Advanced Algorithms Analysis and Design

By

Nazir Ahmad Zafar

Lecture No 18

2-Line Assembly Scheduling Problem

Today Covered

- 2-Line Assembly Scheduling Algorithm using Dynamic Programming
- Time Complexity
- n-Line Assembly Problem
- Brute Force Analysis
- n-Line Assembly Scheduling Algorithm using Dynamic Programming
- Time Complexity
- Generalization and Applications
- Conclusion

Assembly-Line Scheduling Problem

- There are two assembly lines each with n stations
- The jth station on line i is denoted by $S_{i,\ j}$
- The assembly time at that station is $a_{i,j}$.
- An auto enters factory, goes into line i taking time e_i
- After going through the jth station on a line i, the auto goes on to the (j+1)st station on either line
- There is no transfer cost if it stays on the same line
- It takes time $t_{i,j}$ to transfer to other line after station $S_{i,j}$
- After exiting the nth station on a line, it takes time x_i for the completed auto to exit the factory.
- Problem is to determine which stations to choose from lines 1 and 2 to minimize total time through the factory.

Mathematical Model Defining Objective Function

Base Cases

- $f_1[1] = e_1 + a_{1,1}$
- $f_2[1] = e_2 + a_{2,1}$

Two possible ways of computing $f_1[j]$

- $f_1[j] = f_2[j-1] + t_{2,\ j-1} + a_{1,\ j}$ OR $f_1[j] = f_1[j-1] + a_{1,\ j}$

  For $j = 2, 3, \ldots, n$

  $$f_1[j] = \min (f_1[j-1] + a_{1,\ j},\ f_2[j-1] + t_{2,\ j-1} + a_{1,\ j})$$

Symmetrically

For $j = 2, 3, \ldots, n$

$$f_2[j] = \min (f_2[j-1] + a_{2,\ j},\ f_1[j-1] + t_{1,\ j-1} + a_{2,\ j})$$

Objective function = $f^* = \min(f_1[n] + x_1,\ f_2[n] + x_2)$

Dynamic Algorithm

FASTEST-WAY(a, t, e, x, n)

$f_1[1] \leftarrow e_1 + a_{1,1}$

$f_2[1] \leftarrow e_2 + a_{2,1}$

for $j \leftarrow 2$ to n

  do if $f_1[j - 1] + a_{1, j} \leq f_2[j - 1] + t_{2, j - 1} + a_{1, j}$

    then $f_1[j] \leftarrow f_1[j - 1] + a_{1, j}$

      $l_1[j] \leftarrow 1$

    else $f_1[j] \leftarrow f_2[j - 1] + t_{2, j - 1} + a_{1, j}$

      $l_1[j] \leftarrow 2$

  if $f_2[j - 1] + a_{2, j} \leq f_1[j - 1] + t_{1, j - 1} + a_{2, j}$

Contd..

then $f_2[j] \leftarrow f_2[j - 1] + a_{2,\,j}$

$\qquad\qquad l_2[j] \leftarrow 2$

$\qquad$ **else** $f_2[\ j] \leftarrow f_1[\ j - 1] + t_{1,\,j - 1} + a_{2,\,j}$

$\qquad\qquad l_2[\ j] \leftarrow 1$

if $f_1[n] + x_1 \leq f_2[n] + x_2$

$\quad$ **then** $f^* = f_1[n] + x_1$

$\qquad l^* = 1$

$\quad$ **else** $f^* = f_2[n] + x_2$

$\qquad l^* = 2$

Optimal Solution: Constructing The Fastest Way

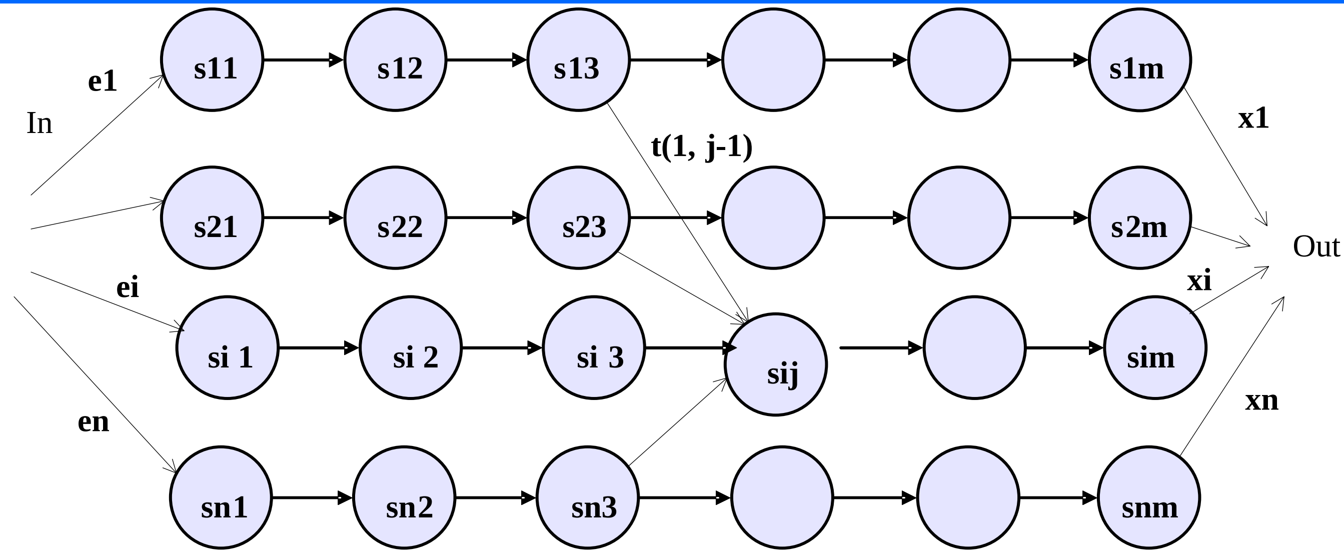
```
Print-Stations (l, n)
    i ← l*
    print “line” i “, station” n
    for j ← n downto 2
        do i ← l_i[j]
            print “line” i “, station” j - 1
```

n-Line Assembly Problem

n-Assembly Line Scheduling Problem

- There are n assembly lines each with m stations
- The jth station on line i is denoted by $S_{i,\,j}$
- The assembly time at that station is $a_{i,j}$.
- An auto enters factory, goes into line i taking time e_i
- After going through the jth station on a line i, the auto goes on to the (j+1)st station on either line
- It takes time $t_{i,j}$ to transfer from line i, station j to line i' and station j+1
- After exiting the nth station on a line i, it takes time x_i for the completed auto to exit the factory.
- Problem is to determine which stations to choose from lines 1 to n to minimize total time through the factory.

n-Line: Brute Force Solution



Total Computational Time

= possible ways to enter in stations at level n x one way Cost

Possible ways to enter in stations at level 1 = n^1

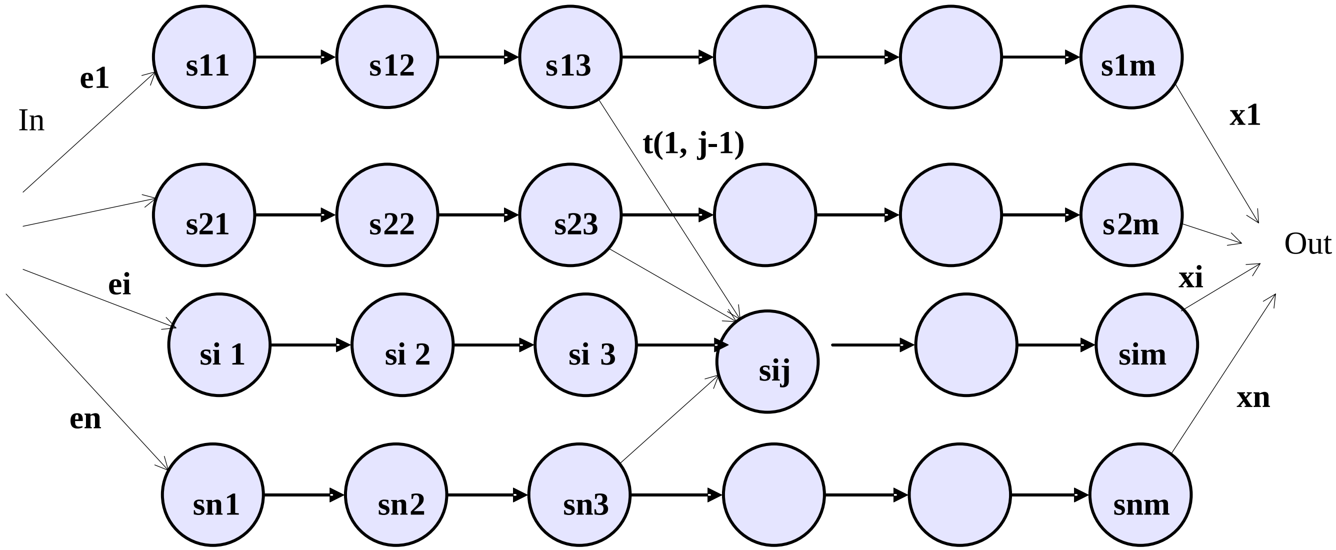
Possible ways to enter in stations at level 2 = $n^2 \cdots$

Possible ways to enter in stations at level m = n^m

Total Computational Time = $\Theta(m.m^n)$

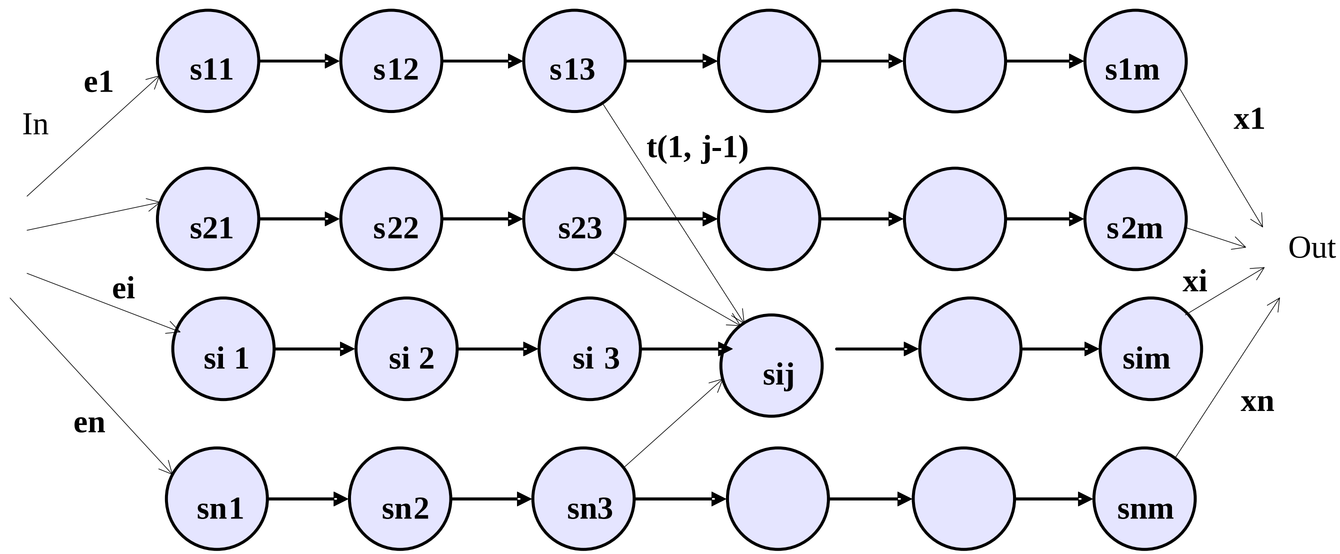
Dynamic Solution

Notations : n-Line Assembly



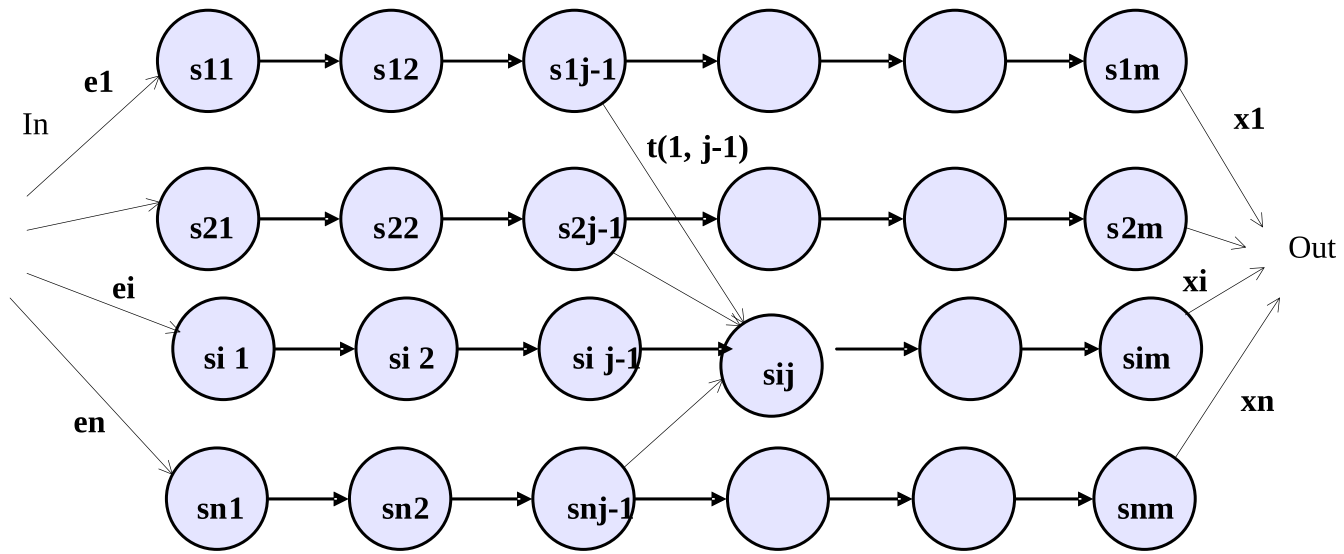
- Let $f_i[j]$ = fastest time from starting point to station $S_{i,\,j}$
- $f_1[m]$ = fastest time from starting point to station $S_{1\ m}$
- $f_2[m]$ = fastest time from starting point to station $S_{2,m}$
- $f_n[m]$ = fastest time from starting point to station $S_{n,m}$
- $l_i[j]$ = The line number, 1 to n, whose station j-1 is used in a fastest way through station $S_{i',\,j}$

Notations : n-Line Assembly



- $t_i[j-1]$ = transfer time from station $S_{i,\ j-1}$ to station $S_{i',\ j}$
- $a[i, j]$ = time of assembling at station $S_{i,\ j}$
- f* = is minimum time through any way
- l* = the line no. whose m^{th} station is used in a fastest way

Possible Lines to reach Station S(i, j)



Time from Line 1, f[1, j-1] + t[1, j-1] + a[i, j]

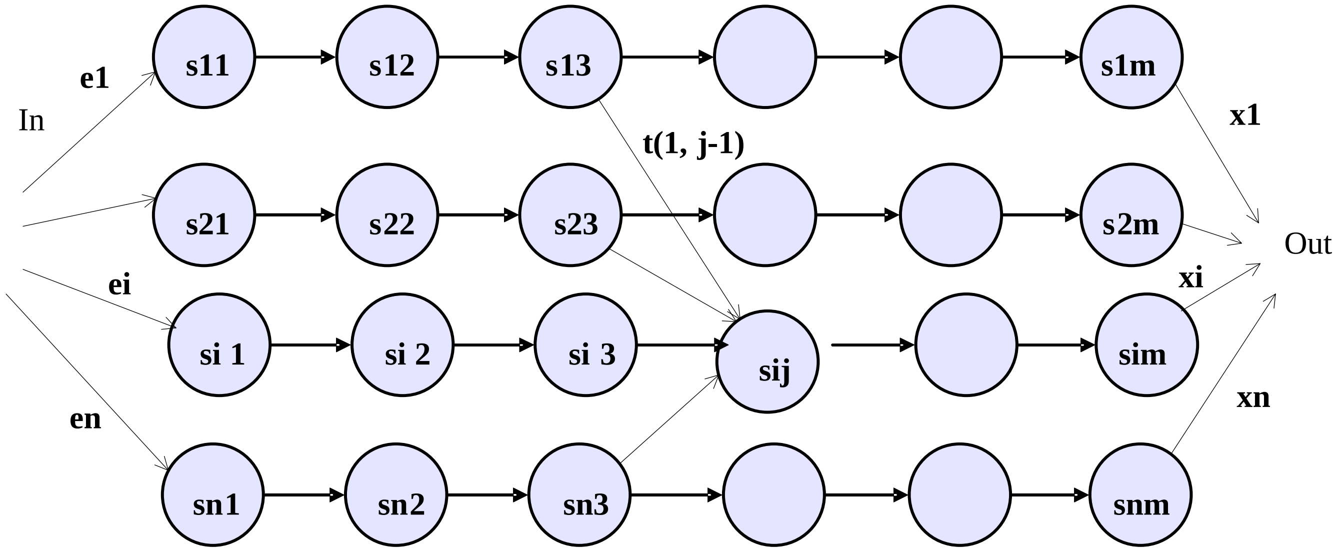
Time from Line 2, f[2, j-1] + t[2, j-1] + a[i, j]

Time from Line 3, f[3, j-1] + t[3, j-1] + a[i, j]

. . .

Time from Line n, f[n, j-1] + t[n, j-1] + a[i, j]

Values of f(i, j) and l* at Station S(i, j)



f[i, j] = min{f[1, j-1] + t[1, j-1] + a[i, j], f[2, j-1] + t[2, j-1] + a[i, j], . . . , f[n, j-1] + t[n, j-1] + a[i, j]}

f[1, 1] = e_1 + a[1, 1]; f [2, 1] = e_2 + a[2, 1], … ,f [n, 1] = e_n + a[n, 1]

f* = min{f[1, n] +x_1, f[2, n] + x_2, . . , f[n, m] + x_n}

l* = line number of m^{th} station used

n-Line Assembly: Dynamic Algorithm

```
FASTEST-WAY(a, t, e, x, n, m)
for i ← 1 to n
    f[i,1] ← e[i] + a[i,1]
for j ← 2 to m
    for i ← 1 to n
    f[i, j] ← f[1, j-1] + t[1, j-1] + a[i, j]
            L[1, j] = 1
    for k ← 2 to n
            if f[i,j] > f[k, j-1] + t[2, j-1] + a[i, j]
        then f[i,j] ← f[k, j-1] + t[2, j-1] + a[i, j]
                        L[i, j] = k
            end if
```

n-Line Assembly: Dynamic Algorithm

$f^* \leftarrow f[1, m] + x[1]$

l* = 1

for k $\leftarrow$ 2 **to** n

  if f* > f[k, m] + x[k]

    then $f^* \leftarrow f[k, m] + x[k]$

    l* = k

Constructing the Fastest Way: n-Line

1. Print-Stations (l^*, m)
2. i ← l^*
3. print “line” i “, station” m
4. **for** j ← m **downto** 2
5. **do** i ← $l_i[j]$
6. print “line” i “, station” j - 1

Generalization: Cyclic Assembly Line Scheduling

Title: Moving policies in cyclic assembly line scheduling

Source: Theoretical Computer Science, Volume 351, Issue (February 2006)

Summary: Assembly line problem occurs in various kinds of production automation. In this paper, originality lies in the automated manufacturing of PC boards.

- In this case, the assembly line has to process number of identical work pieces in a cyclic fashion. In contrast to common variant of assembly line scheduling.
- Each station may process parts of several work-pieces at the same time, and parts of a work-piece may be processed by several stations at the same time.

Application: Multiprocessor Scheduling

- The assembly line problem is well known in the area of multiprocessor scheduling.
- In this problem, we are given a set of tasks to be executed by a system with n identical processors.
- Each task, Ti, requires a fixed, known time pi to execute.
- Tasks are indivisible, so that at most one processor may be executing a given task at any time
- They are un-interruptible, i.e., once assigned a task, may not leave it until task is complete.
- The precedence ordering restrictions between tasks may be represented by a tree or forest of trees

Conclusion

- Assembly line problem is discussed
- Generalization is made
- Mathematical Model of generalized problem is discribed
- Algorithm is proposed
- Applications are observed in various domains
- Some research issues are identified