Algoritmo no determínístico para el problema de Job Shop scheduling codificado en SAT.

Codificacion SAT reducida

Caracteristicas Problemas SAT

- Problemas Teoricos

- Forma aleatoria
- Muchas restricciones
- Pocas soluciones
- Malos casos de prueba para problemas practicos

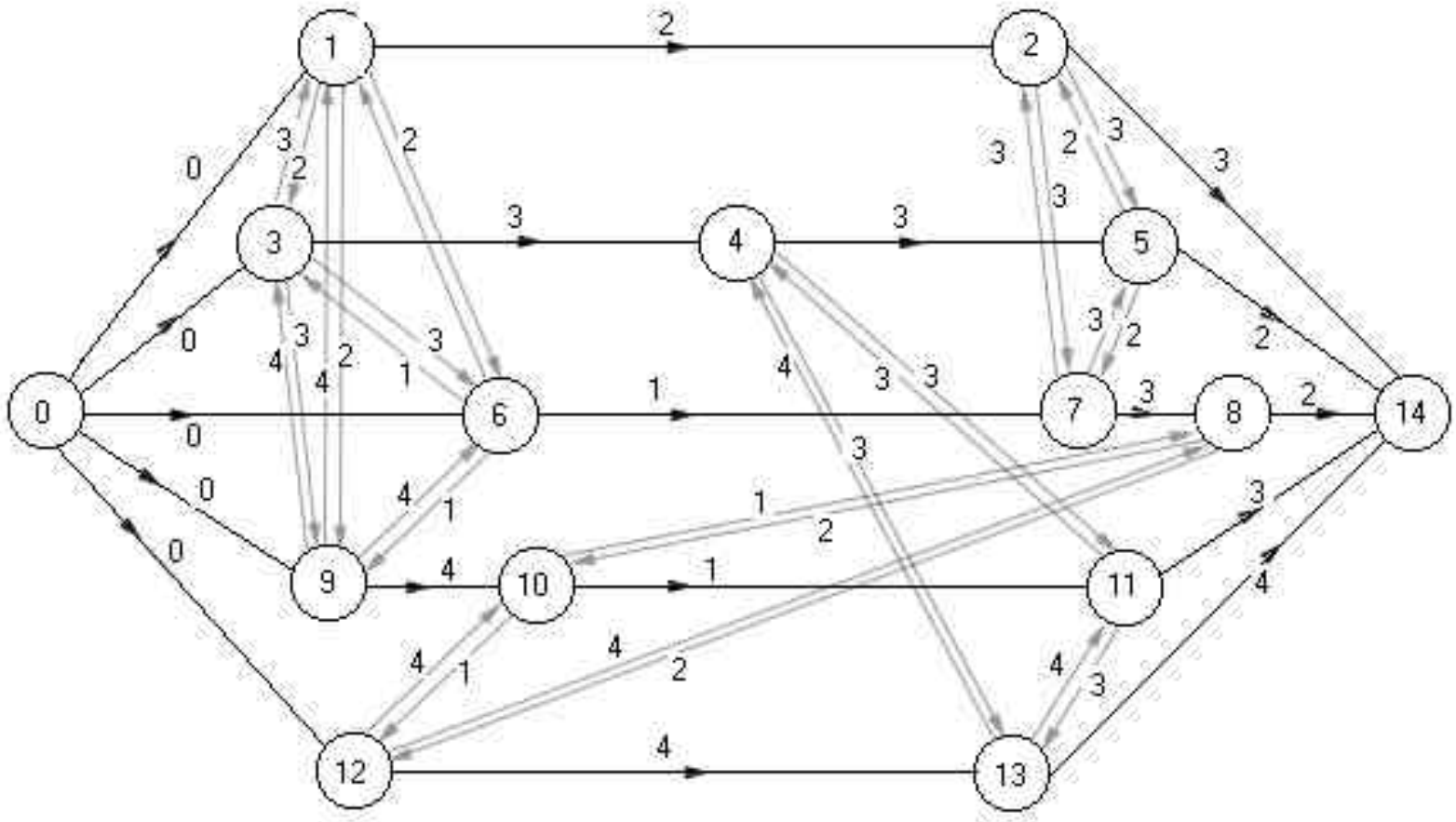
- Problemas practicos

- Gran número de variables
- Pocas restricciones
- Número grande de soluciones.

Job Shop

- Maquinas que realizan tareas en un taller.
- Comparticion de recursos para realizar diversas tareas.
- Una maquina (recurzo), tiene varias herramientas y puede realizr mas de una tarea
- Problema: Optimizar los tiempos en el uso de recursos.

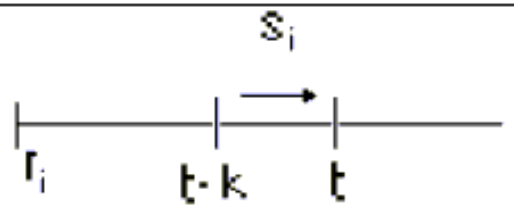
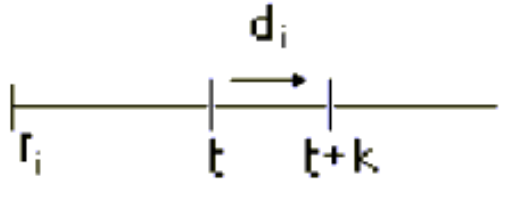
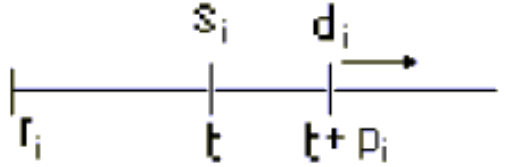
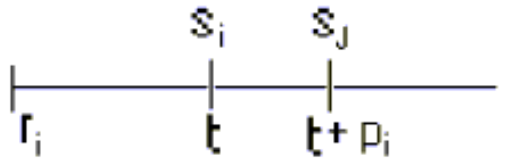
Representacion de Job Shop por Conway, G=(N,C,D)



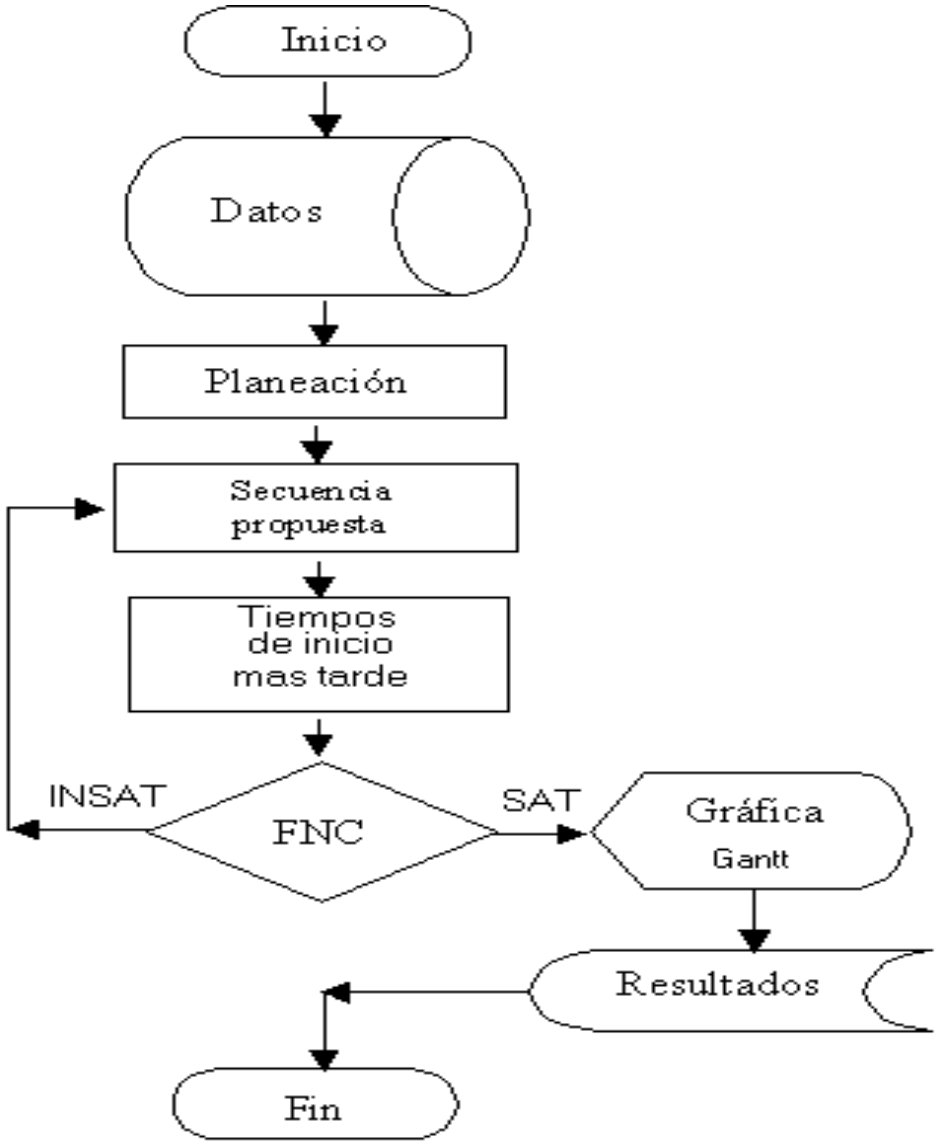
Codificacion CSP a SAT

RESTRICCIONES Job shop scheduling		CODIFICACIÓN A SAT	SIGNIFICADO SAT	Número de Restriccion/Clausula
Precedencia	$s_i + p_i \leq s_j$	$pr_{i,j} = T$	La operación i precede a j.	(1)
Capacidad de recursos	$(s_i + p_i \leq s_j) \vee (s_j + p_j \leq s_i)$	$pr_{i,j} \vee pr_{j,i} = T$	La operación i precede a j o la operación j precede a i.	(2)
Tiempo de inicio de operación.	$s_i \geq r_i$	$\overline{sa_{i,r_i}} = T$	La operación i inicia al tiempo r_i o más tarde.	(3)
Tiempo máximo de duración de la operación.	$s_i + p_i \leq d_i$	$\overline{eb_{i,d_i}} = T$	La operación i termina máximo al tiempo d_i.	(4)

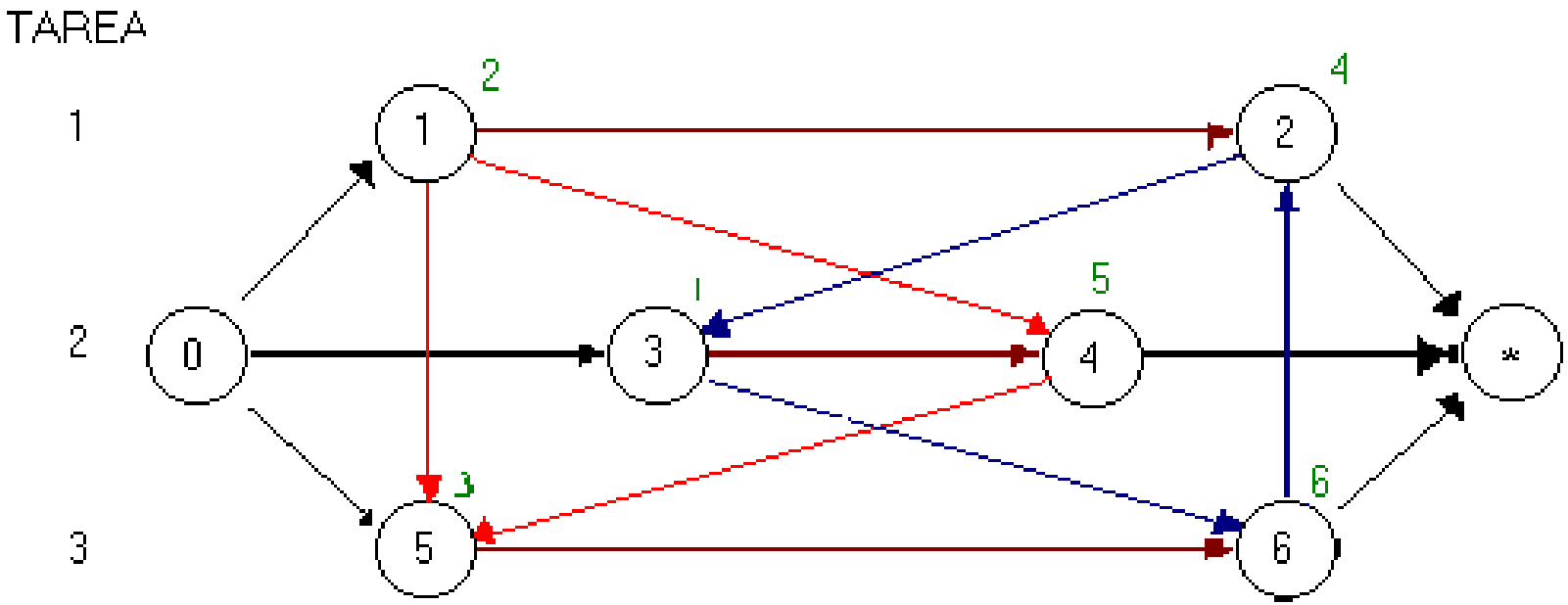
Coherencias y su interpretación

$sa_{i,t} \rightarrow sa_{i,t-k}$	
Si la operación i inicia a o después del tiempo t entonces puede iniciar a o después del tiempo k-1. Donde $0 \leq k \leq t - r_i$	
$eb_{i,t} \rightarrow eb_{i,t+k}$	
Si la operación i termina al tiempo t entonces puede terminar al tiempo t+k. Donde $k \geq 1$.	
$sa_{i,t} \rightarrow \neg eb_{i,t+p_i-k}$	
Si la operación i inicia a o después del tiempo t entonces no puede terminar antes del tiempo $t + p_i$	
$sa_{i,t} \wedge pr_{i,j} \rightarrow sa_{j,t+p_i}$	
Si la operación i inicia a o después del tiempo t y además i precede a j entonces la operación j inicia a o después del tiempo $t + p_i$	

Algoritmo de Solución para job shop scheduling codificado en SAT



Tiempos de inicio mas tardíos para un problema de 2 x 2.

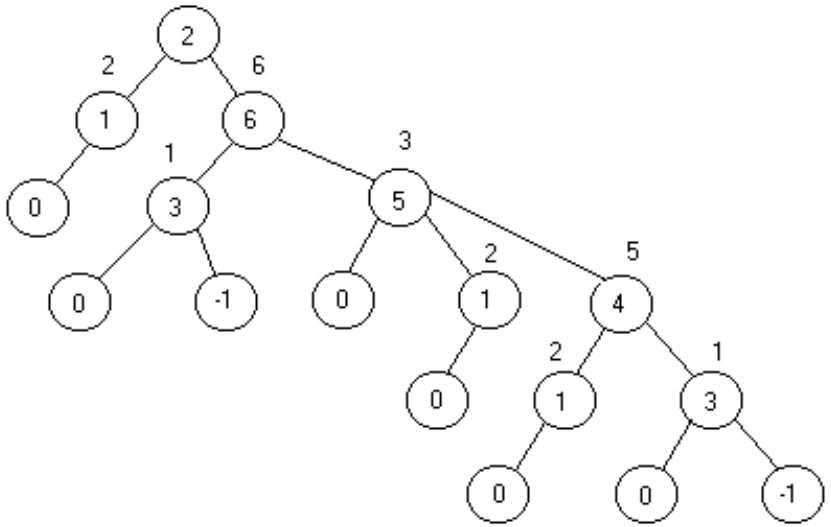


FNC para SAT

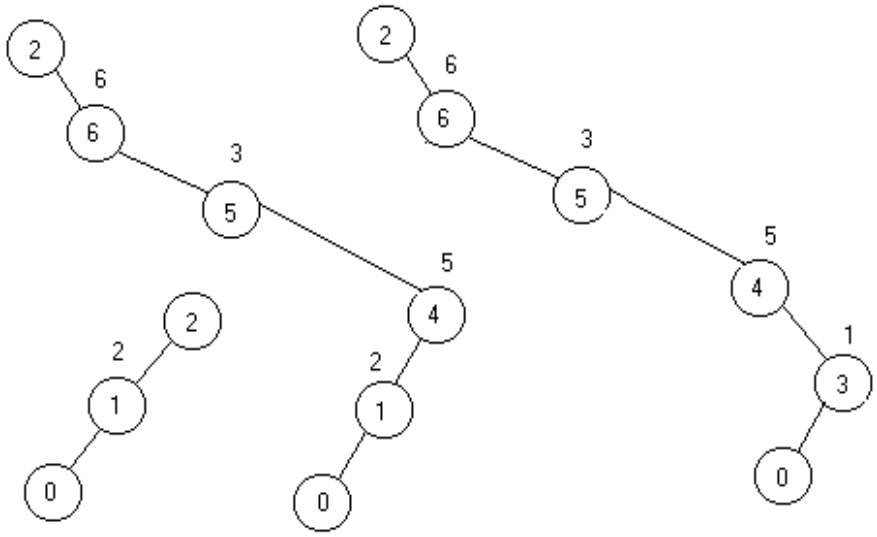
CONDICIONES COHERENTES Para Job shop scheduling en todos sus intervalos de tiempos relevantes [r_i a d_i].	SIGNIFICADO
$\neg sa_{i,t} \vee sa_{i,t-1}$	No inicia la operación i a o después del tiempo t o bien inicia al tiempo t-1.
$\neg eb_{i,t} \vee eb_{i,t+1}$	La operación i no termina al tiempo t o bien termina al tiempo t+1.
$\neg sa_{i,t} \vee \neg eb_{i,t+pi-1}$	La operación i no inicia a o después del tiempo t o bien no termina antes del tiempo t + p_i.
$\left(\neg sa_{i,t} \vee \neg pr_{i,j}\right) \vee sa_{j,t+pi}$	La operación i no inicia a o después del tiempo t o bien no precede a j o j inicia a o después del tiempo t + p_i.

TRANSFORMACIÓN DE LAS COHERENCIAS POR LAS LEYES DE LA LÓGICA PROPOSICIONAL.

Tiempos de inicio mas tardíos



Arbol completo de rutas para la operación 2



Algunas rutas en el arbol

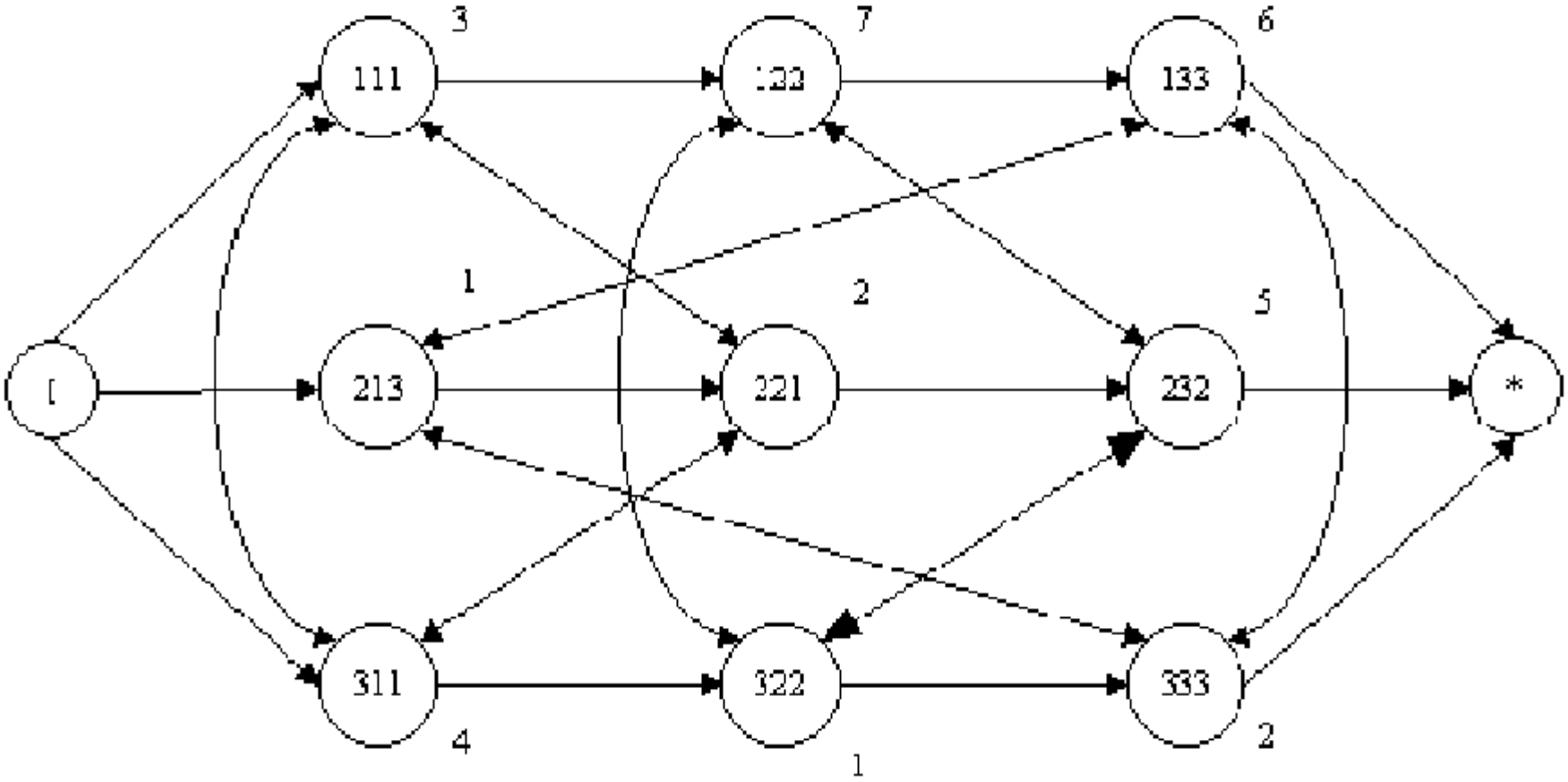
Mejoras en el algoritmo

- 1.- Evitar ciclos en máquinas, para una mejor secuencia propuesta.
- 2.- Mejorar la solución obtenida encontrando un schedule Activo.

Secuencias sin ciclos en máquinas.

- 1.- Se realiza en cada máquina.
- 2.- Permite una mejor secuencia propuesta, con mayor posibilidad de ser satisfactible.
- 3.- Asigna prioridades de ejecución a trabajos en cada máquina (Arcos salientes en cada nodo).
- 4.- La complejidad del algoritmo para encontrar tiempos de inicio tardíos se reduce.

Ejemplo de una secuencia sin ciclos para el siguiente problema



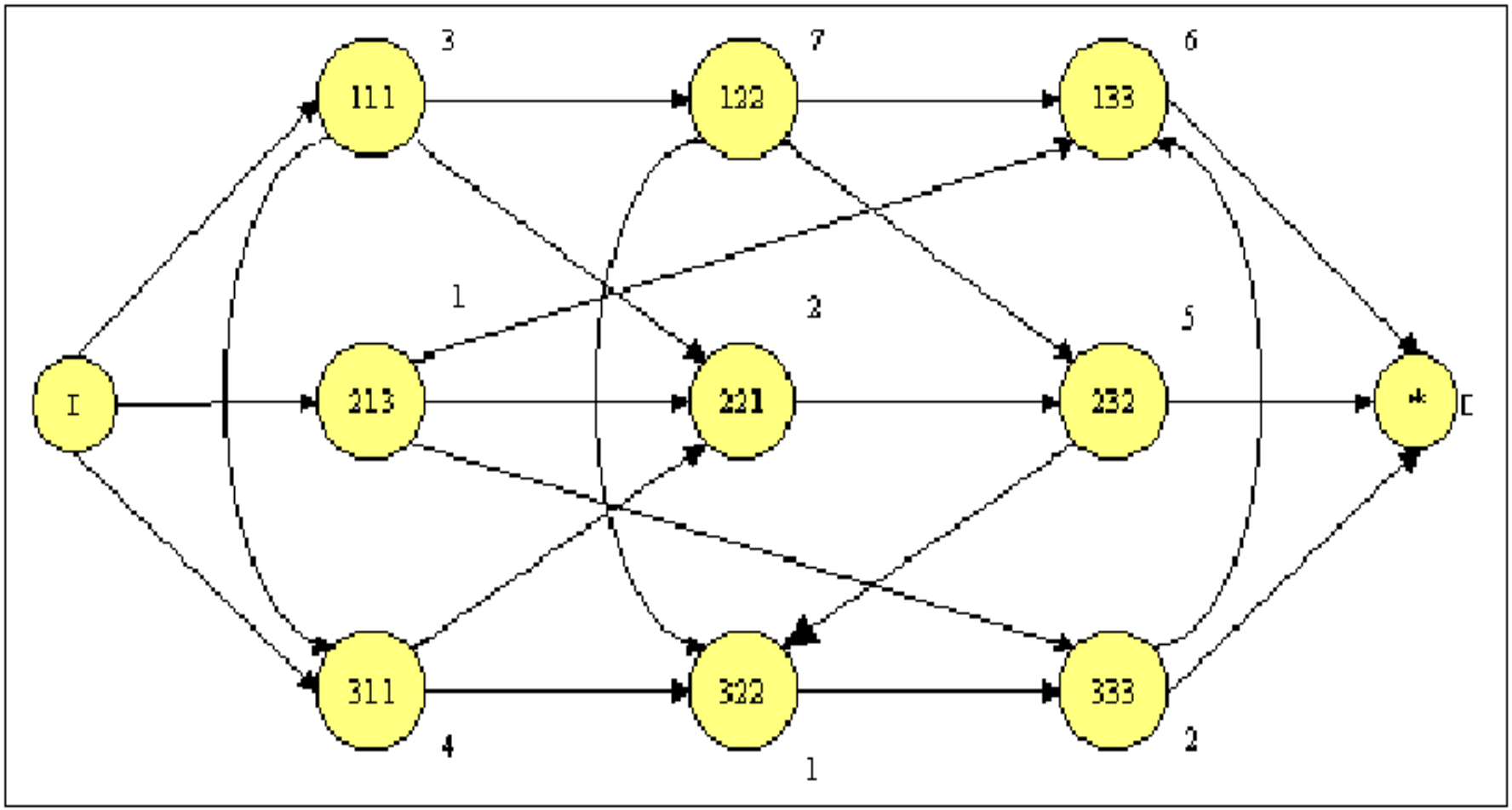
Evitando ciclos

Maquina 1	
Trabajo elegido de forma aleatoria	Prioridad de ejecución
2	0
3	1
1	2

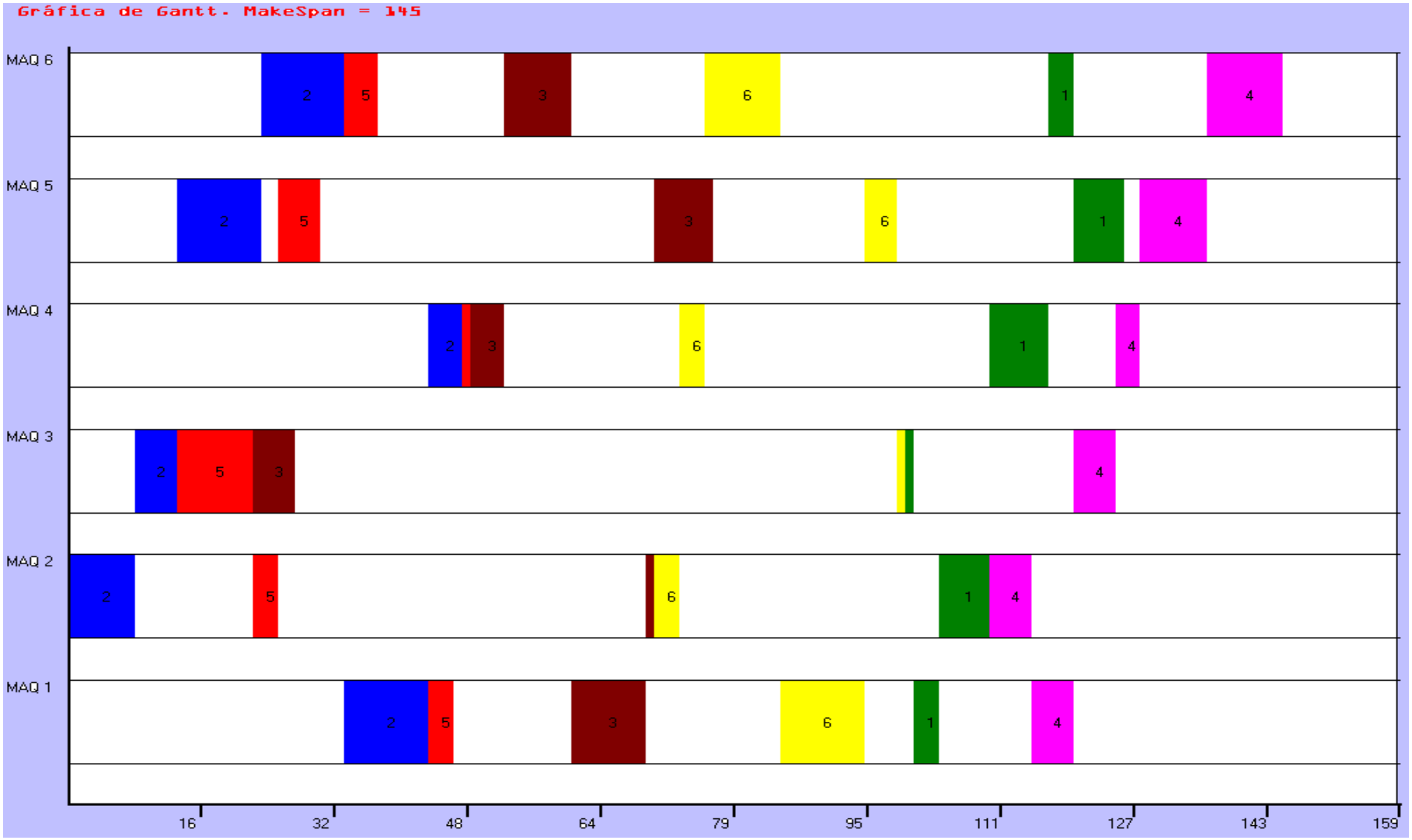
Maquina 2	
Trabajo elegido de forma aleatoria	Prioridad de ejecución
3	0
2	1
1	2

Maquina 3	
Trabajo elegido de forma aleatoria	Prioridad de ejecución
1	0
3	1
2	2

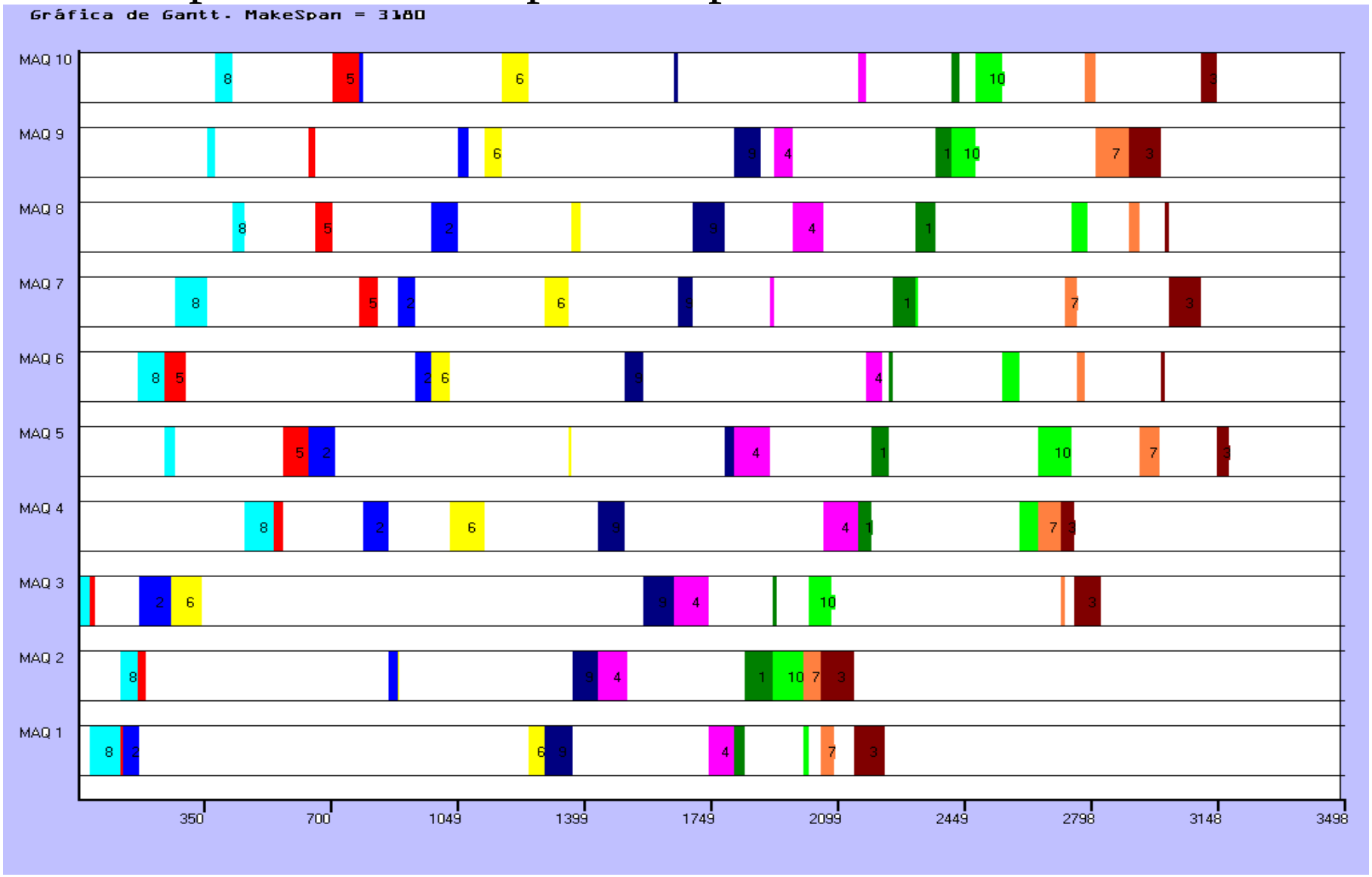
Resultado de una secuencia sin ciclos



Resultado con tiempos de inicio mas tardíos en una primer corrida para un problema de 6 x 6 (Benchmark de Muth y thompson)



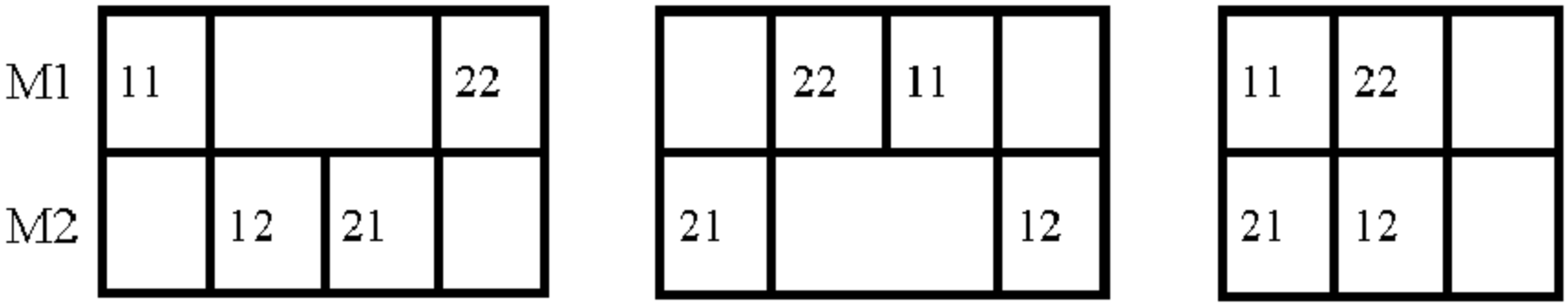
Resultado con tiempos de inicio mas tardíos en una primer corrida para un problema de 10 x 10



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.

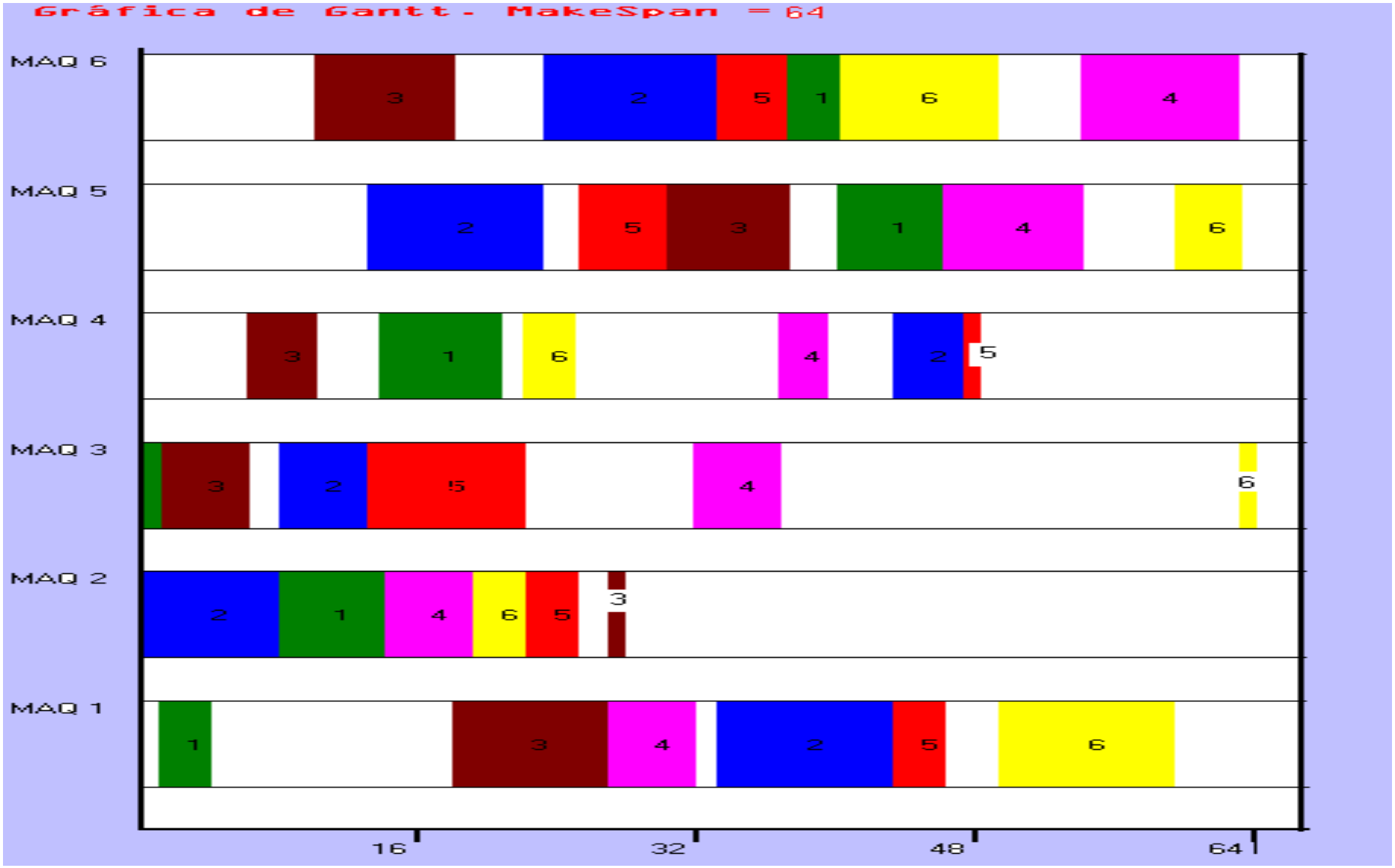
Diagrama de Gantt de 3 schedules semiactivos de un problema De 2x2



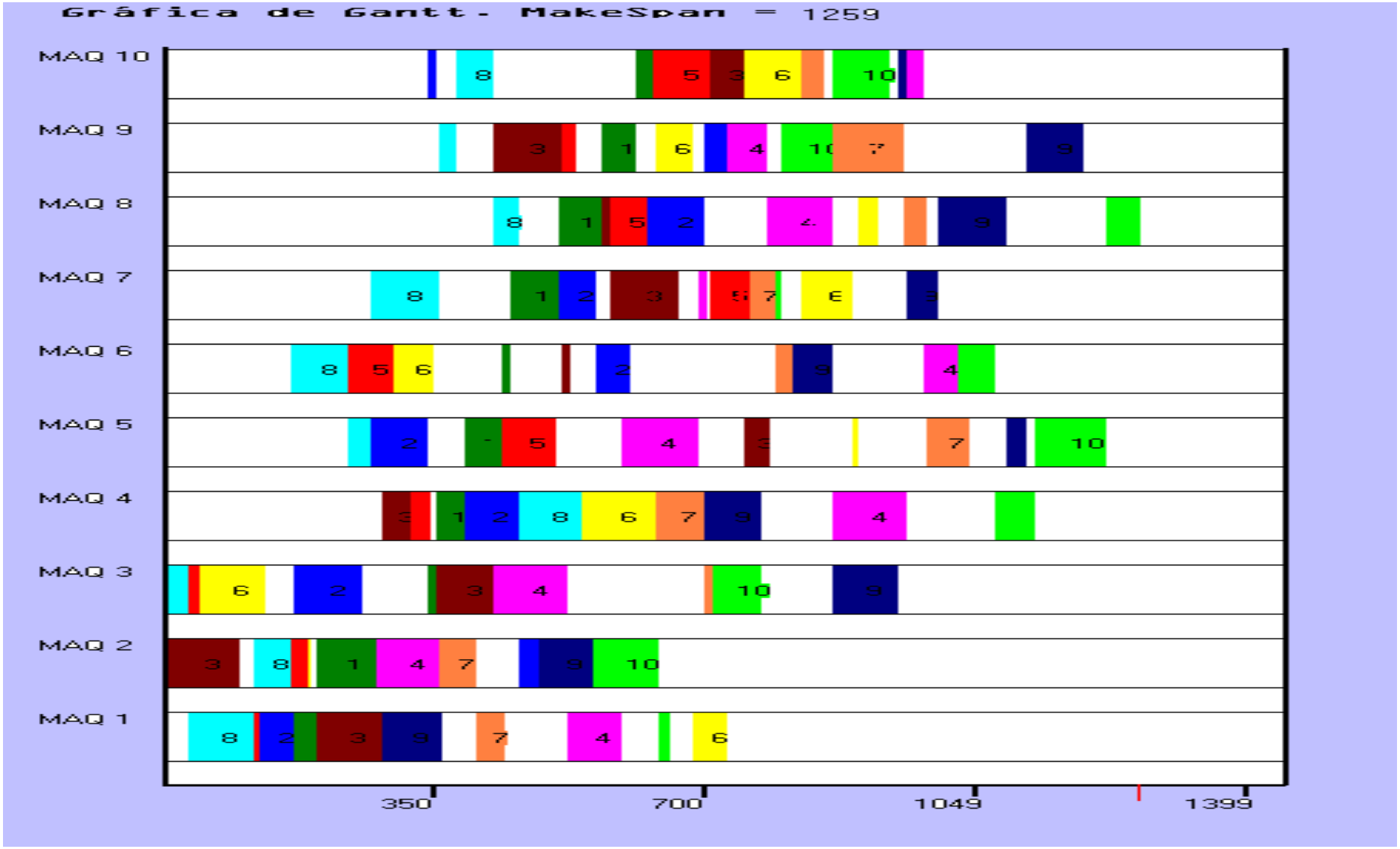
Algoritmo propuesto para obtener schedules activos.

1.- Elegir el siguiente trabajo J, si no hay J ir a 7
2.- Elegir la siguiente operación O_{ij} de J en M_r según precedencia, si no hay O_{ij} ir a 1.
3.- Generar listas de trabajos en M_r con datos: S, C, P. En orden de S ascendente.
4.- Buscar tiempo de ocio en M_r entre par de trabajos k, l, con
restricciones: $t_{ocio} >= P_{ij}$, $t_{ocio} = S_l - C_k$, $S_l < S_{ij}$, k -> l
5.- si $t_{ocio} >= P_{ij}$ mover operación O_{ij} a $S_{ij} = C_k$
6.- ir a 2
7.- Fin

Resultado con tiempos de inicio mas tardíos en una primer corrida para un problema de 6 x 6, después de aplicar el algoritmo para obtener un schedule activo. (Benchmark de Muth y thompson)



Resultado con tiempos de inicio mas tardíos en una primer corrida para un problema de 10 x 10, después de aplicar el algoritmo para obtener un schedule activo. (Benchmark de Muth y thompson)



Optimos reportados por Carlier y Pinson en 1989.

- El óptimo reportado en el problema de 6x6 es de 55. En una primera aproximación por SAT se obtiene 65. Existe un porcentaje de desviación de 18%.
- El óptimo reportado en el problema de 10x10 es de 930. En una primera aproximación por SAT se obtiene 1259. Existe un porcentaje de desviación de 35%.