2,5}\} = \min\{25 + 5, 24 + 3 + 5\} = 30,\ l_2[5] = 2$$

$$f_1[6] = \min\{f_1[5] + a_{1,6}, f_2[5] + t_{2,5} + a_{1,6}\} = \min\{32 + 4, 30 + 1 + 4\} = 35,\ l_1[6] = 2$$
$$f_2[6] = \min\{f_2[5] + a_{2,6}, f_1[5] + t_{1,5} + a_{2,6}\} = \min\{30 + 7, 32 + 4 + 7\} = 37,\ l_2[6] = 2$$

Optimum Schedule: $\min\{f_1[6] + x_1, f_2[6] + x_2\} = \{35 + 3, 37 + 2\} = 38\ ;\ l^{OPT} = 1.$

Example



F_1	F_2
1	2
1	1
2	2
1	1
1	2
2	2

Optimum cost=2+7+2+5+1+3+1+4+5+1+4+3