# Solución de JSSP utilizando un Algoritmo Genético

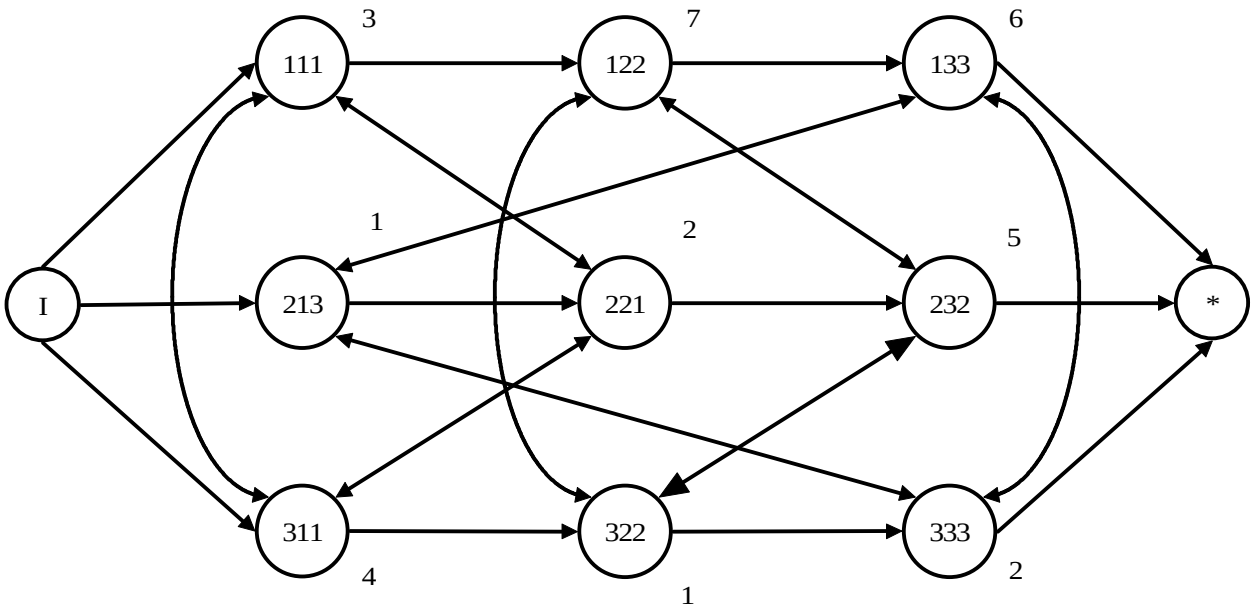
Arnulfo Martinez
Marco Antonio Crúz Chávez
Septiembre de 2000

- Introducción
- Descripción de JSSP
  - Representación por Grafo disyuntivo
  - Restricciones
- Algoritmo genético de Nakano y Yamada (NY)
  - GA Convencional JS
  - Algoritmo de Armonización (Población legal)
    - Armonización local
    - Algoritmo de scheduling
    - Armonización global
  - Convergencia de GA para problema de 10 x 10
- Modificación a GA de Nakano y Yamada
- Algoritmo de Giffler y Thompson (GT) para schedule activo
- Tipo de schedule en JS
  - Schedule semiactivo
  - Schedule activo

# Introduccion

- El problema de job shop se describe como sigue. Se tiene un taller que consiste de un conjunto de máquinas $m = \{M_1, M_2, \ldots, M_m\}$. Sobre estas máquinas un conjunto de trabajos $j = \{1,2,\ldots,n\}$, se necesita obtener un schedule. Cada máquina esta disponible al tiempo cero y puede procesar un solo trabajo en un instante de tiempo.

# Representación de JSSP por un grafo disyuntivo




Representación
de R. Conway G=(N,C,D)

# Restricciones de JSSP

El JSSP a resolver tiene N trabajos que se procesan sobre M maquinas y se asume lo suguiente:

- Una maquina puede procesar solamente un trabajo en un solo instante
- Al procesamiento de un trabajo en una maquina se le llama operación
- Una operación no se puede interrumpir
- Un trabajo consiste de M operaciones
- A la secuencia de operaciones dentro de un trabajo se le llama restricciones de precedencia y junto con los tiempos de procesamiento para las operaciones son datos
- A la secuencia de operaciones sobre una maquina se le llaman restricciones de capacidad de recursos, y son datos desconocidos
- A las secuencias propuestas para un problema se le llaman representación simbólica
- A la representación simbolica factible se le llama schedule

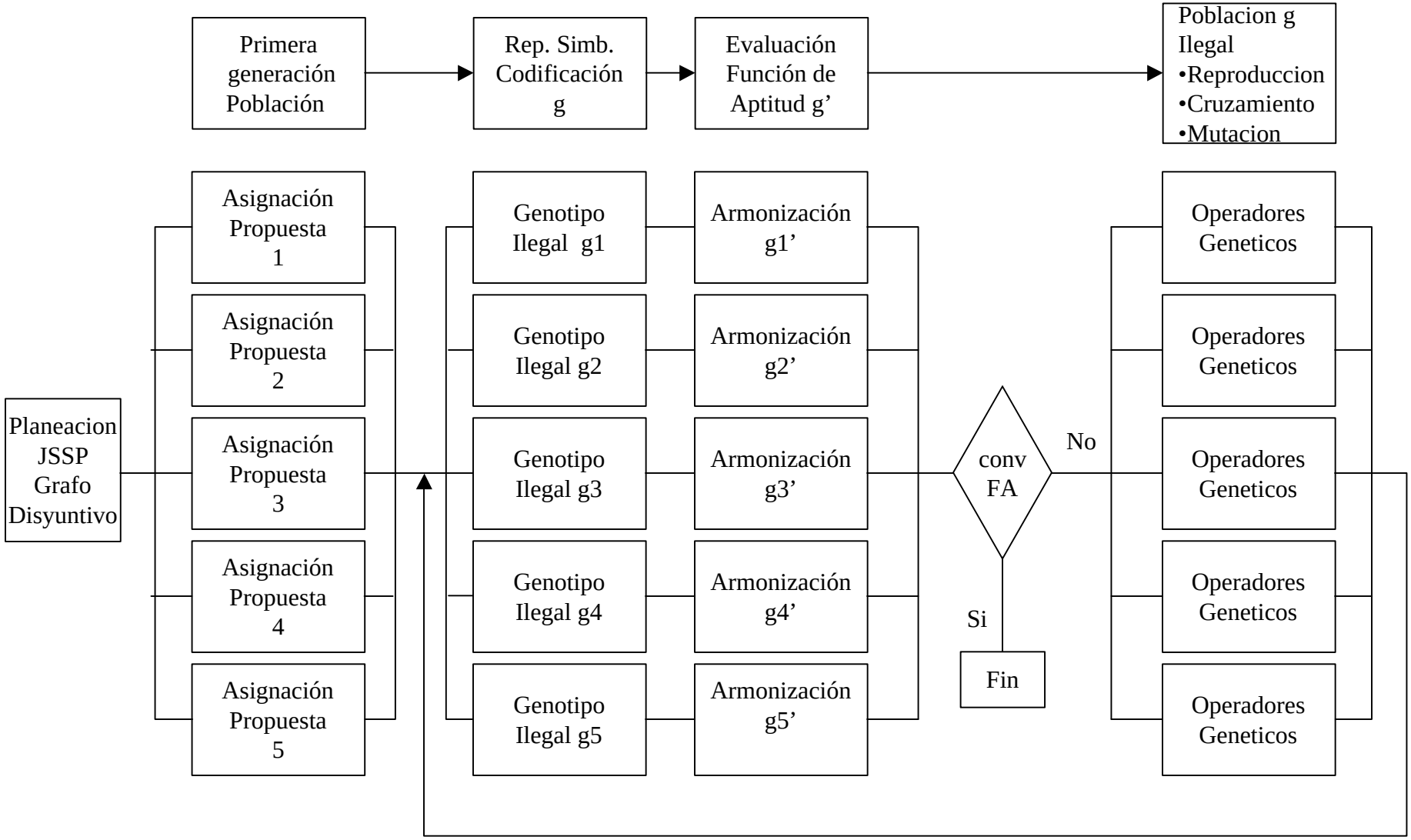
# Objetivo en JSP

Encontrar un schedule que minimice el lapso de tiempo total (Makespan), en el que sean ejecutados todos los trabajos.

# GA convencional para el problema de Job Shop NY

```
Void GA ( )
{
      Plan_jobshop( );
      P=inicializa_población( );
      P=obtener_población_legal( );
      Converge=evaluar_aptitud(P);
      While (!converge){
          P=reproducción_en(P);
          P=cruzamiento_con(P);
          P=mutación_con(P);
          P=obtener_población_legal( );
          converge=evaluar_aptitud(P);
    }
}
```

# Solución de JSSP por medio de un GA



# Representación de un genotipo

## Representación simbólica

| | | | |
|---|---|---|---|
| J1 | 1 | 2 | 3 |
| J2 | 1 | 3 | 2 |
| J3 | 2 | 1 | 3 |

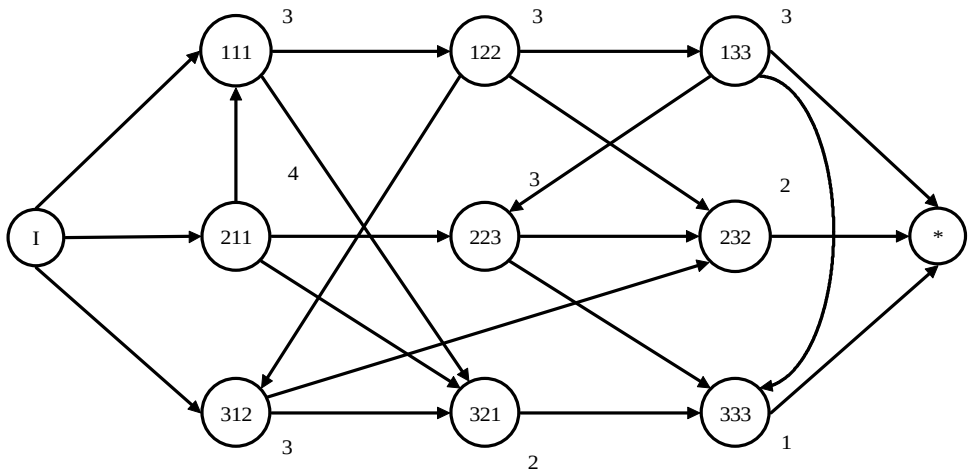
Secuencia de máquinas

| | | | |
|---|---|---|---|
| J1 | 3 | 3 | 3 |
| J2 | 4 | 3 | 2 |
| J3 | 3 | 2 | 1 |

Tiempo de procesamiento

| | | | |
|---|---|---|---|
| M1 | 2 | 1 | 3 |
| M2 | 1 | 3 | 2 |
| M3 | 1 | 2 | 3 |

Secuencia de trabajos

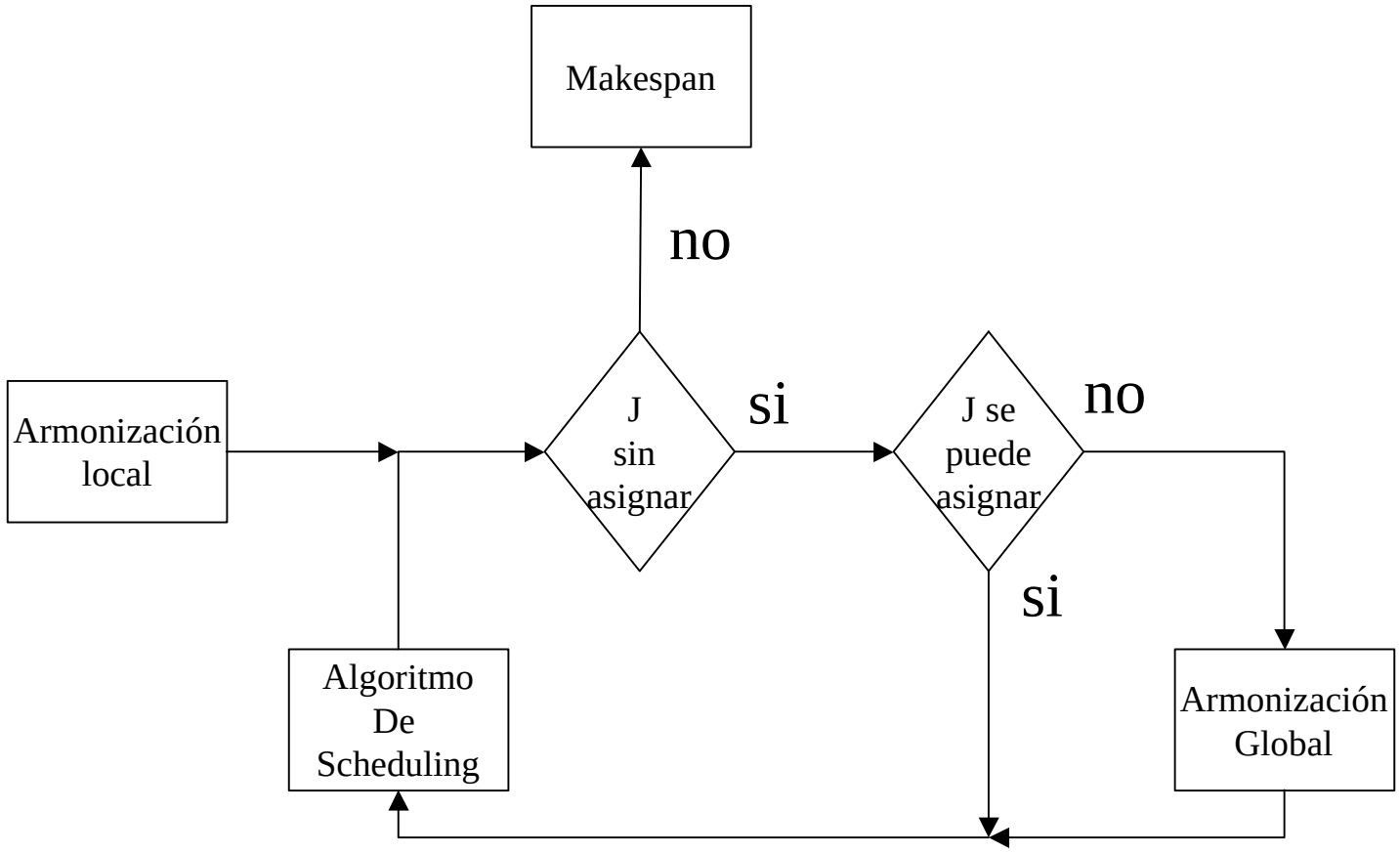



Representacion binaria

Genotipo g
J1<J2 : 011
J1<J3 : 111
J2<J3 : 110

Tamaño del genotipo
Num. par de trabajos = N(N-1)/2 = 3
Num. genes = MN(N-1)/2 = 9

# Algoritmo de Armonización: crea un genotipo g' (legal). Propuesta de NY.



# Algoritmo de scheduling

Datos:

- Representación simbólica
- Tiempos de procesamiento

Se utiliza la siguiente notación:

jnext(j): siguiente operación a ser ejecutada dentro del trabajo j

jnext(j).máquina: máquina a ejecutar jnext(j)

mnext(m): siguiente operación a ser ejecutada sobre la máquina m

El trabajo se podra asignar si:

jnext(j)=mnext(jnext(j).maquina)

Si la relación anterior no concuerda entonces se aplica armonización global

Nota:

jnext(j): se obtiene de las secuencias de máquinas

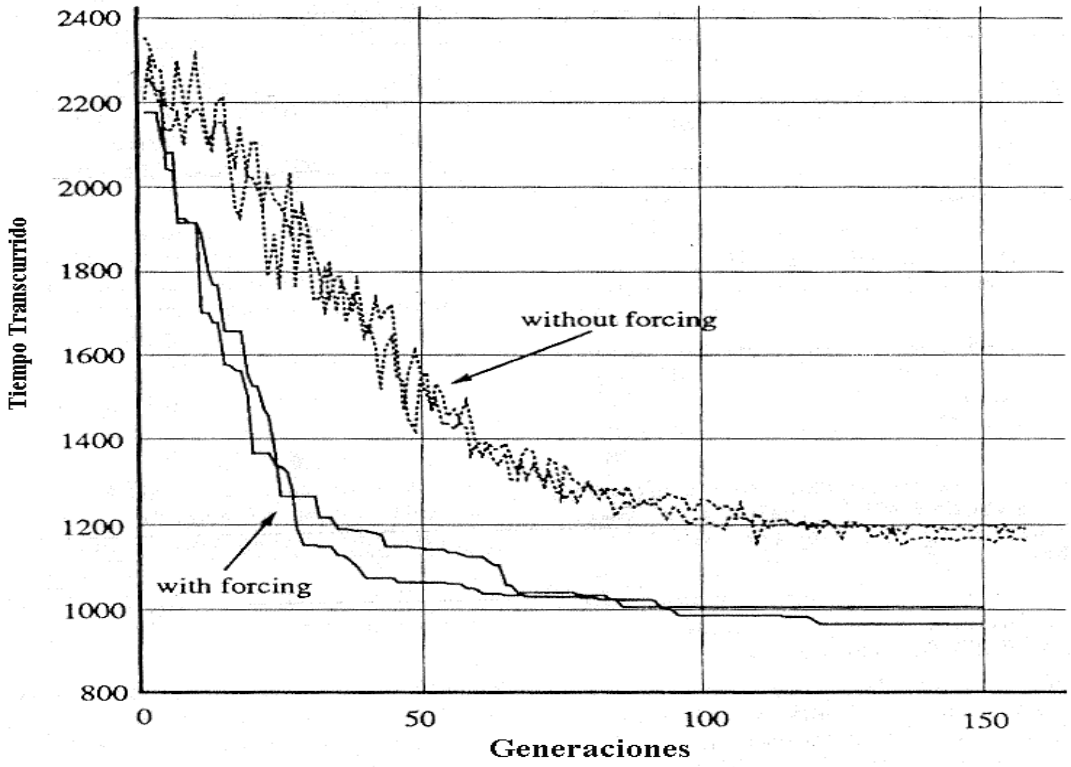
mnext(jnext(j).maquina): se obtiene de las secuencias de trabajos por máquina

# Armonización global

1. Cada trabajo j con turno de asignación se revisa la distancia entre prioridades de asignación de jnext(j) y mnext(jnext(j).maquina)
2. Se selecciona el trabajo con la menor distancia
3. Se permuta la secuencia como mnext(jmax(j),máquina) jnext(j)
4. Regreso al algoritmo de scheduling

# Convergencia de GA para problema de 10 x 10

| Papers | Algorithm | 6 × 6 prob. | 10 × 10 prob. | 20 × 5 prob. |
|---|---|---|---|---|
| Balas1969 | BAB | 55 | 1177 | 1231 |
| McMahon1975 | BAB | 55 | 972 | 1165 |
| Barker1985 | BAB | 55 | 960 | 1303 |
| Carlier1989 | BAB | 55 | 930 | 1165 |
| Nakano1991 | GA | 55 | 965 | 1215 |




**Convergencia de Algoritmos Geneticos para un problema de 10 x 10**

# Modificación a GA de Nakano y Yamada

```
Void GA ( )
{
      Plan_jobshop( );
      P=poblacion_activa( );
      Converge=evaluar_aptitud(P);
      While (!converge){
          P=reproduccion_en(P);
          P=cruzamiento_con(P);
          P=mutacion_en(P);
          P=obtener_poblacionlegal( );
          converge=evaluar_aptitud(P);
     }
}
```

# Tipo de schedule en JS

- En una secuencia propuesta de operaciones en JS se tiene un número infinito de schedules
- En una secuencia propuesta de operaciones se tiene un solo schedule semiactivo
- Schedule semiactivo: ninguna operación se puede mover por un cambio limitado a la izquierda. Un schedule semiactivo se obtiene realizando un limite de cambios a la izquierda sin alterar el orden de las operaciones

# Tipo de schedule en JS

- El número de schedules semiactivos se obtiene por:

$$(n!)^m$$

$n =$ número de trabajos

$m =$ número de máquinas

Sólo para operaciones simétricas:

num operaciones por máquina=n

# Ejemplo:

Para un problema de 3 trabajos y 3 maquinas

$$(3!)^3 = 216 \text{ schedules semiactivos}$$

# Schedule activo

- Ninguna operación se puede mover por un cambio a la izquierda sobre cualquier operación

- Un schedule activo se obtiene realizando cambios a la izquierda de las operaciones, pudiendo alterar el orden de estas pero no incrementando los tiempos de inicio de cualquier otra

# Ejemplo

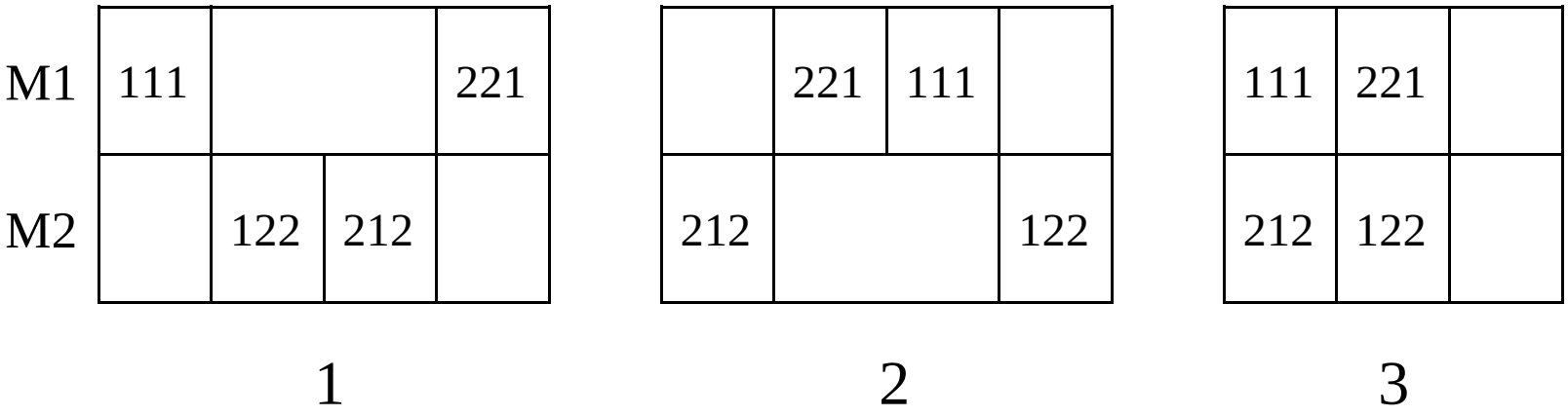



Diagrama de Gantt de 3 schedules semiactivos de un problema De 2x2

# Algoritmo GT para obtener un Schedule Activo

- D es el conjunto de todas las operaciones que inician mas temprano y no asignadas todavia, y Ojr es la operación con el minimo EC en D

$$Ojr = \min\{O \in D \mid EC(0)\}$$

- Asumir que i-1 operaciones han sido asignadas sobre Mr. Un conjunto de operaciones en conflicto $C = [Mr, i]$ se define como

$$C[Mr, i] = \{O_{Kr} \in D \mid O_{Kr} \text{ sobre } Mr, ES(O_{Kr}) < EC(O_{jr})\}$$

# Algoritmo GT para obtener un Schedule Activo

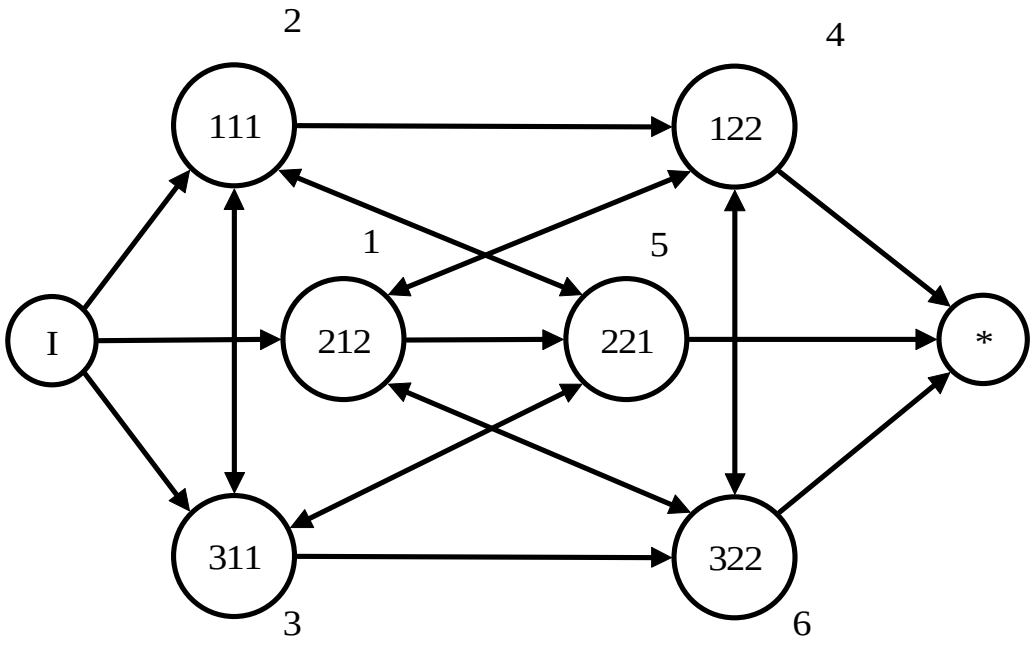
- Seleccionar una operación $O \in C[Mr, i]$
- Asignar O como la ith operacion sobre Mr con su tiempo de termino igual a EC(O)

Significado:

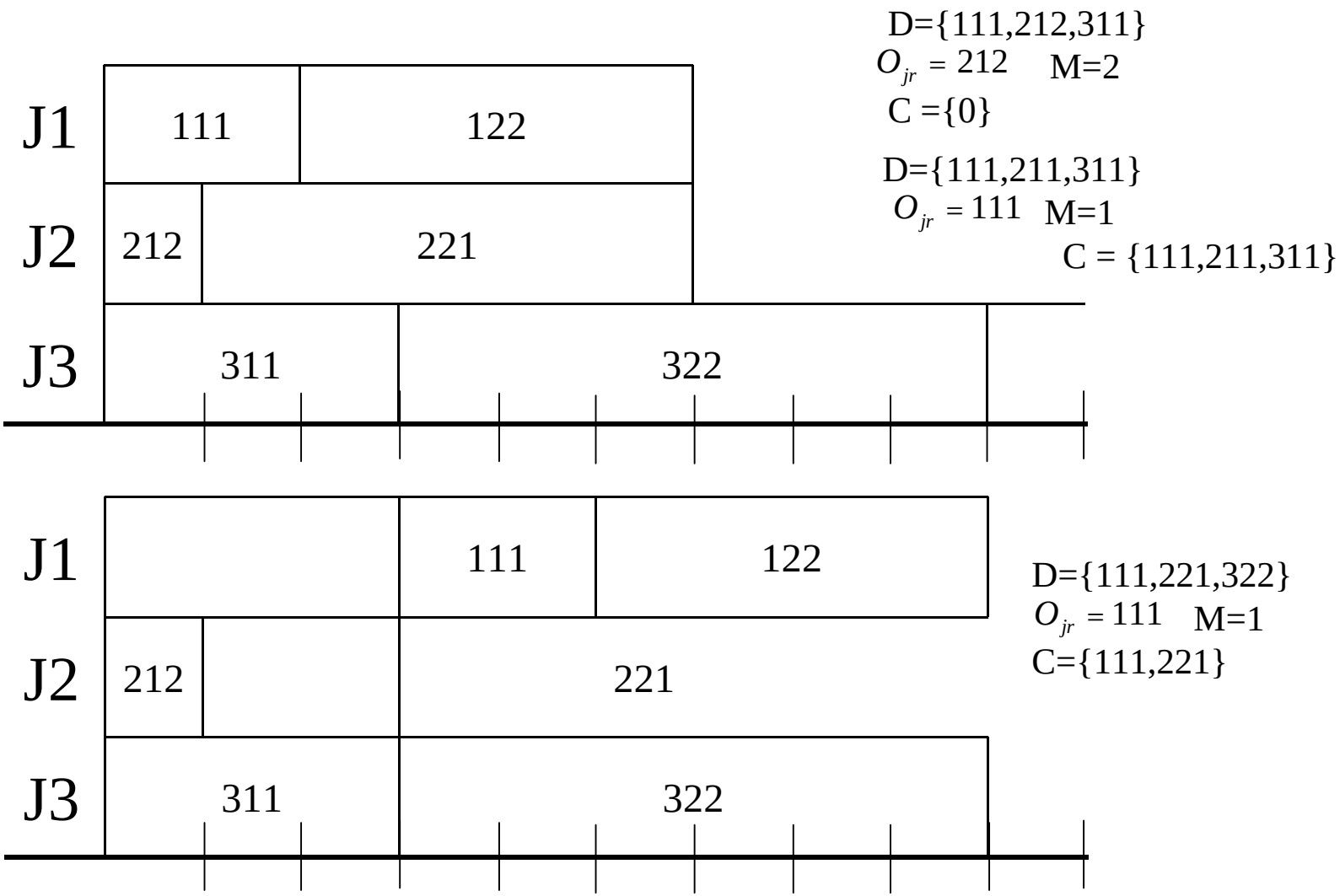
EC=tiempo de termino mas temprano

ES=tiempo de inicio mas temprano

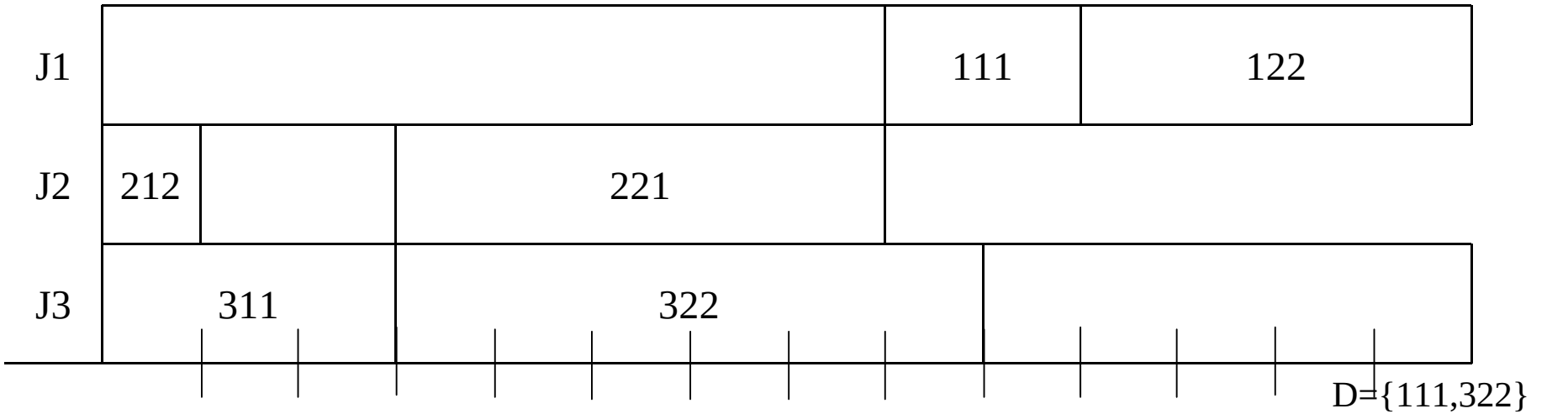
# Aplicacion de algoritmo GT a un JS de 3x3



# Aplicacion de algoritmo GT

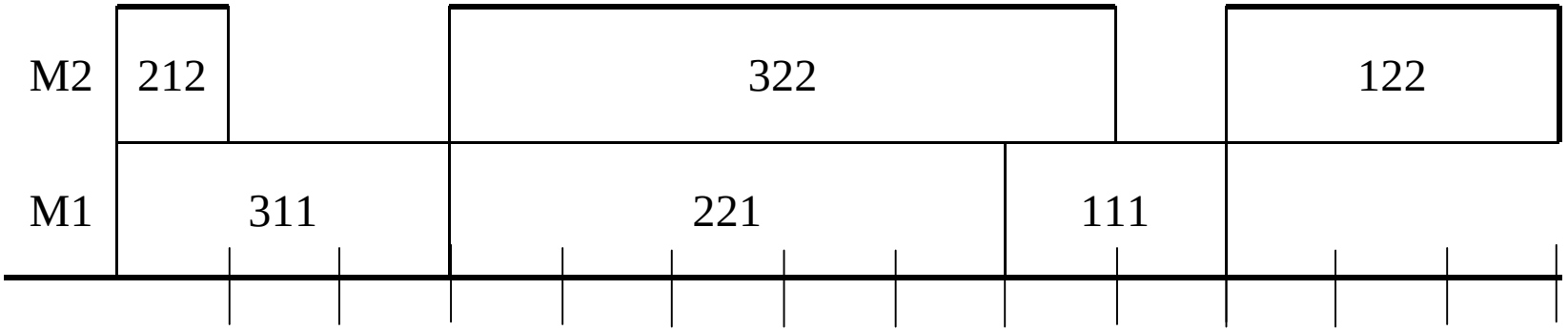


# Aplicacion de algoritmo GT




D={111,322}

$O_{jr}$ = 322 M=2



# Schedule con algoritmo GT