4. Procedimiento de solución

Para probar la validez del modelo desarrollado, se va a implementar su programación en la herramienta de optimización del programa Matlab R2008a.

4.1. Sub-rutina en Matlab®

El objeto de desarrollar una subrutina en matlab es la de crear un método rápido que genere las matrices de coeficientes A, el vector de recursos b y el vector de costes c de un problema de programación lineal del tipo:

$$\text{Min } c \cdot x$$

$$A \cdot x \leq b$$

$$x \geq 0$$

La herramienta Matlab tiene potencia y capacidad suficientes para encontrar la solución a problemas académicos de dimensiones contenidas. Sin embargo, hay una elevada cantidad de datos que se manejan en este problema y se debe de ajustar las dimensiones de las matrices y vectores al mínimo valor posible.

Parece que la tendencia a reducir peso en la aeronáutica se impone en todos sus aspectos, reduciendo también las dimensiones de modelos matemáticos que optimicen su gestión.

4.1.1.- Matriz del sistema - Incógnitas

Para poder manipular las ecuaciones en Matlab se deben de agrupar las incógnitas en un vector y desarrollar la matriz A de coeficientes por sub-matrices de ceros y coeficientes.

Las variables asociadas a una tarea cuentan en el modelo con dos subíndices, *j* e *i*. Para poder agrupar las incógnitas en un vector fila o columna es necesario eliminar el doble subíndice y por ello se ordenan con un único índice, siguiendo sus índices i y *j*.

Las variables se agrupan en un vector *x* formado a su vez de los siguientes sub-vectores:

$$x = [\ \underline{S}\ \mid\ \underline{V}\ \mid\ \underline{f}\ \mid\ \underline{h}\ \mid\ \underline{F}\ ]_n$$

Denominando *d* al número de tareas a realizar cada día o en el horizonte temporal considerado, cada uno de los subsectores anteriores tiene unas dimensiones determinadas:

- *S*: este vector comprende agrupa las variables s_{ij}. Para cada centro de trabajo *i* hay una tarea *j*. Puesto que el índice *i* toma valores de 1 a 9 y el índice *j* desde 1 hasta diferentes límites, para el tratamiento en Matlab de las variables, éstas se ordenan primero por su índice *i* y a continuación por el *j*. Habrá una variable s_i por cada tarea contemplada cada día, por tanto el vector S tendrá *d* elementos.

- *V*: este vector contiene todas las variables v_{igkj}. A cada tarea de índices *ig* le corresponderán (*d*-1) índices diferentes *kj* pues no tiene sentido una variable v_{igig}. Por tanto, el vector V contendrá *d·(d-1)* variables v_{igkj}.

- *f*: es un vector que contiene *d* variables f_{ij}, una por cada tarea *ij.*

- *h:* análogamente, el vector h contiene también *d* variables h_{ij}.

- *F:* éste vector es en sí la variable de flujo neto F.

Denominando *n* al número total de incógnitas del problema, se tiene que el valor de n es función del número de tareas diarias cuya relación es:

$$n\ (d) = d + d\cdot(d-1) + d + d + 1 = d\cdot(d-1) + 3\cdot d + 1$$

4.1.2.- Matriz del sistema - Restricciones

Respecto a las restricciones, en este caso se agrupan en submatrices para formar la matriz de coeficientes A. Las submatrices las he nombrado con dos subíndices: el primero de ellos se corresponde con el grupo de restricciones que representa y el segundo índice es un índice ordinal de izquierda a derecha.

La forma de la matriz se muestra en la figura 4.1:

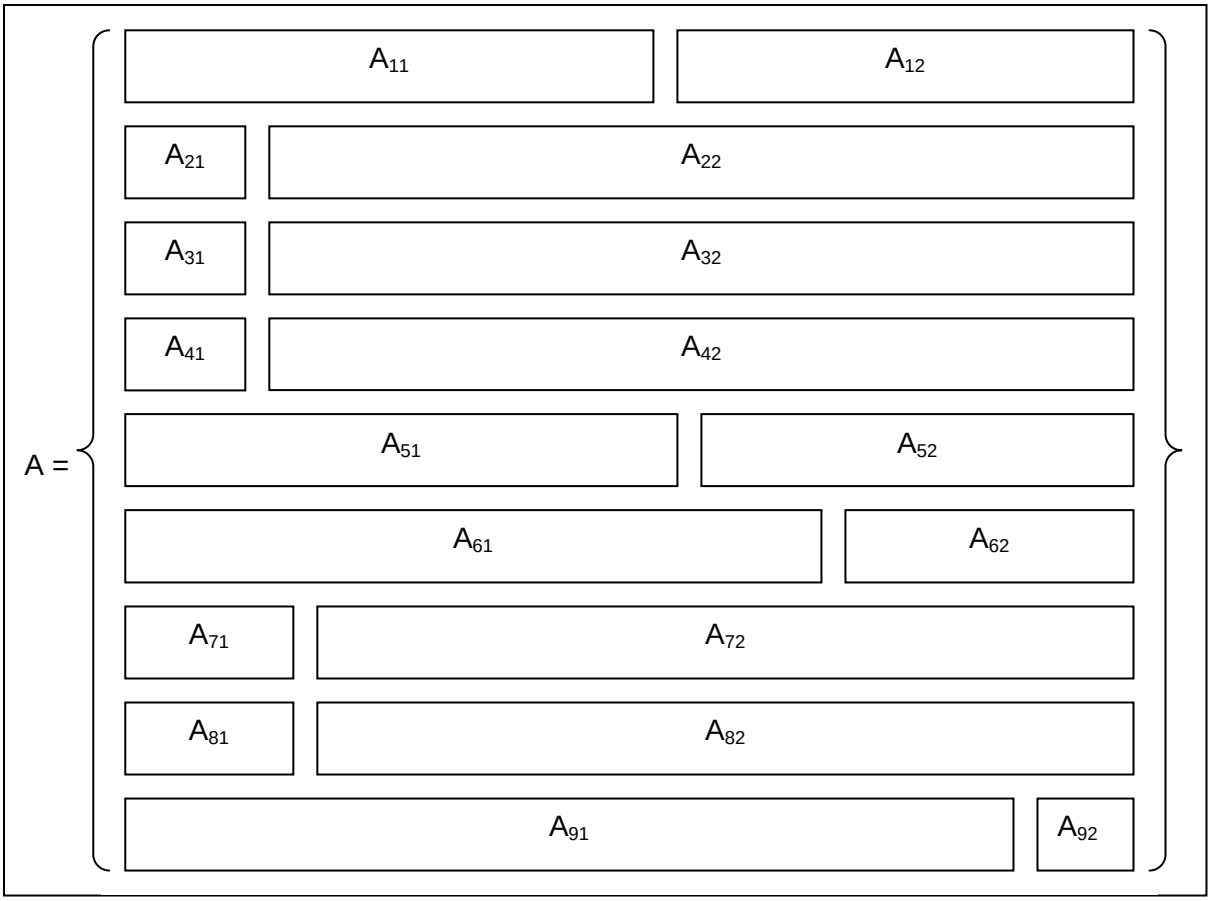


Figura 4.1. Esquema de la composición de la matriz A del problema

El número de columnas de la matriz A lo determina el número de variables que se ha presentado en el punto anterior. El número de filas de la matriz A coincide con el número total de restricciones: para hallarlo más fácilmente, se estudia cada matriz por separado.

En las restricciones (I) sólo aparecen las variables s_{ij} y v_{igkj}, siendo nulos los coeficientes correspondientes a f_{ij}, h_{ij} y F (sub-matriz de ceros A_{12}). Como hay una restricción por cada variable v_{igkj}, la matriz A_{11} tendrá igual número de filas que variables v_{igkj} hay, es decir, $d \cdot (d-1)$.

En las restricciones (II), referentes a la condición de flujo de salida de cada centro de trabajo, se combinan las variables v_{igkj} y h_{ij}. La sub-matriz A_{21}, que contiene los coeficientes de las variables s_{ij}, tiene todos sus términos nulos. La sub-matriz A_{22}, por su parte tendrá tantas filas como variables h_{ij}, es decir, d filas.

En las restricciones (III), referentes a la condición de flujo de entrada a cada centro de trabajo, se combinan las variables v_{igkj} y f_{kj}. Análogamente al grupo (II), la sub-matriz A_{31} tendrá todos sus coeficientes cero y la sub-matriz A_{32} tendrá tantas filas como variables f_{kj} hay, es decir, d filas.

En las restricciones (IV), referentes a los arcos entre cada dos nodos diferentes tienen como variables asociadas v_{igkj}. Se contabilizan tantas ecuaciones como arcos diferentes se puedan trazar: en total, la suma (d-1) + (d-2) + (d-3) + … + 1, es decir, la suma de una serie aritmética, d·(d-1)/2. Como ejemplo véase la figura 4.2 que representa los casos para d =4 y d=5:

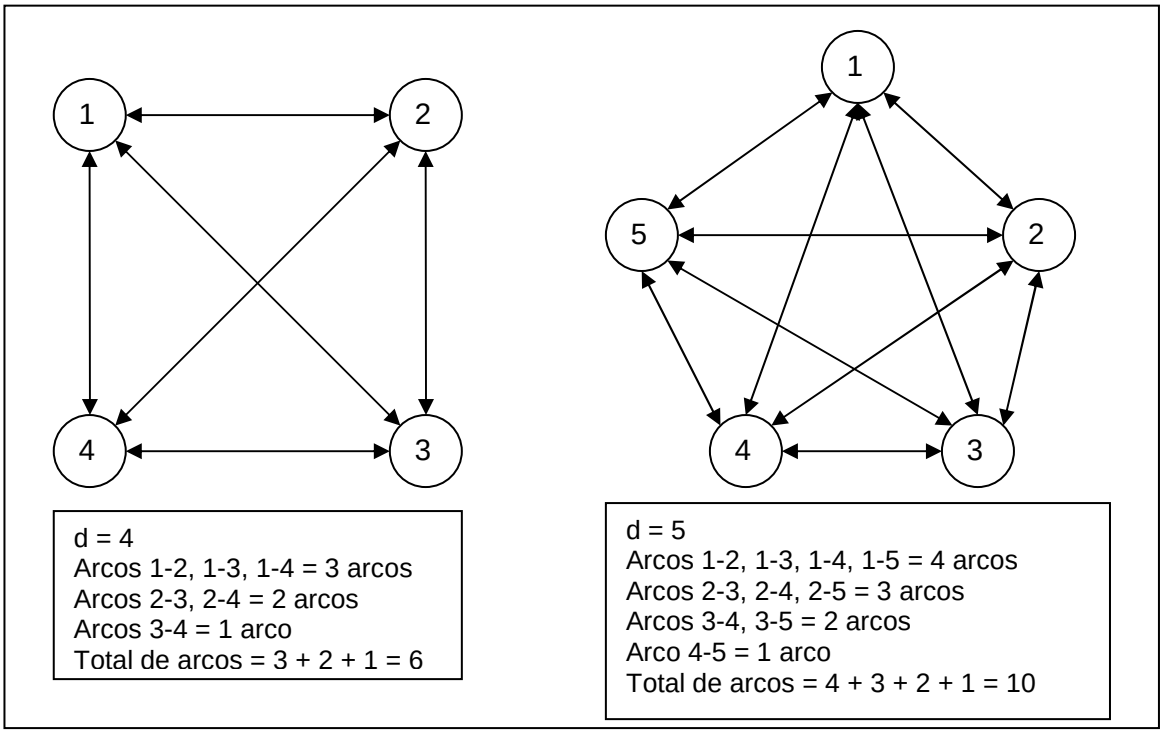


Figura 4.2. Representación de los grafos asociados a las variables v_{igkj} para los casos particulares de sólo 4 y 5 tareas.

En las restricciones (V) y (VI) se impone la igualdad entre el flujo de salida del nodo fuente F y el flujo de entrada del nodo sumidero S. Por tanto, las sub-matrices A_{51} y A_{61}, que contienen los coeficientes de las variables s_{ij} y v_{igkj} son una fila llena de ceros. Por su parte, las sub-matrices A_{52} y A_{62} tendrán una única fila y contabilizarán entre ambas 2 filas.

En las restricciones (VII) y (VIII) se imponen límites superiores e inferiores al término y al inicio de cada tarea *ij* respectivamente. Por tanto, todos los elementos de las sub-matrices A_{72} y A_{82}, coeficientes de las variables v_{igkj}, f_{ij},h_{ij} y F, son cero. Por su parte, las sub-matrices A_{71} y A_{81} tienen tantas filas como *d* variables se tengan que completar cada horizonte temporal considerado.

La restricción (IX) es de una única fila pues es un límite inferior para la variable F.

La suma del número de filas de todas las sub-matrices se corresponde con el número total de filas de la matriz A del sistema: para este modelo, el número de filas sigue la expresión:

$$m = d\cdot(d-1) + d + d + d\cdot(d-1)/2 + 1 + 1 + d + d + 1 = 3/2 \cdot d\cdot(d-1) + 4d + 3 \rightarrow O(d^2)$$

En definitiva, las dimensiones de la matriz A_{mxn} son $[d\cdot(d-1) + [(d-1)+(d-2)+\ldots+1] + 4\cdot d + 3]$ filas y $[d\cdot(d-1) + 3\cdot d + 1]$ columnas.

Para crear los distintos grupos de restricciones se van creando matrices de ceros con las dimensiones apropiadas y se van rellenando los elementos oportunos. Para ello se recurre a bucles “for” anidados con bucles “if” y otros bucles “for”.

Donde se encuentra una mayor complejidad es en la creación del vector de recursos b para el grupo de restricciones (I). Cada elemento es del tipo: $M-t_{ij}-e_{ik}$. Se necesita contabilizar dos veces el índice de la tarea y compararlo con la matriz de valores e_{ik}: esto se consigue con dos bucles for y 4 índices diferentes.

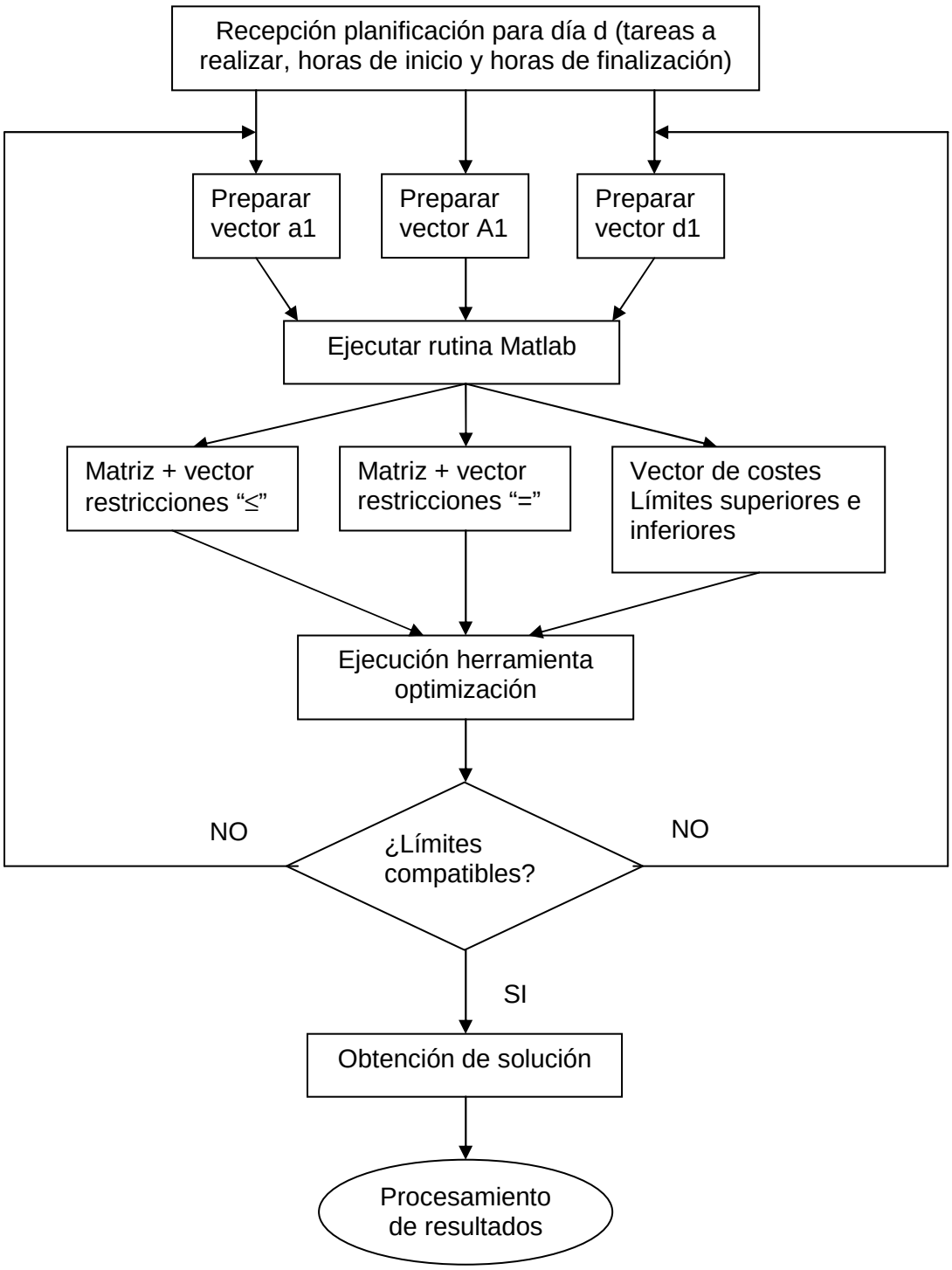
El resultado que aporta la subrutina de Matlab son las matrices y vectores necesarios para la resolución con la herramienta de optimización. Para ejecutar la herramienta se necesitan:

- un vector de costes, “c”, con los pesos de cada variable en la función objetivo;
- una matriz de coeficientes A correspondientes a las restricciones de signo “≤”. Toda restricción de signo “≥” se multiplicará por “-1” para convertir su signo a “≤”;
- un vector de recursos b correspondiente a las restricciones de signo “≤”. Aquéllos términos correspondientes a restricciones del estilo “≥” serán multiplicados por “-1”;
- una matriz de coeficientes A_m de las restricciones de signo “=”;
- un vector de recursos b_m de las restricciones de signo “=”;
- dos vectores de límites superior e inferior con cotas para cada variable.

El pseudocódigo relacionado con la sub-rutina de Matlab® se incluye en el pseudo-código que se presenta en el apartado 4.3.2.

4.2. Diagrama de flujo

A continuación se presenta cómo funciona el procedimiento de solución desarrollado en este proyecto:



4.3. Aplicación del modelo a una situación simplificada empleando la herramienta de optimización de Matlab®.

4.3.1. Consideraciones respecto a la herramienta Solver® de Matlab®

La herramienta para optimizar de Matlab® permite el empleo de diversos métodos internos que ajustan por mínimos cuadrados, minimizan funciones multi-objetivo y otros métodos más aparte del empleo del método Simplex para problemas lineales.

Se debe de elegir, en primer lugar, qué método de optimización emplear: el que mejor se ajusta a las necesidades es el algoritmo Simplex: se toma el de pequeña escala, pues el de grandes tamaños hace una aproximación por objetivos.

Sin embargo, tras realizar unas primeras pruebas, se observa que el algoritmo escogido dentro de la herramienta de optimización de Matlab® no es capaz de interpretar las variables binarias como tales: les da valores reales positivos.

Este hecho ocurre porque el método de Matlab® va modificando el valor de las variables para que la desviación de la restricción sea mínima e itera sucesivamente hasta que dichas desviaciones son inferiores a una determinada tolerancia (por defecto 10^{-6}).

A simple vista, no parece que pueda haber problemas si dichos valores decimales se acercan mucho a valores enteros, en cuyo caso se considerarían como enteros. Pero lo cierto es que se puede volver inútil el algoritmo: la consecuencia de no reconocer variables binarias es que los valores de las mismas, en el presente modelo v_{igkj}, se toman como decimales y se corre el riesgo de que se cumplan las restricciones del grupo (I), pero se incumplan las relaciones de precedencia de unas tareas a otras. Por tanto, no es admisible dicho comportamiento.

Para mayor claridad del conflicto que surge, se presenta el siguiente ejemplo:

- Supongamos que
    - la tarea 1 tiene que empezar 60 minutos después del inicio de la jornada y una duración de 120 minutos, esto es, $S_1 \geq 60$ y $t_1 = 120$.

- otra tarea 2 tiene que empezar 90 minutos después del inicio de la jornada y tiene una duración de 15 minutos, esto es, $S_2 \geq 90$ y $t_2 = 15$.

- Las ecuaciones que relacionan ambas variables son:

  - $S_1 + t_1 + e_{12} \leq S_2 + M\cdot(1-v_{12}) \rightarrow S_1 + t_1 + e_{12} - S_2 \leq M\cdot(1-v_{12})$
  - $S_2 + t_2 + e_{21} \leq S_1 + M\cdot(1-v_{21})$
  - $V_{12} + v_{21} \leq 1$

- La herramienta Solver® de Matlab® puede hacer que se cumpla que S_1=60, S_2=90 y v_{12}~1 y que la primera restricción sea cierta.

La primera ecuación está diseñada para ser redundante en el caso de que ambas tareas no sean sucesivas y para que impida el solape de tareas cuando sí lo sean. Si el valor de v_{12} no es 1, sino que se le aproxima, la cantidad $M\cdot(1-v_{12})$ = "$M\cdot\sim0$", una cantidad que es finita y puede ser superior a "$S_1 + t_1 + e_{12} - S_2$". Se cumpliría la restricción, se consideraría la tarea 2 sucesora de la 1 cuando, realmente, por el tiempo necesario para completar la primera tarea, t_1, sería físicamente imposible.

Considerando esa limitación de la herramienta Matlab®, queda una opción para resolver un caso particular y probar la validez del modelo: se puede resolver el problema relajado, imponiendo valores a las variables v_{igkj} y resolviendo cada una de las combinaciones posibles.

Para ello se empleará un método de exploración dirigida ("branch & bound" en inglés).

4.3.2. Solución del problema relajado mediante la exploración dirigida

En este apartado se presenta en forma de pseudocódigo el procedimiento seguido para resolver el problema relajado.

En primer lugar se presenta en la figura 4.3 la simbología utilizada:

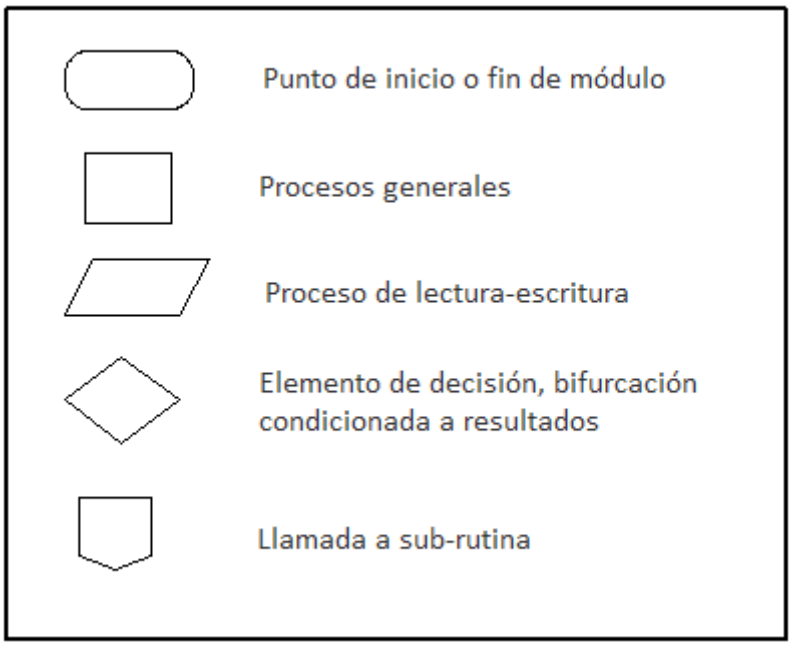


Figura 4.3. Simbología en el pseudocódigo.

A continuación, se presenta el pseudocódigo de la rutina principal empleada en Matlab®:

Nombre de la rutina: Main_02

Parámetros de entrada: tareas a realizar en horizonte temporal determinado, vectores de valores a_{ij} y d_{ij}, combinación de valores para variables v_{igkj}.

Parámetros de salida: vector de costes C_j de la función objetivo, matriz sistema *AM* de restricciones "=", matriz sistema *am* de restricciones "=", vector de términos independientes *bM*,

vector de términos independientes *bm*, vector de límites inferiores de las variables *lowerbound*, vector de límites superiores de las variables, vector de límites superiores de las variables *upperbound*.

Definición: a partir de los datos de entrada genera las sub-matrices presentadas en la figura 4.1. Para la definición de la matriz emplea de forma recurrente bucles “for” e “if”.

En la figura 4.4 se presenta un diagrama de flujo donde se utiliza:

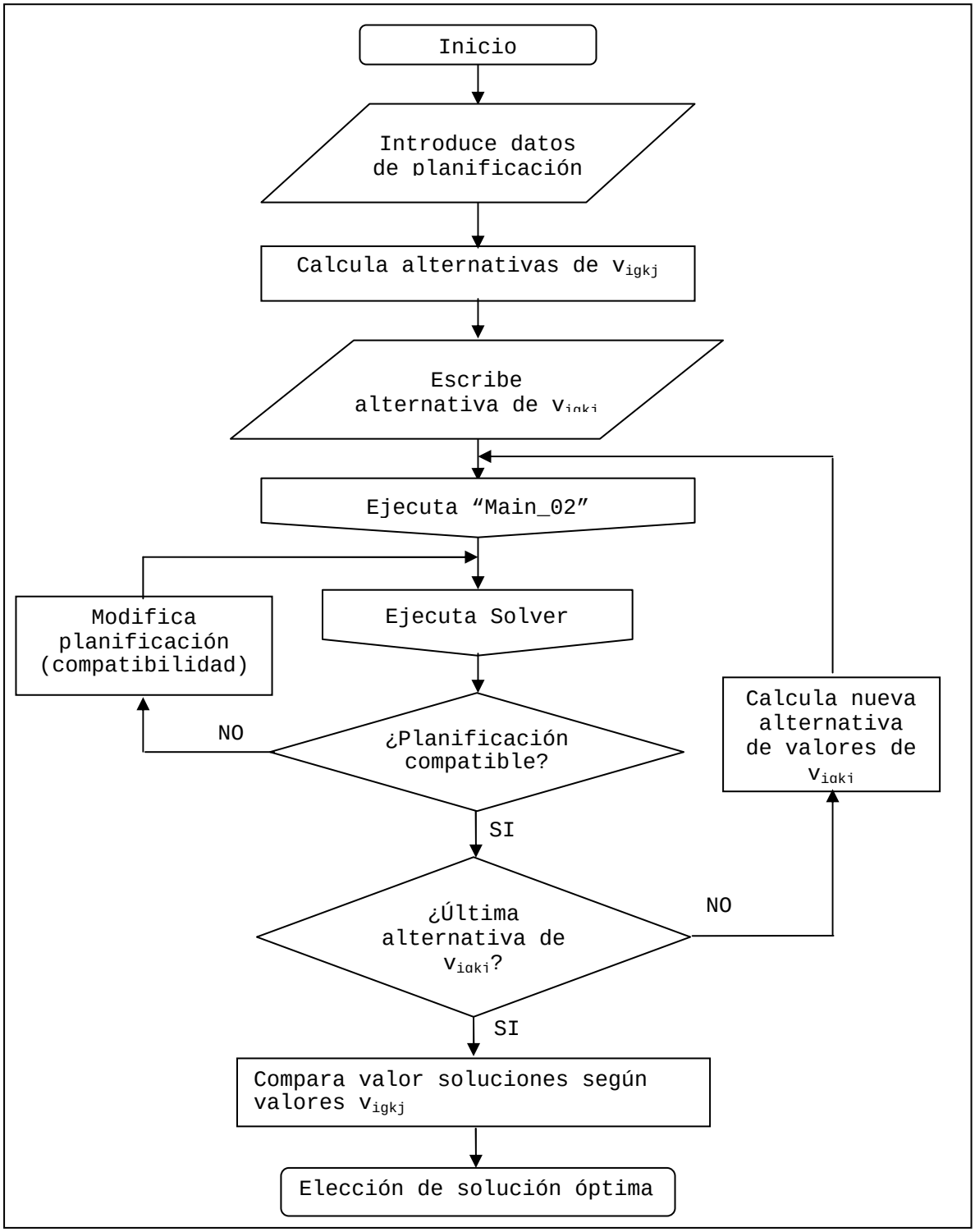


Figura 4.4. Diagrama de flujo donde se emplea la función Main_02.