4. MÉTODO EXACTO DE RESOLUCIÓN Y ALGORITMO TABÚ

En este apartado vamos a realizar una detallada descripción de las metodologías que posteriormente utilizaremos para contrastar el nivel de calidad alcanzado por la metaheurística GRASP generada en este proyecto.

4.1. Algoritmo exacto.

En el estudio de métodos para la resolución de este tipo de problemas, los modelos exactos se han mostrado aceptables únicamente para instancias pequeñas, ya que son necesarios algoritmos complejos y de alto nivel computacional. Sin embargo, es necesario disponer de un conjunto de resultados óptimos a la hora de poder estudiar el nivel de calidad y robustez alcanzado con la metaheurística GRASP aquí generada.

El algoritmo exacto que vamos a utilizar para la resolución del problema de POC considerado en este proyecto, fue desarrollado por Arkin y Silverberg en 1987. Se trata de un modelo cuya construcción consta de dos fases claramente diferenciadas:

En la primera de ellas, se transforma el problema original de partida en un grafo dirigido y acíclico. Una vez definido éste, se pasa a una segunda fase en la que se resuelve obteniendo la ruta máxima que va desde el nodo de salida S hasta el de llegada T.

Para ver de una manera más clara la forma de trabajo del algoritmo vamos a aplicarlo a un ejemplo sencillo como el siguiente:

Supongamos que se quiere procesar un conjunto de cuatro trabajos { J_1, J_2, J_3, J_4 } y se dispone para ello de dos máquinas {M_1, M_2}. Existen, además, tres clases de trabajos (A = 3) y dos clases de máquinas (C = 2) de forma que cada trabajo podrá ser realizado por una de las máquinas en función de la compatibilidad existente entre los grupos a los que pertenecen. Estas coincidencias se reflejan en la matriz de compatibilidades, que tiene la forma siguiente:

L_{AxB}	c_1	c_2
a_1	1	0
a_2	1	1
a_3	0	1

Tabla 4.1: Matriz de compatibilidades.

Según lo anterior tenemos que:

- Los trabajos del tipo 1 sólo podrán ser procesados por la máquina 1.
- Los trabajos del tipo 2 podrán ser procesados de forma indiferente por las máquinas 1 y 2.
- Los trabajos del tipo 3 sólo podrán ser procesados por la máquina 2.

El resto de las características de cada uno de los trabajos se describen en la siguiente tabla:

Trabajo	Peso (w_i)	Inicio (s_i)	Fin (f_i)	Tipo
J_1	7	0	4	1
J_2	8	1	12	2
J_3	4	2	6	3
J_4	3	8	11	1

Tabla 4.2: Características de los trabajos.

La situación en el tiempo de cada uno de los trabajos se puede ver de forma más clara en la siguiente figura:

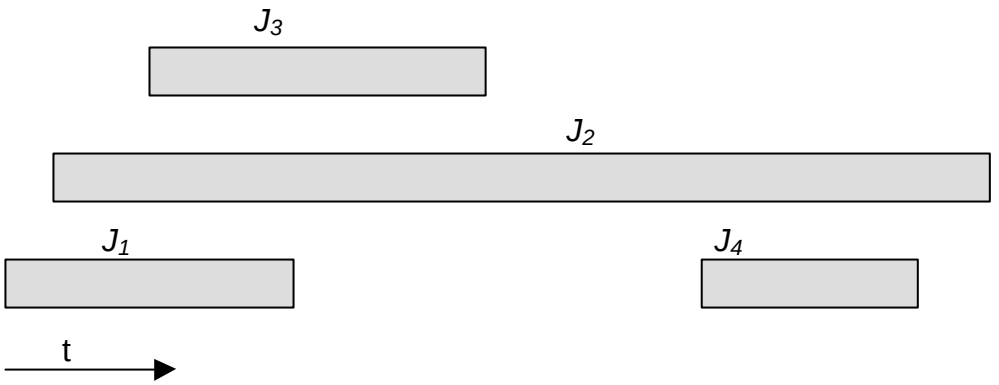

Figura 4.1: Diagrama temporal de trabajos.

Una vez definidas las características del problema a resolver, pasamos a la aplicación de la primera fase del algoritmo exacto de resolución.

Si definimos como “eventos” cada uno de los instantes en que se inicia o finaliza un trabajo, tenemos que, al haber cuatro trabajos en nuestro ejemplo, existen ocho eventos, dos por cada uno de los trabajos. Creamos entonces un nodo de salida S que representa el instante de inicio del primer trabajo y un nodo de llegada T que representa el instante de finalización del último de los trabajos. Ambos serán los eventos de salida y llegada respectivamente. Entre estos nodos vamos a ir colocando otros que irán representando las situaciones admisibles existentes en cada uno de los momentos del proceso de planificación.

Comenzamos colocando cinco nodos entre los eventos de salida y entrada que representan las cinco posibilidades que tenemos de procesar los trabajos y desde el nodo S hacemos partir un arco hasta el evento de comienzo de cada uno de estos nodos. Los arcos salientes representan las opciones de asignación que aparecen a partir de ese instante y están representados en la figura 4.2:

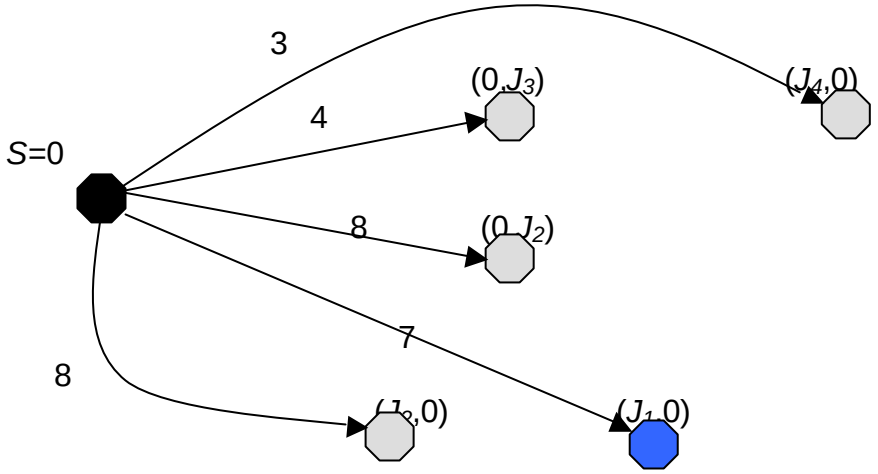

Figura 4.2: Primera fase de construcción.

En cada uno de los nodos se define el estado de las máquinas del sistema mediante el vector (M_1, M_2), de manera que, si en un instante una máquina está desocupada colocamos un cero en su posición correspondiente, mientras que si la máquina está ocupada colocamos el trabajo que en ese momento está procesando.

Luego, siempre y cuando una máquina esté libre, para cada evento aparecerá un nodo que se corresponde con una combinación que puede llevarse a cabo en ese instante, es decir, a partir de cada nodo saldrán un conjunto de arcos cuyo número será igual al de posibilidades de asignaciones que se tengan en el instante siguiente. Así, por ejemplo, para el nodo marcado en azul en la figura en el que M_1 está procesando el trabajo J_1 se nos plantean las siguientes posibilidades:

Figura 4.3: Estados admisibles a partir del nodo (J_1, 0).

- Que la máquina 2 comience a procesar el trabajo J_2 con lo que se llegaría al nodo (J_1, J_2). (Opción marcada en rojo).
- Que la máquina 2 comience a procesar el trabajo J_3 con lo que se llegaría al nodo (J_1, J_3). (Opción marcada en amarillo).
- Que la máquina 1 termine de procesar el trabajo J_1 sin que la máquina 2 comience a procesar trabajo alguno, con lo que se llegaría al nodo (0, 0). (Opción marcada en verde).

Teniendo en cuenta estas posibilidades, el grafo se transformaría en la manera siguiente:

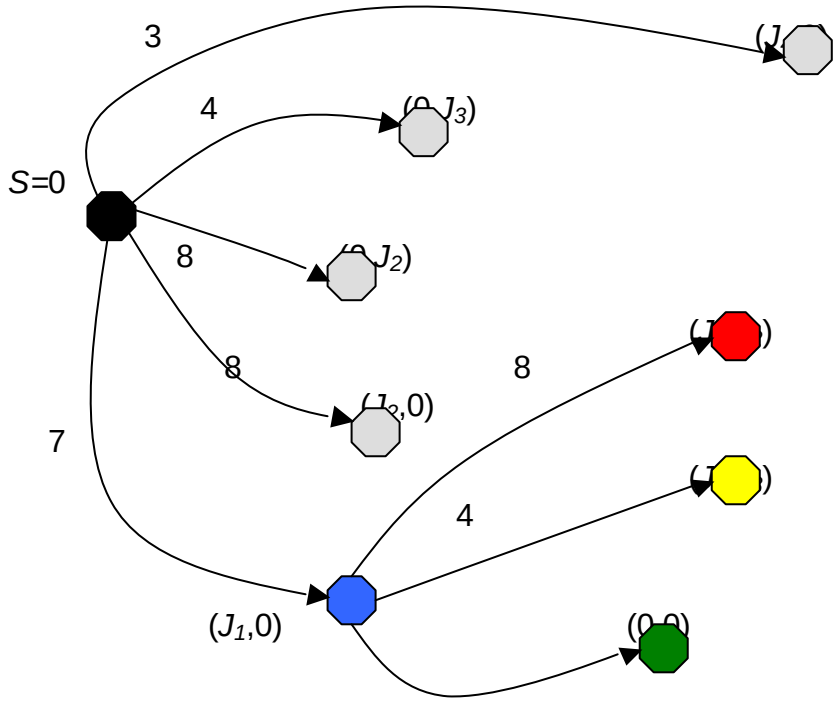

Figura 4.4: Segunda fase de construcción.

El peso asociado a cada una de las posibilidades vendrá dado por la suma del correspondiente a la realización del trabajo J_1 y el asociado a cada uno de los arcos que salen del nodo. Obviamente, en la opción marcada en verde, dado que la máquina 2 no realiza ningún trabajo mientras la máquina 1 procesa el trabajo J_1, el peso asociado a esa opción será únicamente el derivado de procesar el trabajo J_1.

Realizando el mismo proceso con todos los nodos alcanzaremos el nodo de llegada T, con lo que la construcción del grafo se dará por concluida. En nuestro ejemplo, el grafo resultante es el mostrado en la figura 4.5:

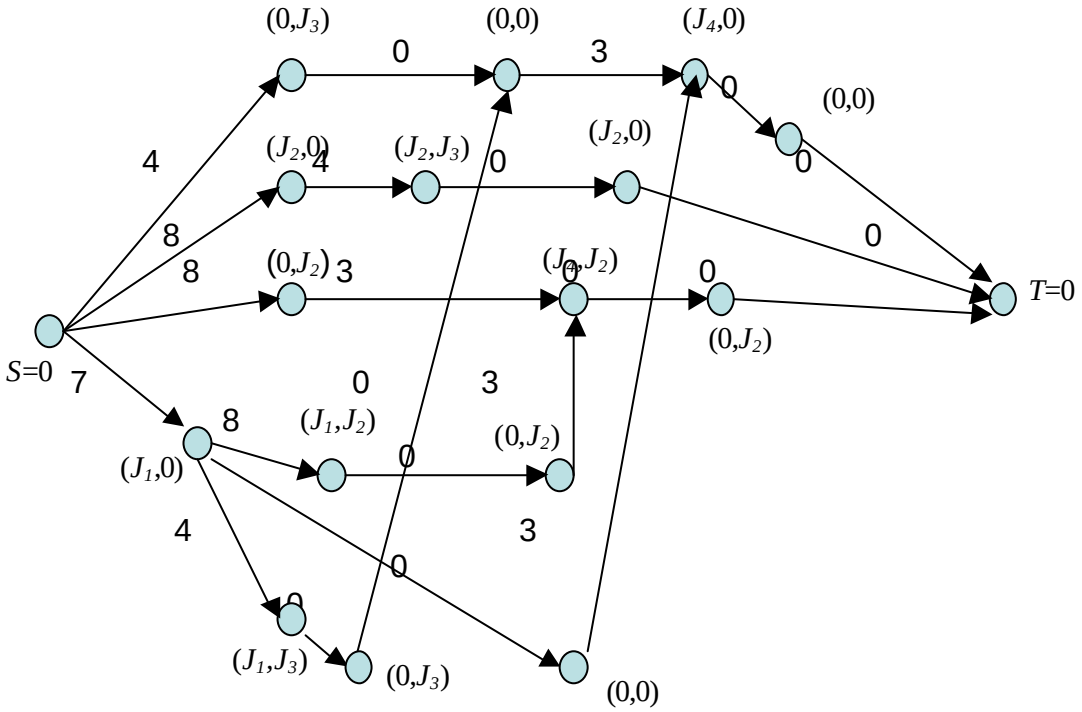

Figura 4.5: Grafo resultante.

Una vez determinado el grafo, pasamos a la segunda fase del algoritmo, su resolución:

Como se ha dicho anteriormente vamos a abordar la resolución del grafo como un problema de ruta máxima en el que los pesos asociados a cada arco van a definir ahora la distancia que separa a los nodos correspondientes. Se trata pues de determinar el camino a recorrer entre el nodo de salida S y el nodo de llegada T de manera que se maximice la distancia recorrida. Una vez hallada la solución, los nodos pertenecientes a ella determinarán los trabajos a realizar y las máquinas que se emplearán para ello.

Lo primero que vamos a hacer es plantear el modelo en su forma estándar. Así, tenemos que la función objetivo viene dada por:

$$Max \sum_{i} \sum_{j} w_{ij} \cdot X_{ij}$$

Donde X_{ij} es una variable que toma el valor 1 si el camino elegido utiliza el arco que va desde i hasta j y 0 si el camino elegido no pasa por dicho arco.

Vamos a considerar que del nodo de salida S sólo sale una unidad de flujo, que debe llegar al nodo de entrada T. A medida que viaje por el grafo iremos acumulando un valor que depende de las distancias recorridas en el camino

elegido. Dicho valor es el que tenemos que maximizar. La razón por la que se trabaja con una sola unidad de flujo es que el camino elegido es independiente del número de unidades de flujo que se envían, es decir, si se mandan diez unidades en lugar de una, todas irán por el mismo camino por lo que resulta mucho más sencillo a la hora de realizar los cálculos trabajar con una sola de ellas.

Maximizar la expresión anterior equivale a:

$$Max \sum_i \sum_j w_{ij} \cdot X_{ij} = Min \sum_i \sum_j -w_{ij} \cdot X_{ij} = Min \sum_i \sum_j C_{ij} \cdot X_{ij}.$$

Por lo tanto, podemos considerar también como criterio de optimización:

$$Min \sum_i \sum_j C_{ij} \cdot X_{ij}.$$

Donde los C_{ij} se interpretan como los costes asociados a la utilización de cada uno de los arcos. La solución óptima será aquella que proporcione un menor coste y se corresponderá con el camino que maximiza la longitud recorrida.

Las restricciones a imponer para la correcta resolución del problema serán:

El problema puede ser representado en forma matricial, mediante la formulación de la matriz de incidencia, que tiene tantas filas como nodos y tantas columnas como arcos. Así, para un arco cualquiera *i-j*, el vector columna que representa el valor de X_{ij} en la matriz de incidencia presenta la siguiente forma:

Figura 4.6: Arco i-j.

Arcos(*A*)

Nodos (*N*)	...	X_{ij}	...
S	...	0	...
...	...	0	...
I	...	1	...
…	...	0	...
j	...	-1	...
…	...	0	...
T	...	0	...

Tabla 4.3: Columna representativa de la matriz de incidencia.

Donde los valores de la matriz de incidencia pueden ser:

- 1 si el arco sale del nodo.
- -1 si el arco entra en el nodo.
- 0 en cualquier otro caso.

Obviamente, en cada una de las columnas de la matriz de incidencia habrá un valor 1 y un valor –1, siendo nulo el resto de valores. Esto se debe a que cada arco sale de un nodo y entra en otro.

La resolución de este problema pasa por la resolución de su dual, cuya formulación es la siguiente:

Las variables del dual serán l_j, existiendo una por cada fila (nodo) de la matriz de incidencia. Por lo que se tienen (l_S, l_1, l_2, ..., l_j, ..., l_T).

Como en cada columna (arco) de la matriz de incidencia hay un 1 y un –1, y el resto de valores son nulos, en cada una de las restricciones del problema dual se tendrán dos variables landa, una sumando y otra restando. Por consiguiente, la forma general de dichas restricciones será:

$$l_i - l_j \leq C_{ij}$$

La función objetivo en este caso vendrá dada por:

$$Max \sum_{j} l_j \cdot b_j$$

Que, si sustituimos los valores de b_j se transforma en:

$$Max\ l_S - l_T$$

Siendo las variables l_i libres.

La elaboración de un algoritmo para la resolución del problema de flujo con coste mínimo se basa en proponer un procedimiento para la obtención de una solución admisible para el problema dual, mostrando que la solución complementaria es a su vez admisible para el problema primal.

El algoritmo que se presenta para la resolución de este grafo requiere que dicho grafo sea un grafo numerado, de forma que se constituya una red creciente.

El procedimiento de numeración es el siguiente:

- Se asigna un 1 a un nodo j de forma que todos los arcos (i, j) que terminan en j provienen del nodo S.
- Una vez numerado el subconjunto de nodos c ={1, 2, ..., n} se asigna el número n+1 al nodo j de forma que todos los arcos (i, j) del grafo procedan de nodos $i \subseteq$ c, los cuales ya han sido numerados.

Un resultado de la dualidad es que si $X_{ij} > 0$ en el óptimo del primal, los costes relativos del dual $r(X_{ij}) = 0$.

Teniendo en cuenta que las holguras del problema dual coinciden con los costes relativos del primal:

$$H_j = r(X_{ij})$$

Llegamos a la conclusión de que las holguras del problema dual son nulas, por lo que las restricciones se transforman a la forma siguiente:

$$l_i - l_j = C_{ij}$$

Una vez resuelto el problema, habremos encontrado una ruta del grafo en la que la suma de los C_{ij} de los arcos pertenecientes a la solución es mínima:

$$\text{Ruta mínima: } C_{Si} + C_{ij} + ... + C_{gk} + C_{kT}$$

Lo que a su vez implica que:

$$X_{si} = X_{ij} = ... = X_{gk} = X_{kT} = 1$$

Siendo nulas el resto de variables.

Dado que $l_i - l_j = C_{ij}$, sustituyendo en la ruta mínima tenemos que:

$$C_{Si} + C_{ij} + ... + C_{gk} + C_{kT} = l_S - l_I + l_i - l_j + ... + l_g - l_k + l_k - l_T = l_S - l_T$$

Por lo tanto la ruta mínima se puede resolver hallando $l_S - l_T$. Este término se puede simplificar asignando a l_T un valor arbitrario, por ejemplo cero. De esta forma, la ruta mínima será l_S.

Una vez formulado el problema vamos a ver cómo trabaja el algoritmo:

En primer lugar, este algoritmo asigna un valor cero a l_T, y considera ese nodo marcado.

En los siguientes pasos se van marcando sólo aquellos nodos j cuyo conjunto adyacente posterior este totalmente marcado. Así, se van marcando sucesivamente los nodos hacia atrás en el grafo, hasta llegar al nodo de salida S. La ruta seleccionada será aquella que minimice el valor de landa según la ecuación:

$$l_j = Min\ (C_{jk} + l_k)$$

La ruta de coste mínimo la determinarán los nodos que van haciendo mínimos los sucesivos l_j.

El proceso finaliza cuando el nodo inicial *S* es marcado.

La solución encontrada en el primal cumple las restricciones del primal, mientras que las soluciones encontradas para el dual también cumple las restricciones del dual. Tenemos entonces que la solución es admisible para ambos casos por lo que podemos afirmar que la solución encontrada por el algoritmo es óptima.

Para ver de una manea más clara el funcionamiento del algoritmo, vamos a resolver el siguiente ejemplo:

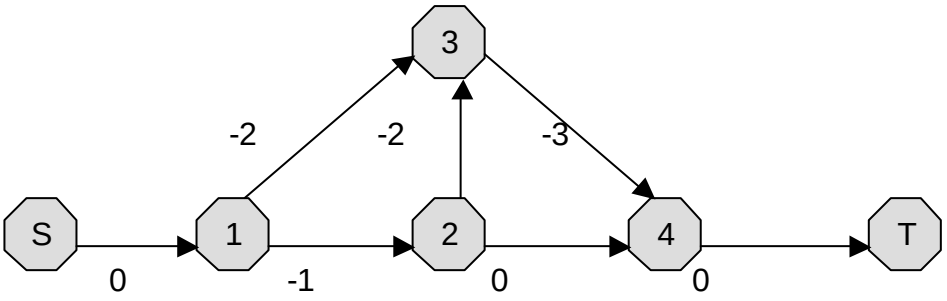

Figura 4.7: Ejemplo de localización de ruta máxima.

El proceso de aplicación del algoritmo comienza asignando $l_T = 0$ y considerando al nodo T como marcado.

Los valores numéricos asociados a cada arco son los costes C_{ij} del problema estándar, por lo que los valores de los arcos son $w_{ij} = - C_{ij}$.

El siguiente nodo en ser marcado es el 4, teniéndose que:

$$l_4 = Min\ (C_{4T} + l_T) = Min\ (0 + 0) = 0$$

Cabe preguntarse ahora qué nodo debe ser marcado; El nodo 2 no puede ser porque no tiene todos sus nodos adyacentes marcados. Por lo tanto marcamos el nodo 3, resultando:

$$l_3 = Min\ (C_{34} + l_4) = Min\ (-3 + 0) = -3$$

A continuación marcamos el nodo 2 que ahora sí tiene todos sus nodos adyacentes posteriores marcados:

$$l_2 = Min\ (C_{23} + l_3, C_{24} + l_4) = Min\ (-2 - 3, 0 - 0) = -5$$

Realizando el mismo proceso con el nodo 1 obtenemos:

$$l_1 = Min\ (C_{12} + l_2, C_{13} + l_3) = Min\ (-1 - 5, -2 - 3) = -6$$

Por último: $l_S = Min\ (C_{S1} + l_1) = -6$.

Luego la ruta mínima tiene un valor igual a –6 y los nodos que la forman son: S, 1, 2, 3, 4, T.

La solución del primal vendrá dada por:

$$X_{S1} = X_{12} = X_{23} = X_{34} = X_{4T} = 1$$

Esta solución cumple las restricciones del dual por la propia forma en la que se ha calculado las variables λ. Por tanto la solución encontrada con este algoritmo es óptima.

Considerando el problema de maximizar, el resultado de la función objetivo seria FO(máximo) = - FO(mínimo)= 6, que resulta ser el resultado de la ruta máxima.

4.2. Algoritmo de búsqueda TABÚ

En esta sección se describe el algoritmo de búsqueda tabú propuesto para resolver el problema POC. El método de búsqueda tabú es un procedimiento heurístico que explora el espacio de soluciones a través de movimientos repetidos desde una solución a la mejor de sus soluciones vecinas, con ciertas restricciones en la elección de la solución de cada iteración del proceso. Búsqueda tabú ha sido utilizada con éxito para resolver complejos problemas de optimización combinatoria, particularmente en el campo de la programación de trabajos. Los elementos del algoritmo se describen a continuación:

Solución Inicial: Para obtener una solución inicial consideramos una asignación de carácter voraz, según el peso de cada uno de los trabajos. Cada trabajo se asigna de forma aleatoria a una máquina de un tipo compatible con el trabajo. El proceso finaliza cuando no es admisible la asignación de ningún otro trabajo.

Movimientos y Estructura de la Vecindad: Dada una solución S, denotamos por N(S) al conjunto de todas las soluciones admisibles que pueden ser obtenidas desde S usando uno de los siguientes movimientos:

- Intercambio: Un trabajo $J_j \in$ S se sustituye por otro trabajo J_k que no está en la solución actual y que cumple los siguientes requisitos:

    1. Se solapa con el trabajo J_j en algún instante de tiempo.
    2. Es compatible con el tipo de máquina asignada al trabajo J_j.

- Extracción: Movimiento de extraer un trabajo de la solución. Este movimiento es debido a la definición de lista tabú realizada al final de

esta sección. Obviamente, este movimiento lleva a soluciones de peor calidad pero es útil como herramienta de diversificación.

- Inserción: En ocasiones, después de aceptar un movimiento de intercambio o extracción, otros trabajos podrían ser introducidos en el conjunto solución. Este movimiento, por ello, es el primer tipo de movimiento considerado en cada iteración del algoritmo.

Cuando no existe ningún movimiento de inserción admisible, se acude al movimiento de intercambio para generar soluciones vecinas. Si todos los trabajos que no pertenecen a la solución son tabú, entonces se realiza un movimiento de extracción.

En los diagramas expuestos a continuación se ilustran dos iteraciones del algoritmo, asumiendo que inicialmente no existen pedidos tabú:

- Número de máquinas = 2.
- Número de trabajos = 6.
- Solución Actual *S* = {A, B, D, E}.

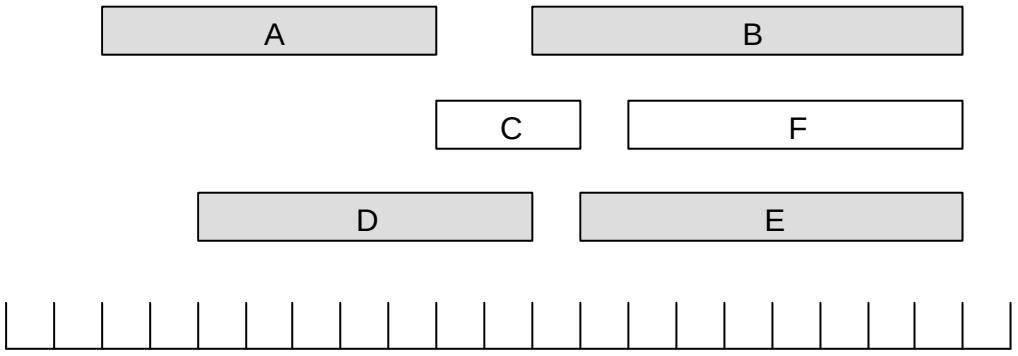

Figura 4.8: Solución de partida.

- 1ª iteración: Intercambio $\rightarrow$ *S* = {A, C, D, E}.

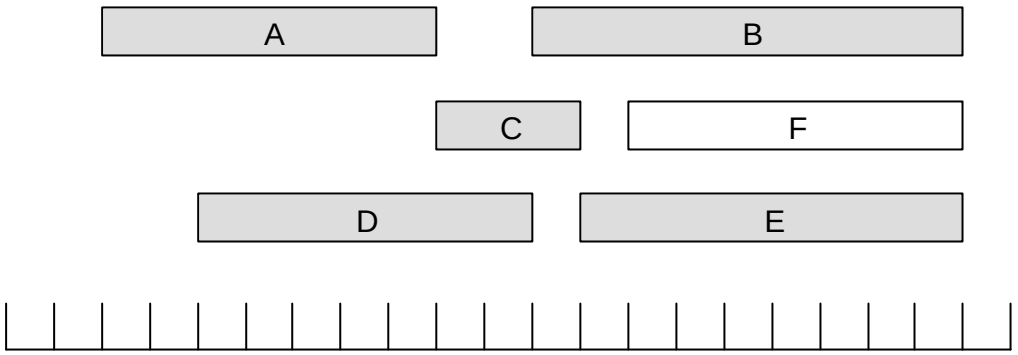

Figura 4.9: Movimiento de intercambio.

- 2ª iteración: Inserción $\rightarrow$ *S* = {A, C, D, E, F}.

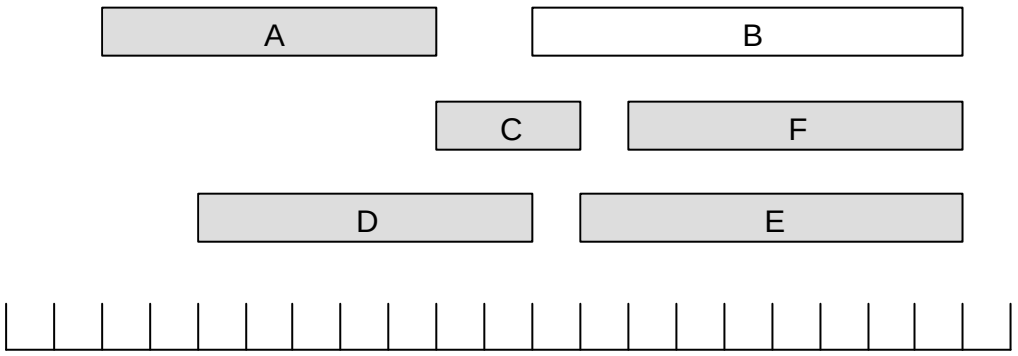

Figura 4.10: Movimiento de inserción.

Lista tabú: Cuando se reemplaza o elimina un trabajo de la solución, introducir este trabajo de nuevo en la solución se considera tabú durante un número determinado de iteraciones. En la búsqueda tabú, la lista tabú puede ser fija o variable. En nuestro método implementamos una estrategia de tamaño variable, donde el tamaño se modifica, con incrementos y decrementos, con objeto de ajustar el número de iteraciones entre la aceptación de dos movimientos de extraer un trabajo de la solución. Un movimiento de extracción es únicamente aceptado cuando todos los trabajos que no están en la solución son tabú. Por tanto, necesitamos definir una frecuencia de incremento *frecinc*, una cantidad a incrementar *I* y una cantidad a

decrementar *D*, para la lista tabú. No necesitamos definir una frecuencia de decremento porque el tamaño de la lista tabú se reduce automáticamente cada vez que se acepta un movimiento de extracción. Inicialmente, el tamaño de la lista se incrementa en cada iteración hasta que todos los trabajos que están fuera de la solución sean tabú.

Otro aspecto a considerar en el algoritmo para mejorar su funcionamiento y permitir una mejor exploración del espacio de soluciones, es la de evitar que siempre sean los mismos trabajos los que salgan de la solución cuando sea necesario aplicar un movimiento de extracción. Esto suele ocurrir generalmente, puesto que el algoritmo escoge siempre el que posee un peso mayor. Por ello, el algoritmo mantiene también una segunda lista tabú que prohíbe a un trabajo, durante un número determinado de movimientos de extracción *Tb_Ext*, abandonar la solución.

Criterio de Aspiración: Usamos la forma más simple de criterio de aspiración, la cual acepta un movimiento tabú si produce mejor solución que la mejor solución encontrada hasta ese momento.

Criterio de Parada: El algoritmo finaliza después de un número fijo de iteraciones.