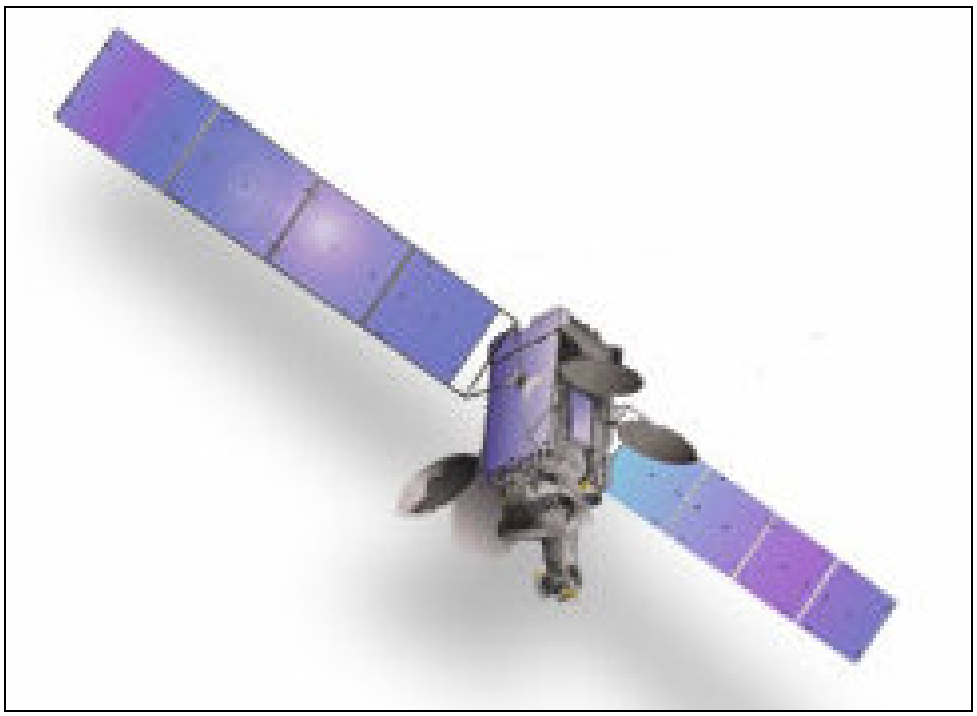
ESCUELA SUPERIOR DE INGENIEROS DE SEVILLA
DEPARTAMENTO DE ORGANIZACIÓN INDUSTRIAL
Y GESTIÓN DE EMPRESAS.

PROYECTO FIN DE CARRERA

PLANIFICACIÓN DE TAREAS DE SATÉLITES DE OBSERVACIÓN TERRESTRE.



AUTOR: MIGUEL A. GALLEGO PINOS
TITULACIÓN: INGENIERO INDUSTRIAL SUPERIOR
TUTOR: D. JOSÉ MANUEL GARCÍA SÁNCHEZ

ÍNDICE GENERAL

1 INTRODUCCIÓN Y OBJETIVOS DEL PROYECTO.

La empresa Earth Observing Satellites (EOS) controla el globo terrestre mediante satélites que realizan continuamente distintas operaciones, como captura de imágenes, mediciones, operaciones de telecomunicaciones, etc. Casi todos estos satélites se encuentran normalmente sobrecargados de forma crítica. Por ejemplo, hay muchísimas más peticiones de observación, por parte de dichos satélites, de las que posiblemente pueden ser satisfechas en realidad. Los sistemas de programación o scheduling suelen satisfacer tantas de estas peticiones como es posible, favoreciendo aquellas que tengan una prioridad más alta. Actualmente, cada satélite está programado por separado mediante una coordinación manual. Y debido a que el número de satélites está creciendo de forma continua e imparable esta manera de controlar dichos satélites llegará a ser cada vez más ineficaz.

El problema de programación de la observación del EOS está caracterizado por múltiples restricciones complejas, incluyendo la energía, capacidad de datos, así como el tiempo limitado que cada satélite pasa sobre cada objetivo. Algunos satélites del EOS pueden hacer centenares de procesamientos de tareas por día, así como cada petición puede tener docenas de oportunidades de ejecución de tareas a la semana, llegando las peticiones de reservas a millares. Así, encontrar una programación óptima o cercana a la óptima implica un espacio de búsqueda muy grande. En general, el tamaño y la complejidad de este espacio imposibilitan una búsqueda completa.

El problema de programación del EOS es un ejemplo de un problema críticamente sobrecargado; esto significa que existen más solicitudes de uso de un recurso de las que pueden ser satisfechas, asegurando que algunas solicitudes serán incumplidas en el horizonte de estudio. Dichas solicitudes no satisfechas en el presente periodo, serán priorizadas, y en consecuencia sí serán satisfechas o procesadas, para el siguiente periodo de trabajo de los satélites. Tales problemas requieren para su estudio telescopios y otras instalaciones de pruebas. Estos problemas implican, pues, la asignación de recursos costosos, a menudo con restricciones complejas y tareas priorizadas. Las programaciones pobres darán lugar a la falta de aprovechamiento de dichas instalaciones costosas y, por tanto, a una pérdida sustancial de dinero.

El problema que trataremos en este proyecto es, por tanto, la optimización de los recursos disponibles (número de satélites para el ejemplo del EOS) para atender a la máxima cantidad posible de tareas involucradas (o solicitudes de tareas por parte de los satélites, en el ejemplo del EOS) dentro del horizonte de planificación, teniendo en cuenta, que existen restricciones de compatibilidad entre tareas y máquinas, que cada tarea o trabajo posee una necesidad de recurso concreta (tiempo dedicado de una máquina) además de un intervalo de activación o comienzo de ejecución.

Este problema puede ser denominado, según el tipo de problema al que pertenece, como **Operational Variable Interval Job Scheduling Problem** (OVISP), teniendo en cuenta ciertas particularidades que serán comentadas en capítulos posteriores, atacando dicho problema de dos maneras claramente diferenciadas.

El desarrollo del proyecto se ha llevado a cabo de la siguiente manera: en los capítulos segundo y tercero se describe y se formula el problema concreto de planificación de los recursos disponibles en sistemas logísticos con restricciones de tiempo, indicando además diversas situaciones concretas donde aparece dicho problema; en el cuarto capítulo resolveremos una batería de problemas haciendo uso de librerías de optimización XA (XA Professional Linear Programing Optimizer), donde se busca que el optimizador resuelva cada problema proporcionando soluciones enteras. Se podrá constatar, a partir de los resultados obtenidos, que conforme aumenta el grado de de complejidad del problema lo hace también el tiempo de resolución; en un quinto bloque de este proyecto se describe un algoritmo de Exploración Dirigida de carácter heurístico, el cual intenta obtener la solución óptima del

problema o, al menos, una buena, esto es, una solución que siendo factible esté muy próxima a la cota superior del problema, de forma que el tiempo empleado sea inferior al del primer método (el del Capítulo 4); en el capítulo sexto se describirá, de forma detallada el diseño funcional que se ha seguido para llevar a cabo la resolución de un problema mediante el uso de las mencionadas librerías XA, así como mediante el algoritmo de Exploración Dirigida implementado; en el capítulo séptimo presentaremos los resultados obtenidos al resolver, como se comentó anteriormente, una batería de problemas generada de forma aleatoria, primero mediante librerías de optimización XA y, posteriormente, mediante el mencionado método de Exploración Dirigida; finalmente, en el capítulo octavo analizaremos los resultados obtenidos y expondremos las principales conclusiones obtenidas en este proyecto, comparando las dos metodologías empleadas en la resolución de los problemas estudiados.

2 PLANIFICACIÓN DE LOS RECURSOS DISPONIBLES EN SISTEMAS LOGÍSTICOS CON RETRICCIONES DE TIEMPO

Los sistemas logísticos con restricciones de tiempo son aquellos sistemas en los que las tareas tienen que ser realizadas dentro de un intervalo de tiempo. Debido a esta restricción, la problemática fundamental se centra en la planificación de los recursos necesarios para realizar dichas tareas. A continuación se describen los diferentes tipos de problemas en este tipo de sistemas, dentro de los cuales se encuentra el nuestro, así como diversas aplicaciones prácticas en las que aparecen.

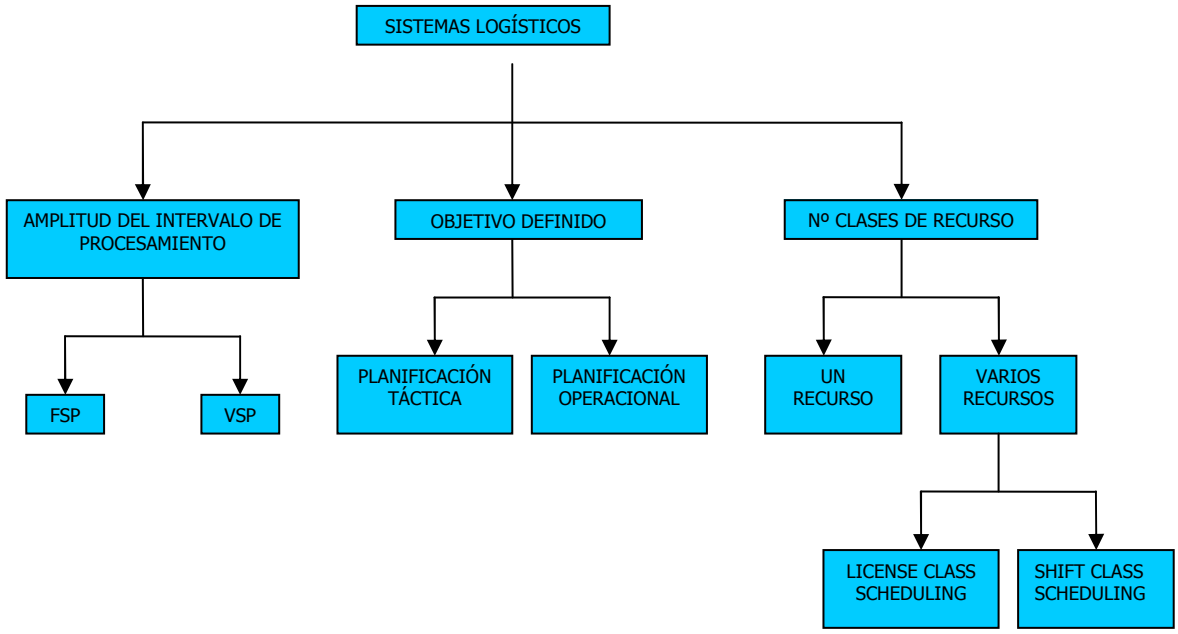
2.1. CLASIFICACIÓN DEL PROBLEMA GENERAL.

De forma general podemos empezar afirmando que la asignación de recursos a actividades fijas es un problema de programación de trabajos sobre un conjunto de máquinas en paralelo. Debido a que las actividades poseen restricciones de tiempo (motivadas por aspectos físicos de los materiales involucrados en la fabricación, por estrictas normas de proceso o, simplemente, por la imposición de un intervalo de procesamiento) el tiempo de finalización de las tareas, se encuentra restringido por ventanas temporales, no permitiéndose cualquier instante del horizonte temporal como tiempo de finalización. Esto implica, que al ser el intervalo temporal fijo, es inadmisible la realización parcial o total de la

actividad fuera del mismo. En estos casos la programación de las tareas depende principalmente de la capacidad del sistema, es decir, de la cantidad de recursos disponibles.

Los sistemas logísticos con restricciones de tiempo, aparecen en la literatura con diferentes variaciones, dependiendo de los intervalos de tiempo para el procesamiento de las tareas, del objetivo definido, del número de clases de recursos y de las características que motivan las diferentes clases.

Un esquema a modo de visión global de las distintas clasificaciones de estos sistemas es la que se muestra a continuación:



Esquema 2-1. Clasificación de los sistemas logísticos.

Una primera clasificación de estos sistemas, se puede hacer atendiendo a la amplitud del intervalo para el procesamiento de las tareas. Según esta clasificación tenemos:

Problemas de Programación de Trabajos Fijos (FSP).

Son problemas donde el tiempo de proceso de las tareas, coincide con la amplitud del intervalo, es decir; no existe ningún tipo de holgura para el procesamiento del trabajo.

Debido a que los trabajos son fijos en el tiempo, pueden ser representados en un diagrama de Gantt, tal y como podemos ver en el ejemplo que se muestra a continuación, en el cual consideramos el caso más simple del problema FSP (una única clase de máquina y de trabajo). En él, cada trabajo puede ser procesado por cualquier máquina y las máquinas están disponibles durante todo el horizonte de planificación.

El ejemplo consiste en 9 actividades, cada una de ellas con un tiempo de inicio y de finalización establecidos de antemano:

Trabajo	Instante de Inicio	Instante de Finalización
1	0	4
2	0	7
3	1	6
4	2	7
5	2	4
6	3	7
7	5	9
8	7	10
9	8	10

Tabla 2-2. Tiempos de los trabajos que intervienen en el modelo.

Como podemos observar, se producen solapamientos entre muchos de los trabajos, lo que se traduce en que será necesaria la intervención de más de una máquina para poder llevar a cabo el procesamiento de todos. A continuación presentamos en un diagrama de Gantt una posible configuración del procesado de los trabajos mediante el uso de "6" máquinas.

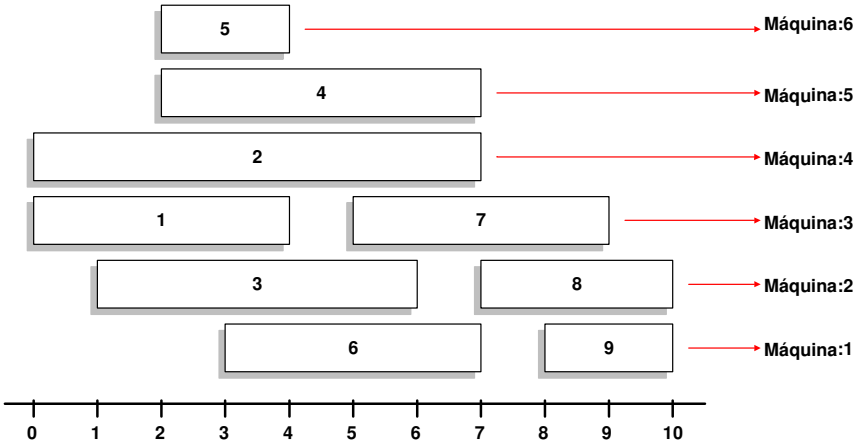


Figura 2-1. Diagrama de Gantt de posible configuración de procesado.

Como se ha comentado, las máquinas que intervienen son de la misma clase al igual que los trabajos.

Problemas Programación de Trabajos Variables (VSP).

El presente proyecto trata sobre la resolución de este tipo de problemas VSP.

Este grupo, comprende los problemas donde el intervalo de tiempo para el procesamiento de cada tarea es mayor o igual que el tiempo de proceso del mismo.

Dado que las tareas son variables en el tiempo, se presenta una holgura para el procesamiento del trabajo. A continuación intentaremos mediante un diagrama de Gantt reflejar lo explicado anteriormente.

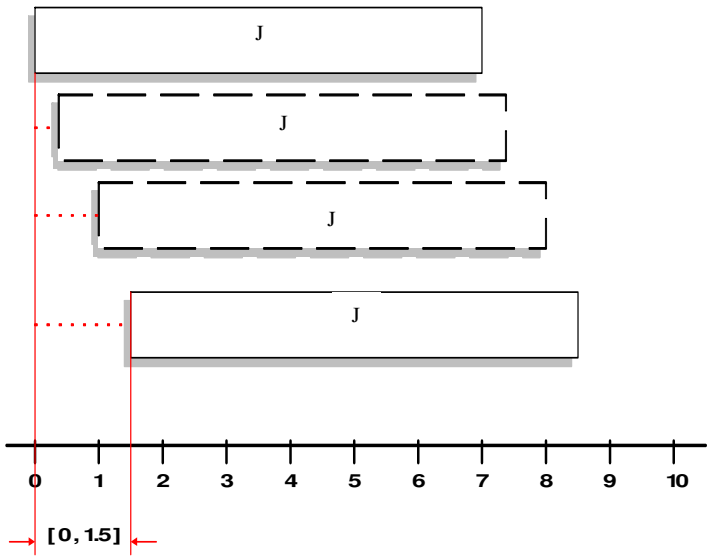


Figura 2-2. Diagrama de Gantt de la holgura que presenta el procesamiento de una tarea.

Se puede observar en el diagrama, que el trabajo "J" presenta una variación de comienzo de procesado que esta comprendido entre [0,1.5]. Esto nos viene a indicar que el comienzo de procesado de una tarea puede ser cualquiera de los dos valores límites del intervalo (en el diagrama vienen marcados con trazo continuo) o un instante perteneciente al interior del mismo (como son los dos casos marcados con trazo discontinuo en el diagrama).

Otro tipo de clasificación sería atendiendo a los objetivos perseguidos, pudiéndose orientar la optimización en dos direcciones:

Planificación táctica de la capacidad.

Se pretende optimizar el uso de los recursos necesarios para atender todas las tareas. Dicho de otro modo, lo que se persigue es minimizar en lo posible el coste total en máquinas requeridas para procesar todos los trabajos.

Podemos ver en la ilustración mostrada para el caso FSP:

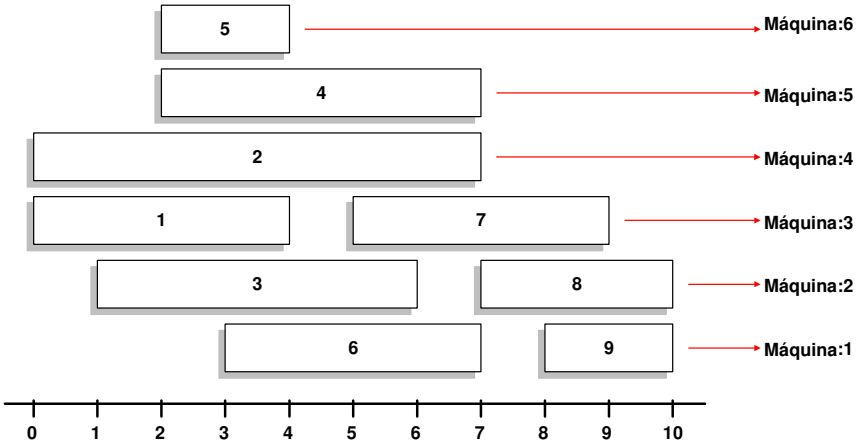


Figura 2-3. Diagrama de Gantt de procesado mediante 6 máquinas.

Se trata de una posible solución, para procesar todos los trabajos sin que existan problemas de solapamientos entre ellos. Como resultado, tenemos que para llevarlo a buen fin es necesaria la intervención de "6" máquinas.

Cabe mencionar, que en este tipo de problemas se les da el mismo peso a todos los trabajos que intervienen.

Planificación operacional de la capacidad.

En este otro caso, a partir de un conjunto de recursos, se trata de asignarlos a tareas con objeto de maximizar el peso total de las tareas completadas. Por tanto, a diferencia del Táctico, aquí toma gran importancia el peso que tenga cada trabajo a la hora de establecer la prioridad de procesado.

El presente proyecto trata sobre la resolución de este tipo de problemas (Planificación Operacional).

Para ilustrarlo, vamos a tomar el mismo grupo de trabajos recogidos en Tabla 2-1 a los que hemos asignado unos pesos para denotar la preferencia de unos sobre otros.

Trabajo	Tiempo de Inicio	Tiempo de Finalización	Peso
1	0	4	6
2	0	7	8
3	1	6	5
4	2	7	3
5	2	4	8
6	3	7	9
7	5	9	1
8	7	10	2
9	8	10	4

Tabla 2-3. Tiempos y pesos de los trabajos que intervienen en el modelo.

Los pesos de los trabajos se han valorado en el intervalo [0, 10] en el que el "0" es el peor valor y por consiguiente el "10" el máximo.

Reflejando los datos de la tabla en un diagrama de Gantt y ordenándolos en orden de mayor a menor peso en sentido ascendente, nos quedan del siguiente modo:

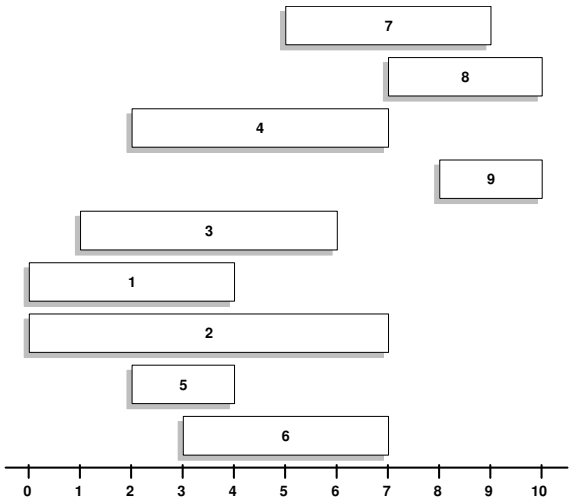


Figura 2-4. Diagrama de Gantt de los trabajos ordenados según peso.

Si consideramos el caso más sencillo del problema FSP (una clase de máquina y de trabajo) y que tan sólo se disponen de tres máquinas para llevar a cabo el máximo número de trabajos atendiendo al peso; una posible configuración podría ser la que mostramos en el siguiente diagrama de Gantt.

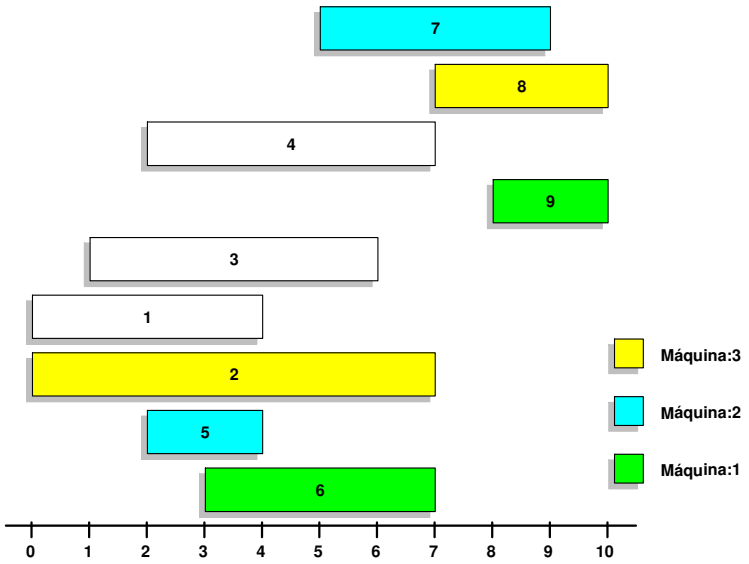


Figura 2-5. Diagrama de Gantt indicando posible configuración de procesamiento.

Como podemos observar en la figura anterior, los trabajos que se procesan son los de mayor relevancia y sin que se produzcan solapamientos. También cabe indicar que los trabajos "1", "3" y "4" no se llevan a cabo por insuficiencia de medios.

De acuerdo al número de clases de recurso, los primeros trabajos sobre el problema consideraron una única clase de recurso, de forma que cada tarea podía ser procesada por cualquier recurso. Sin embargo, el caso más interesante es considerar varias clases de recurso. En este caso se pueden establecer dos tipos claramente diferenciables:

License Class Scheduling.

En este primer caso lo más importante a destacar es que todos los recursos están disponibles durante todo el horizonte de planificación. La compatibilidad entre tareas y recursos tan sólo se rige por motivos técnicos. Es decir, cada clase de máquina que intervenga, estará cualificada para poder procesar ciertas clases de trabajo. Como ejemplo

aclaratorio mostramos la siguiente tabla de compatibilidades para el caso de "3" clases de trabajo y "2" clases de máquina:

		Clase de Máquina	
		1	2
Clase de Trabajo	1	X	-
	2	-	X
	3	X	X

Tabla 2-4. Compatibilidades presentes entre clases de máquina y de trabajo.

Resumiendo:

- Las máquinas de clase "1" pueden procesar aquellos trabajos que sean de la clase "1 y 3".
- Las máquinas de clase "2" pueden procesar aquellos trabajos que sean de la clase "2 y 3".
- Este es el caso estudiado en este proyecto.

Shift Class Scheduling.

En este segundo caso, a diferencia del anterior, cada recurso está sólo disponible en un intervalo o conjunto de intervalos de tiempo. Un recurso puede procesar únicamente las tareas que se encuentran programadas dentro de su intervalo de tiempo.

2.2. OTRAS SITUACIONES REALES.

Los problemas prácticos que implican la resolución de instancias del problema VSP aparecen en múltiples entornos logísticos. Las situaciones reales más importantes que han sido estudiadas hasta la fecha, aparte de la desarrollada en este proyecto referida a la

ejecución de tareas por parte de los satélites, son las que a continuación exponemos brevemente:

Proceso de mantenimiento de aviones en aeropuertos mediante la asignación de técnicos a tareas: Kolen y Kroon (1991/92/93/94); Jason (1994).

Cada día, en todos los aeropuertos del mundo, son miles los aviones que aterrizan y despegan. Estos aparatos, por razones obvias de seguridad, deben ser sometidos a operaciones de mantenimiento durante el periodo de tiempo que permanecen en el aeropuerto.

Dado que los horarios de salidas y llegadas de los aviones son conocidos de antemano, el problema planteado consiste en minimizar el número de operarios de forma que se garantice el mantenimiento de todos los aparatos que pasen por el aeropuerto de la forma lo más satisfactoriamente posible. Este caso correspondería con el problema FSP con objetivo táctico.

También otro factor a tener en cuenta es la condición de que si todos los operarios no están especializados en todo tipo de reparaciones. Si esto ocurre, entonces hay que considerar que existen diferentes clases de máquina y trabajo, siendo el problema además de "Táctico" de la modalidad "License Class Scheduling".

Por otro lado, también se puede ver desde el punto de vista de un problema operacional en el que no todas las operaciones de reparación tendrán la misma relevancia, estableciéndose un peso para cada una de ellas que variará en función de su importancia.

Una variante del caso anterior podría ser el que, debido al reducido número de operarios especializados en unas concretas tareas, las cuales nadie puede llevar a cabo, sus horarios no son capaces de abarcar todo del horizonte de planificación, por lo que el problema se transforma en uno del tipo "Shift".

Asignación de puertas a vuelos en aeropuertos con el objeto de reducir el traslado de pasajeros hasta el terminal: Kroon (1991).

Esta aplicación en concreto, es en la que está basado nuestro proyecto. El modelo consiste en asignar a aviones de diferentes tipos una puerta de embarque durante un intervalo de tiempo fijo. Dado que cada tipo de avión presenta unas características técnicas especiales, aparecen restricciones de compatibilidad entre las puertas de embarque y los aviones. Si no se consigue asignar una puerta a un avión, los pasajeros deben ser transportados hasta el mismo en autobús, aumentando de este modo el coste económico y temporal para el aeropuerto, con el consiguiente deterioro en el servicio para los pasajeros. El objetivo en este caso, consiste en encontrar la mejor solución para la asignación de puertas de embarque, para así minimizar el número de aeronaves que queden fuera de esta asignación y de este modo evitar el uso de autobuses.

Asignación de aulas de clase: Carter (1989).

El problema de la asignación de aulas de clase es un problema bastante próximo a FSP, aunque no posee exactamente las mismas características. Pasamos a realizar una mera descripción del caso. En la mayoría de colegios o universidades el número de aulas de clases disponibles suele ser limitado. Parte de estas aulas pueden ser utilizadas para la impartición de clases; sin embargo existen otras cuyo uso, ya sea por razones de espacio, por la disposición de sus mesas, o por otros motivos, está restringido únicamente a la realización de tareas especificas. Dado que la duración de los trabajos que se realizan en las aulas es un dato que se suele conocer de antemano, en este caso el problema de planificación tendría como objetivo maximizar el número de usos de cada una de las aulas, teniendo en cuenta las restricciones técnicas de utilización de cada una de ellas.

Si partimos del hecho de que el colegio o universidad no tiene pensado la ampliación del número de aulas, nos encontraríamos ante el caso de un problema de planificación operacional. Si por el contrario, la institución lo que pretende es que se impartan todas las clases del modo que sea, el problema planteado sería de planificación táctica.

También a la hora de realizar la asignación, se puede dar la situación en que las clases que se vayan a impartir, según sus periodos de impartición, tengan horarios más fijos

o flexibles, presentándose entonces el problema como uno de planificación con trabajos fijos o variables.

Asignación de conductores en líneas de autobuses: Fishetti (1992).

La variable a considerar en este caso es la planificación de los horarios de trabajo de los conductores de autobuses de tal modo que se minimicen los costes asociados para dotar de servicio a todas las líneas de transporte existentes, entendiendo como costes el número de vehículos y conductores a utilizar. Como condicionante a este modelo se ha de tener en cuenta que cuando se asigna a un mismo conductor dos jornadas determinadas durante dos días consecutivos, éste debe descansar un número mínimo de horas. Además, la solución a este problema deberá ser tal que el reparto de turnos entre trabajadores y entre vehículos sea lo más equitativo posible.

En esta aplicación se pueden dar todas las variantes explicadas en la aplicación de "Proceso de mantenimiento de aviones en aeropuertos mediante la asignación de técnicos a tareas".

Otras aplicaciones.

Asignación de salas de operaciones en hospitales, control del tráfico aéreo, asignación de habitaciones en hoteles, producción y distribución de productos perecederos bajo pedido (el hormigón)...

3 PLANIFICACIÓN OPERACIONAL DE LA CAPACIDAD CON RESTRICCIONES DE TIEMPO.

En este capítulo se describe el problema de la planificación operacional de la ejecución de tareas mediante satélites, mostrando restricciones que hacen complicada dicha planificación. Posteriormente se describirá una formulación formal del problema OVISP.

3.1. PLANIFICACIÓN DE LA EJECUCIÓN DE TAREAS MEDIANTE SATÉLITES (GABREL 1995).

El estudio llevado a cabo en el presente proyecto centra su atención, pues, en la modalidad de la Planificación Operacional con varias clases de recurso e intervalo de proceso mayor o igual al del tiempo de proceso del trabajo (problema OVISP), analizando la complejidad del problema y proponiendo un algoritmo de Exploración Dirigida de base heurística (basada en un Branch & Bound) como alternativa a los métodos exactos ya existentes propuestos por Arkin y Silverberg (1987) y Brucker y Nordmann (1994).

Así, consideremos un satélite que describe una órbita alrededor de la tierra y cuya misión es realizar distintas operaciones, como pueden ser por ejemplo la captura de imágenes de la superficie terrestre, mediciones (atmosféricas o de La Tierra), operaciones de telecomunicaciones, etc. Dado que son numerosas las zonas que demandan alguna de las

operaciones mencionadas por parte del satélite, se plantea el problema de seleccionar cuál debe ser la secuencia de tareas a realizar durante un horizonte temporal dado, de manera que se maximice el número de trabajos ejecutados por cada satélite (problema operacional), siendo las restricciones del problema las siguientes:

- Un satélite sólo podrá realizar tareas referidas a la parte de la superficie terrestre expuesta al sol.

- Debido al movimiento del satélite, la ejecución de cada tarea requerida sólo podrá comenzar en una ventana de posibles instantes de tiempo determinados, dentro del horizonte temporal considerado.

- El satélite sólo puede procesar una tarea en cada instante, teniendo además que dejar un tiempo de transición, supuesto constante, entre dos tareas pertenecientes a la misma secuencia. Ya que los recursos no se encuentran disponibles durante todo el horizonte de planificación es un problema del tipo "Shift Class Scheduling".

Lo que se pretende es maximizar no el número de tareas realizadas, sino más bien el beneficio total reportado por las tareas realizadas, ya que consideraremos que cada operación ejecutada aporta un beneficio, económico o de otro tipo, distinto al que aporta el resto. Según lo descrito anteriormente se trata de un problema operacional, con trabajos con instantes de comienzos variables (ya que el instante de comienzo de ejecución de cada tarea puede variar en el tiempo) y recursos no disponibles durante todo el horizonte de planificación (el satélite tiene unos tiempos de transición entre tarea y tarea).

El problema lo podríamos transformar en uno táctico, si planteásemos que el nuevo objetivo es el de llevar a cabo todas las fotografías. Para ello, ahora lo que habría que calcular sería el mínimo número de cámaras para realizar todas las fotografías. Este no es el caso del estudio llevado a cabo en el presente proyecto, ya que nos quedaremos con el caso operacional.

El **Operational Variable Interval Scheduling Problem** (OVISP) se define como el problema de maximizar el beneficio total obtenido de las tareas que logran ser ejecutadas en

un sistema con restricciones de tiempo con trabajos variables y varias clases de trabajo y máquinas. Cada tarea pertenece a un tipo de trabajo y cuenta con unos tiempos de inicio de procesado variables, de forma que posee (cada trabajo) una ventana de posibles instantes de comienzo de procesamiento. Como condicionante, cada máquina sólo puede realizar un tipo de tarea de un conjunto predeterminado de tareas, pudiendo tan sólo procesar una tarea al mismo tiempo.

A continuación vamos a describir un caso práctico que se asemeja a este tipo de problema y que será el ejemplo sobre el que basaremos nuestro proyecto. Se trata del caso mencionado anteriormente en la introducción de este proyecto, referido a los satélites del EOS que realizan distintas tareas para La Tierra. La tarea será, en este ejemplo, la captura de una imagen de una zona concreta del planeta.

La programación implementada de cada uno de los satélites del EOS procede a capturar tantas imágenes (solicitudes de observación determinadas) con prioridad alta como le sea posible, dentro de un período del tiempo fijo, con la ayuda de un sistema fijo de sensores que está incorporado al satélite. Por ejemplo, se considera que el planificador (en inglés scheduler) del satélite de *Landsat 7* ha hecho un buen trabajo si se hacen cada día al menos 250 observaciones. La manera de programar o planificar la realización de tareas del EOS es complicada debido a un número importante de restricciones. [Potin 1998] enumera algunas de éstas de la siguiente manera:

1. Limitaciones de visita posterior. Las blancos de la observación deben estar dentro de la vista del satélite. Los satélites del EOS viajan en órbitas fijas, generalmente cerca de 800 kilómetros sobre la superficie del planeta, y que vienen a tardar unos 100 minutos en recorrer el perímetro de la tierra cada vez. Estas órbitas pasan sobre cualquier lugar concreto de La Tierra en un número limitado, aunque predecibles, de veces; por ello hay solamente algunas ventanas panorámicas (y a veces ninguna) para un blanco y dentro de un período de tiempo dados.

2. Tiempo requerido para tomar cada imagen. La mayoría de los satélites que observan La Tierra toman una imagen unidimensional y utilizan el movimiento orbital de la nave espacial para barrer fuera del área que será capturada. Por ejemplo, una imagen de Landsat requiere 24 segundos de movimiento orbital.

3. Almacenaje de datos a bordo limitado. Las imágenes se almacenan típicamente en un registrador de datos (Solid State Recorder, SSR), hasta que dichas imágenes pueden ser enviadas a La Tierra.

4. Disponibilidad de la estación terrestre. Los datos del SSR se envían a la tierra cuando el satélite pasa sobre una estación de tierra. Las ventanas de la estación de tierra están limitadas como con cualquier otro objetivo.

5. Tiempo de la transición entre los ángulos de la mirada. Algunos instrumentos de los que se montan en los motores pueden apuntar hacia un lado y hacia el otro fácilmente (recorrido en cruz). Estos motores deben disminuir su potencia, para que dichos instrumentos puedan realizar su labor, y así reducir dicho tiempo de transición al mínimo.

6. Ángulo de apunte. Las imágenes de más alta resolución se toman cuando el blanco está directamente debajo del satélite. Otros ángulos de apunte son a veces requeridos para propósitos específicos.

7. Disponibilidad de la energía. La mayoría de los satélites tienen presupuestos de energía muy restrictivos.

8. Control térmico. Debido a que los satélites se encuentran en un determinado instante dentro de la sombra de La Tierra y, seguidamente, lo hacen fuera de ella, resulta que el ambiente térmico cambia radicalmente y de forma continua. Esto añade restricciones al uso del sensor del satélite.

9. Coordinación de múltiples satélites. En particular, asignando responsabilidades de la toma de algunas imágenes concretas de forma apropiada.

10. Interposición de nubes. Algunos sensores de satélites no pueden ver a través de las nubes.

11. Observaciones múltiples del mismo objetivo por diversos sensores o el mismo sensor en diversos instantes.

Este tipo de restricciones son típicas del tipo de problemas anteriormente denominados Shift Class Scheduling.

En nuestro problema, para emular el método OVISP tomaremos las siguientes consideraciones:

- 1. Se denominará tarea o trabajo a cada una de las solicitudes de ejecución de una tarea. En este proyecto, consideraremos que cada tarea o trabajo poseerá únicamente una ventana de posibles instantes de comienzo de procesamiento.

- 2. Cada satélite será modelado como una máquina que puede tan sólo llevar a cabo una tarea al mismo tiempo, esto es, que no puede procesar dos o más tareas en el mismo instante de tiempo.

- 3. Cada satélite puede procesar ciertos tipos o clases de tareas, esto es, de realizar ciertas clases de trabajos (por ejemplo por problemas de la órbita descrita).

El OVISP da como resultado el número de tareas que pueden ser procesadas en el horizonte de estudio, por ejemplo cada 24 horas. Además proporciona de forma concreta aquellas tareas que serán procesadas y aquellas que no.

3.2. FORMULACIÓN DEL MODELO.

En el OVISP el objetivo es encontrar las tareas que serán procesadas, de todas las solicitudes de tareas recibidas para ejecutar sobre el horizonte de planificación. Cada tarea tiene fijada una ventana de tiempo para el periodo de inicio de su realización, no siendo posible, como comentamos en el apartado anterior, su interrupción una vez comenzada dicha realización.

Vamos a introducir la siguiente notación para definir formalmente el OVISP. Supóngase que existen J tareas requeridas para ser procesadas, es decir, J tareas con solicitudes para ser realizadas, en el horizonte de trabajo $[0,T]$, teniendo asignada cada una de ellas un peso o beneficio p_j, con j= $1\ldots J$, reportado si llega a procesarse dicha tarea. El

Tipo de Tarea al que pertenece la tarea j viene denotado por a_k . El número total de diferentes tipos de tareas viene denotado por A . Los posibles instantes (i) de inicio de la realización de cada tarea o trabajo j vienen definidos por $i = 1 \ldots s_j$. Como comentamos anteriormente cada satélite puede procesar un número específico de clases de tareas. El número total de distintas clases de satélites es denotado por C . El conjunto A_c representa, pues, el conjunto de clases de tareas que pueden ser procesadas por un satélite de la clase c $(c = 1 \text{K } C)$. El conjunto J_c alberga a todas las tareas que pueden realizar los satélites de la clase c . Asumiremos que no existe ninguna clase de satélite dominante dentro del conjunto de clases de satélites, es decir, que no existe ninguna clase de satélite tal que $A_c \subset A_{c'}$ para algunos satélites de la clase c' . El conjunto C_a denota el conjunto de los satélites que pueden ser usados para procesar tareas de la clase a $(a = 1 \text{K } A)$.

A continuación formulamos el modelo OVISP mediante programación. Para ello definimos:

- x_{ijq} : Variables de decisión binaria para denotar si una tarea j , de clase a , es procesada por alguno de los satélites (en concreto el satélite q) que son compatibles, esto es, un satélite perteneciente al conjunto c_a , comenzando dicho procesamiento en el instante i .

  Operativamente iremos asignando variables con subíndices estrictamente crecientes, controlando en todo momento a qué tarea pertenece dicha variable, con qué satélite se corresponde, instante de comienzo de procesamiento, etc.

- p_j : Representa el peso o beneficio derivado de la realización de la tarea j , si ésta llega a producirse durante el horizonte planificado.

Ahora el OVISP puede ser formulado como el siguiente problema entero:

$$Z = Max \sum_{i=1}^{T} \sum_{j=1}^{J} \sum_{q=1}^{C} p_j * x_{ijq} \qquad (3.1)$$

s.a.:

- $\sum_{i=1}^{Sj} \sum_{q \in C_a} x_{ijq} \leq 1$; Para cada trabajo $j = 1 \ldots J$ (3.2)

- $\sum_{j=1}^{J} x_{ijq} \leq 1$; Para cada máquina $q \in C_a$ y cada instante (3.3)

- $x_{ijq} \in \{ 0,1 \}$ (3.4)

Seguidamente pasamos a comentar cada una de las partes de las que se compone el modelo del problema:

- (3.1) La Función Objetivo asegura que la configuración obtenida es tal que el beneficio total es máximo.

- (3.2) Este conjunto de restricciones asegura que cada tarea es procesada a lo sumo una vez. En este caso C_a representa el conjunto de los satélites que pueden procesar la tarea j, de clase a.

- (3.3) En este otro conjunto de restricciones se asegura que cada satélite debe procesar, como máximo, una sola tarea en cada instante de tiempo. Para ello se analiza, para cada satélite, cada uno de los instantes de tiempo del horizonte de planificación, viendo las variables que intervendrían en el instante estudiado si tomaran valor unidad dichas variables.

- (3.4) Son las restricciones binarias para las variables de asignación.

Para aclarar todos los conceptos anteriormente expuestos, a continuación mostraremos un sencillo ejemplo, especificando cada uno de los términos definidos.

Así pues sean, por ejemplo, 2 satélites distintos y 5 peticiones de ejecución de tareas. Supondremos que cada uno de los satélites puede procesar tareas para órbitas distintas, salvo una zona que es común a las respectivas órbitas de ambos satélites. Igualmente

supondremos que, de las 5 peticiones de ejecución de tareas, dos están en la órbita del primer satélite (Tarea 1 y Tarea 2), otras dos (Tarea 3 y Tarea 4) pertenecen a la órbita del segundo satélite, mientras que la última tarea a realizar es común para los dos satélites (Tarea 5).

Supongamos también que el horizonte de estudio es, por ejemplo, de 10 instantes de tiempo, siendo las ventanas de inicio de procesamiento, el tiempo necesario para su ejecución (duración), el peso o beneficio individual reportado, así como el tipo de trabajo al que pertenece cada petición, para cada tarea pedida, por ejemplo los siguientes:

- Tarea 1--> S1 = [0, 1]; duración: 2 instantes; p1 = 15; tipo trabajo: a1
- Tarea 2--> S2 = [0, 1, 2]; duración: 1 instante; p2 = 10; tipo trabajo: a1
- Tarea 3--> S3 = [2, 3]; duración: 3 instantes; p3 = 27; tipo trabajo: a2
- Tarea 4--> S4 = [3, 4, 5]; duración: 1 instante; p4 = 18; tipo trabajo: a2
- Tarea 5--> S5 = [4, 5]; duración: 2 instantes; p5 = 23; tipo trabajo: a3

Por tanto, los términos definidos anteriormente serían:

- Horizonte de trabajo: [0,10]
- Tareas a procesar: J = [T1, T2, T3, T4, T5]
- Clases de Tareas: A = [1, 2, 3]
- Satélites disponibles: [1, 2]
- Clases de Satélites: C = [1, 2]
- A1 = [Tareas Clase 1, Tareas Clase 3]; A2 = [Tareas Clase 2, Tareas Clase 3]
- J1 = [Tarea 1, Tarea 2, Tarea 5]; J2 = [Tarea 3, Tarea 4, Tarea 5]

- C1 = [Satélite 1]; C2 = [Satélite 2]; C3 = [Satélite 1, Satélite 2]

4 RESOLUCIÓN DEL PROBLEMA OVISP MEDIANTE EL OPTIMIZADOR XA

El estudio del **Operational Variable Interval Scheduling Problem** (OVISP) que se ha llevado a cabo en el presente proyecto consta de dos fases:

- En una primera fase se resuelve el problema mediante el uso de las librerías de optimización XA. Se pueden constatar a partir de los resultados obtenidos, que conforme aumenta el grado de complejidad del problema también lo hace el tiempo de resolución. Hemos comprobado en la práctica que el tiempo de resolución de los problemas con alto grado de complejidad es superior a las 12 horas.

- El reducir éstos tiempos de computación tan elevados, ha sido el objetivo del proyecto y, para ello, hemos propuesto en una segunda fase (Capítulo 5) un método aproximado, consistente en un algoritmo de Exploración Dirigida (basado en el algoritmo de Branch & Bound).

4.1. LIBRERIAS DE OPTIMIZACIÓN XA.

La librerías de optimización son cada día más útiles gracias a la mejora de las tecnologías (mayor capacidad de memoria, más velocidad de computación) y de los algoritmos estándar de resolución que forman parte de las mismas. Esta metodología

permite resolver un problema de forma aproximada, mediante condiciones de tiempo de parada. Como ejemplo de algunas librerías de optimización tenemos:

- CPLEX (Más usada actualmente en investigación).
- XA.
- LINGO.
- GAMS (Freeware).
- Solver (uso académico).

El único esfuerzo que requieren en la ejecución es la introducción del modelo en la librería. Para llevarlo a cabo se disponen de diversos formatos y dos formas de hacerlo:

- Mediante un archivo de texto (Offline) [Formatos MPS, SREADSHEET,...]
- Mediante vectores y matrices en memoria (Online).

En los dos próximos subapartados, vamos a enumerar las características más relevantes de las Librerías XA y realizaremos una implementación del modelo. Para hacer más amena la exposición, utilizaremos el sencillo problema que pusimos de ejemplo en el capítulo anterior, el cual constaba tan sólo de cinco tareas, tres clases de tareas y dos satélites, cada uno de una clase distinta. Recordamos los datos de las peticiones de tareas a procesar:

- Tarea 1--> S1i = [0, 1]; duración: 2 instantes; p1 = 15; tipo tarea: a1
- Tarea 2--> S2i = [0, 1, 2]; duración: 1 instante; p2 = 10; tipo tarea: a1
- Tarea 3--> S3i = [2, 3]; duración: 3 instantes; p3 = 27; tipo tarea: a2
- Tarea 4--> S4i = [3, 4, 5]; duración: 1 instante; p4 = 18; tipo tarea: a2

- Tarea 5--> S5i = [4, 5]; duración: 2 instantes; p5 = 23; tipo tarea: a3

4.2. CARACTERÍSTICAS DEL XA.

XA (eXtended Application) es una evolución de sus predecesores LP83/MIP83 ofreciendo un número de ventajas comparativas mejoradas. Su facilidad de uso esta realzada por su formulación detallada. Existen dos modos para la ejecución:

- Mediante archivo texto : XA.EXE
- Para incluir en programas C, Visual Basic o FoxPro: Librerías XA Callable Library

El número máximo de variables, depende de la versión y de la licencia que se este utilizando. La versión general, posee aproximadamente 100.000 variables y 500.000 restricciones.

A continuación pasamos a tratar en tres secciones distintas:

- FORMATO DE ENTRADA.
- PANTALLA DE EJECUCIÓN DEL XA.
- FICHERO DE SALIDA.

4.2.1. FORMATO DE ENTRADA.

Cualquier Editor de Texto o procesador de texto puede ser usado para introducir los datos de entrada para este modo de entrada, con tal que la salida sea compatible con EDLIN (es el editor con el que viene provisto el ordenador). Los informes de entrada son todos de formato libre. Aunque XA no tiene requisitos rígidos de espaciamiento, los archivos problemáticos son más fáciles para leer si las declaraciones son hechas con un espaciado consistente.

La disposición que debe presentar el formato de Entrada XA para el Archivo de Texto consta de las siguientes 4 secciones:

- **Título:** Descripción del problema.
- **Función Objetivo**: Maximizar o minimizar con los valores de los costes de todas las variables.
- Acotaciones de las variables (opcional)
- Restricciones del problema

Cada sección de las anteriormente enumeradas debe aparecer en el orden dispuesto y como máximo una vez en cada problema formulado. Pasamos a continuación a comentarlas con mayor profundidad:

TITULO.

Esta sección es la que encabeza cada problema y es requerida en cada uno que se plantee. El tamaño máximo que puede tener el título es de 128 caracteres. Tiene un formato bastante simple, comienza con "..TITLE" en el primer reglón y a continuación el titulo del problema en el siguiente reglón. El formato presentaría la siguiente forma:

FORMATO:

..TITLE

NOMBRE DEL MODELO

En nuestro ejemplo vendría dado por:

..TITLE

MODELO VSP

FUNCIÓN OBJETIVO.

Esta sección es la que prosigue a la Sección del Título y comienza su declaración en la línea siguiente. En esta sección se lleva a cabo la declaración de:

- OBJETIVO DEL PROBLEMA.
- COEFICIENTES DE COSTES Y VARIABLES DE DECISIÓN.

Pasamos a continuación a comentar con más detalle cada uno de ellos por separado.

Objetivo del problema.

En esta sección se define si nuestro problema es un problema de maximización o minimización. Para ello tan sólo debemos escribir MINIMIZE (si lo que estamos es minimizando) o MAXIMIZE (si lo que pretendemos es maximizar) a continuación de haber escrito "..OBJETIVE".

Coeficientes de costes y variables de decisión.

En esta sección se declaran todas las variables del problema junto con sus coeficientes de costes. Aquellas variables que no entran en la función objetivo pero que serán usadas en otras secciones, como por ejemplo en la parte de las Restricciones, deberán ser declaradas pero con coste nulo para de este modo no tomarlas en consideración.

Las variables se suponen por defecto continuas y mayores que cero, y cuando queramos que algunas variables de decisión sean binarias tan sólo deberemos definirlas entre corchetes. Se pueden usar ( ) y escribir un coste igual para varias variables.

El formato que toma esta sección en el caso de nuestro ejemplo debería ser como el que se muestra a continuación:

FORMATO:

..OBJECTIVE MAXIMIZE

15 [x1] + 15 [x2] + 10 [x3]+ 10 [x4]+ 10 [x5] + 27 [x6]+ 27 [x7]+

+18 [x8]+ 18 [x9]+ 18 [x10] + 23 [x11]+ 23 [x12]+ 23 [x13]+ 23 [x14]

ACOTACIÓN DE VARIABLES.

En esta sección se describen las restricciones que limitan el rango permisible de las variables de decisión. Estas cotas restrictivas pueden ser alternativamente impuestas en la Sección de Restricciones. Describiendo este tipo de restricciones en esta sección, el algoritmo eficiente de acotación es usado en la ejecución posterior del XA. Las reglas que gobiernan son las siguientes:

- Esta sección se considera como una sección opcional, pero, si se encuentra presente, debe aparecer a continuación de la Sección FUNCIÓN OBJETIVO.
- Cuando las cotas no se encuentran fijadas para una variable, la variable tomará un valor comprendido entre 0 e infinito.
- El valor de la parte derecha de la ecuación debe ser una constante.
- Si las variables de decisión se encuentran explícitamente fijadas pueden tomar cualquier valor dentro del rango de los números reales.
- A las variables enteras se les da automáticamente una cota superior.

El formato que presenta esta sección es el siguiente:

FORMATO:

..BOUNDS

X1 >= 10

X2 <=20 >= 2

En nuestro ejemplo las variables no se encuentran acotadas, por lo que no es necesaria la inclusión de esta sección.

RESTRICCIONES.

Esta sección es obligatoria y se encuentra ubicada a continuación de las Secciones de la FUNCIÓN OBJETIVO y de la de COTAS (esta última si es usada) y en ella se debe:

Poner nombre a las restricciones.

El sistema XA da nombre a todas sus restricciones, esto es opcional pero muy recomendable para hacer el informe más fácil de leer y entender.

Si bien, no le causa ningún problema a XA, es recomendable que los nombres de las restricciones no coincidan con los de las variables para evitar confusiones en los informes.

Los nombres pueden ser cualquier cadena de 1 a 8 caracteres terminado por "**:**". Dicho símbolo tan sólo ha de aparecer una vez al comienzo de cada restricción.

Especificar el tipo de restricción.

Una restricción puede ser definida como <=,>=,= o FREE.

Definir los coeficientes independientes.

Debe ser una constante positiva, negativa o la palabra "FREE". Si una restricción posee límites tanto superior como inferior pueden ir especificados ambos en la misma restricción.

Especificar la restricción completamente.

Cada vez que se inicie una restricción, esta debe ser acabada antes de que comience la siguiente restricción. El final de la "..CONSTRAINT SECTION" coincide con el final del archivo.

Por último, para nuestro ejemplo el formato que presenta esta sección es el siguiente:

FORMATO:

..CONSTRAINTS

1: X1+ X2 ≤ 1

2: X3+ X4+ X5 ≤ 1

3: X6+ X7 ≤ 1

4: X8+ X9+ X10 ≤ 1

5: X11+ X12+ X13+ X14 ≤ 1

} RESTRICCIONES (3.2)

6: X1+ X2 ≤ 1 ⟹ RESTRICCIÓN TIPO (3.3) PARA EL SATÉLITE 1 Y EL INSTANTE t=0

...

11: X6 ≤ 1 ⟹ RESTRICCIÓN TIPO (3.3) PARA LA EL SATÉLITE 2 Y EL INSTANTE t=2

DEFINIR LAS HOLGURAS.

Éstas son generadas automáticamente por el programa, por lo que no es necesario ingresar estos datos en el problema.

Como últimas consideraciones formales a tener en cuenta a la hora de redactar el archivo de texto se enumeran las siguientes:

- Se ignoran las líneas en blanco.
- Los comentarios que realicen deben ir precedidos por **"*"**.
- No se distingue entre mayúsculas y minúsculas.
- Los nombres no se pueden cortar al final de la línea.
- La longitud máxima de una línea es de 512 caracteres.

4.2.2. PANTALLA DE EJECUCIÓN DEL XA.

El programa es llamado mediante la línea de comandos:

- > C:\directorioXA\xa.exe *fichero_entrada* output *fichero_salida*

Pero en nuestro caso su ejecución se activa mediante una llamada desde nuestro programa el cual está escrito en lenguaje Visual Basic. La orden de arranque del XA es la siguiente:

- > Shell ("xa.exe entrada.txt listinput no output Salida.res", vbNormalFocus)

A continuación el programa arranca y nos muestra la siguiente interfaz visual en la que podemos observar la evolución de la solución y el número de iteraciones realizadas hasta ese momento:

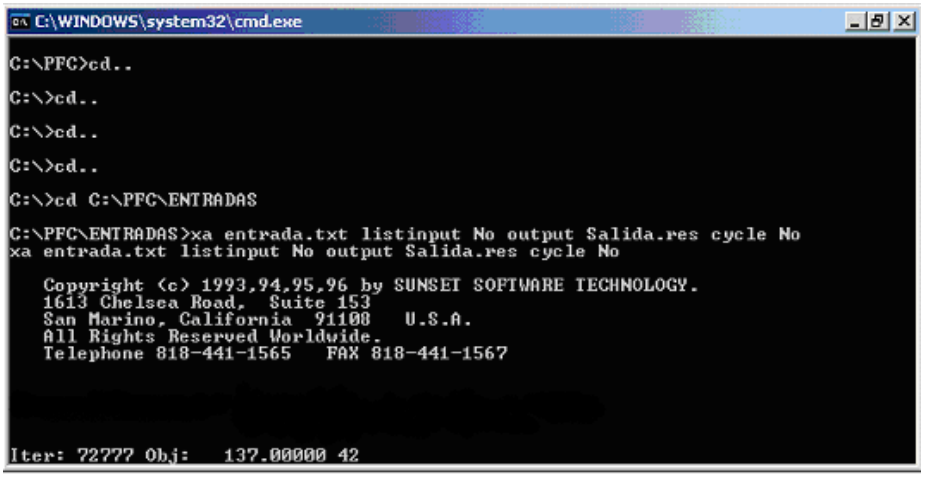


Figura 4-1. Interfaz visual que muestra la evolución de la búsqueda del programa.

En la siguiente imagen se muestra lo que el programa nos da como resultado cuando lo interrumpimos mediante el comando Ctrl+Z:

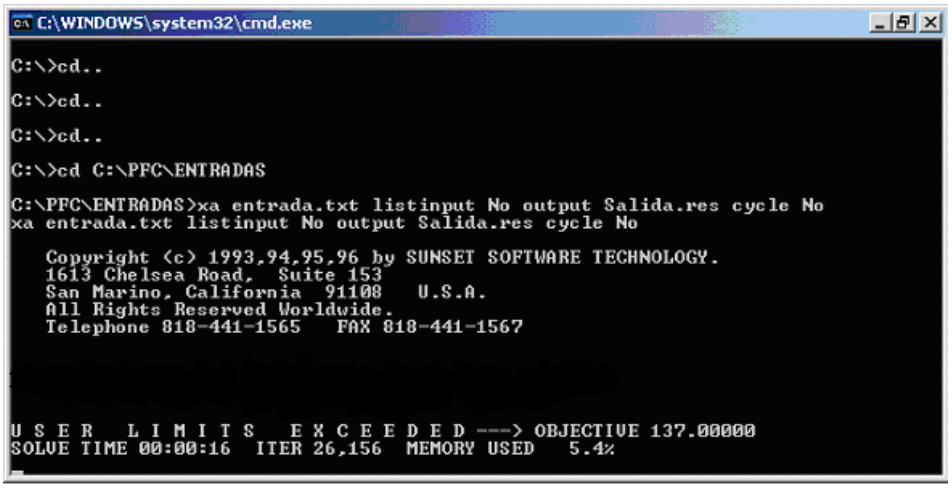


Figura 4-2. Visualización de la solución admisible en el instante de interrupción del programa.

4.2.3. FICHERO DE SALIDA.

Una vez ejecutado el programa, éste devuelve un archivo de salida en el que podemos distinguir tres secciones:

- INFORMACIÓN GENERAL DEL PROBLEMA.
- INFORME DE LA SOLUCIÓN DEL PROBLEMA.
- INFORME DE LAS RESTRICCIONES DEL PROBLEMA.

Pasamos a continuación a mostrar cada una de las secciones y a explicarlas.

INFORMACIÓN GENERAL DEL PROBLEMA.

En este informe es dónde se reflejan todas las características del programa. Para el caso con el que vamos a ilustrar la explicación, el informe que se obtiene es el siguiente:

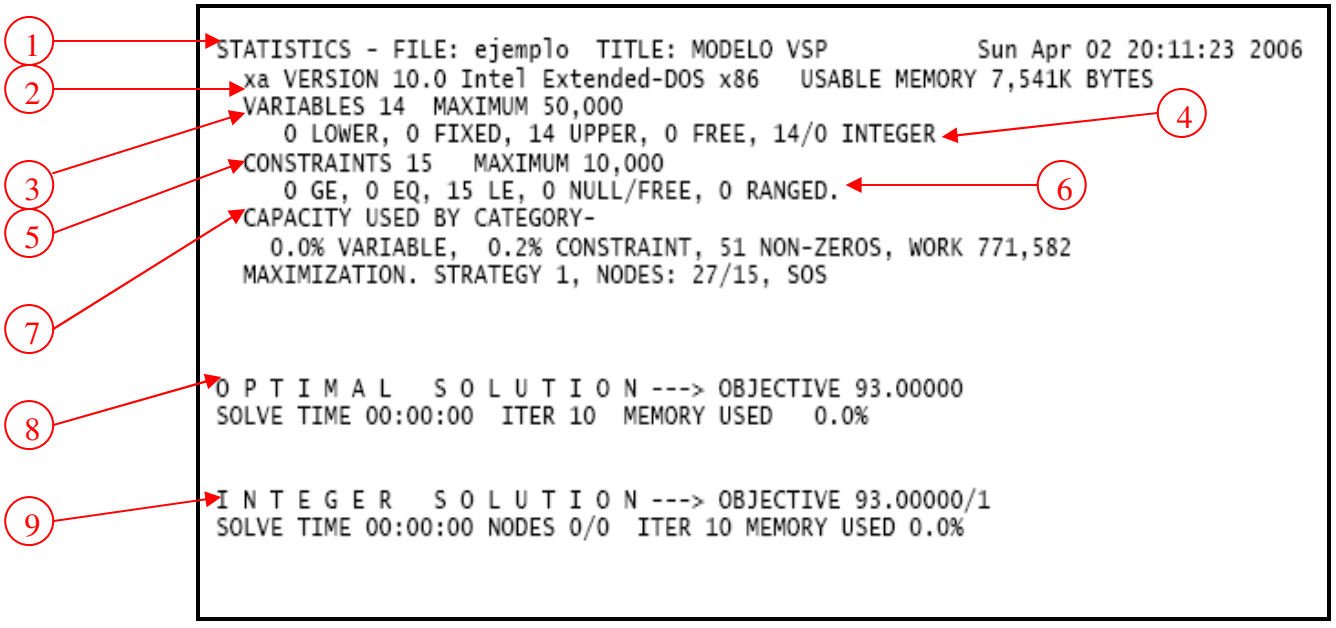


Figura 4-3. Ejemplo de la información general facilitada por el programa una vez resuelto el problema

Pasamos seguidamente a explicar la información que de él se obtiene:

- 1) La primera línea contiene el nombre, el título del problema y la fecha y hora actual.

- 2) La versión del XA de la que estamos haciendo uso y la cantidad de memoria que se esta utilizando.

- 3) Aquí se definen el número de variables que intervienen en el problema.

- 4) Se detalla el número de variables que poseen cota superior, un valor exacto, una cota inferior, un valor cualquiera o han de ser variables enteras.

- 5) En esta línea se especifican el número de restricciones del problema.

- 6) Se detalla el número de restricciones de GE (<=), EQ (=), LE (>=), null/free y "ranged".

- 7) La capacidad usada por categoría.

- 8) Facilita los resultados de la solución óptima (Optimal Solution):

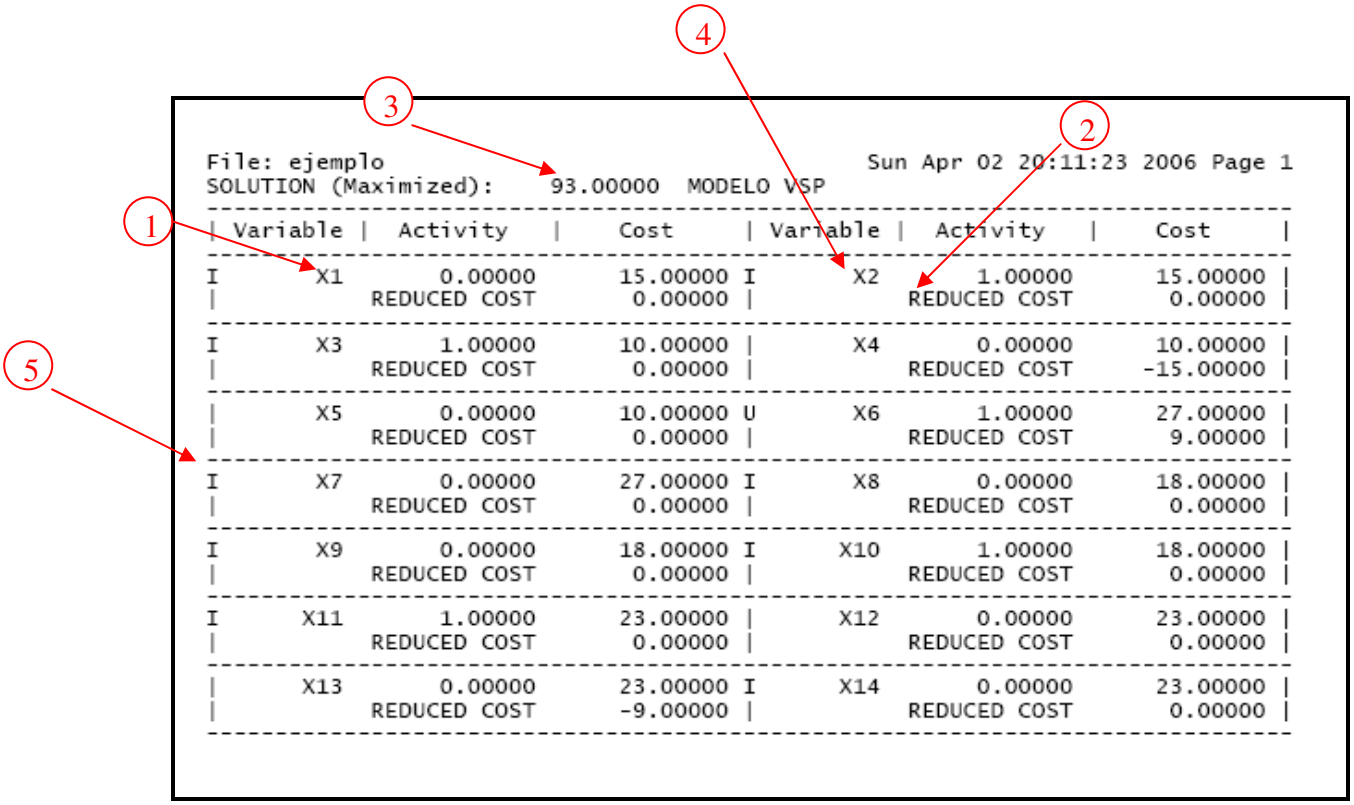
    - Valor de la función objetivo.
    - Tiempo de iteración.
    - Número de iteraciones.
    - Memoria usada.

- 9) Facilita los resultados de la solución entera (Integer Solution):

    - Valor de la función objetivo.
    - Tiempo de iteración.
    - Número de iteraciones.
    - Memoria usada.

INFORME DE LA SOLUCIÓN DEL PROBLEMA.

El informe de soluciones nos da el nivel de actividad de las variables de decisión, el valor que adquiere la función objetivo y los costes reducidos. Para su explicación, hacemos uso de la solución obtenida para nuestro problema, el cual posee la siguiente solución:



File: ejemplo Sun Apr 02 20:11:23 2006 Page 1
SOLUTION (Maximized): 93.00000 MODELO VSP

Variable	Activity	Cost	Variable	Activity	Cost
I X1	0.00000	15.00000	I X2	1.00000	15.00000
	REDUCED COST	0.00000		REDUCED COST	0.00000
I X3	1.00000	10.00000	X4	0.00000	10.00000
	REDUCED COST	0.00000		REDUCED COST	-15.00000
X5	0.00000	10.00000	U X6	1.00000	27.00000
	REDUCED COST	0.00000		REDUCED COST	9.00000
I X7	0.00000	27.00000	I X8	0.00000	18.00000
	REDUCED COST	0.00000		REDUCED COST	0.00000
I X9	0.00000	18.00000	I X10	1.00000	18.00000
	REDUCED COST	0.00000		REDUCED COST	0.00000
I X11	1.00000	23.00000	X12	0.00000	23.00000
	REDUCED COST	0.00000		REDUCED COST	0.00000
X13	0.00000	23.00000	I X14	0.00000	23.00000
	REDUCED COST	-9.00000		REDUCED COST	0.00000

Figura 4-4. Formato que presenta la solución del problema dada por el XA.

A continuación vamos a realizar una explicación de los distintos indicadores que contiene:

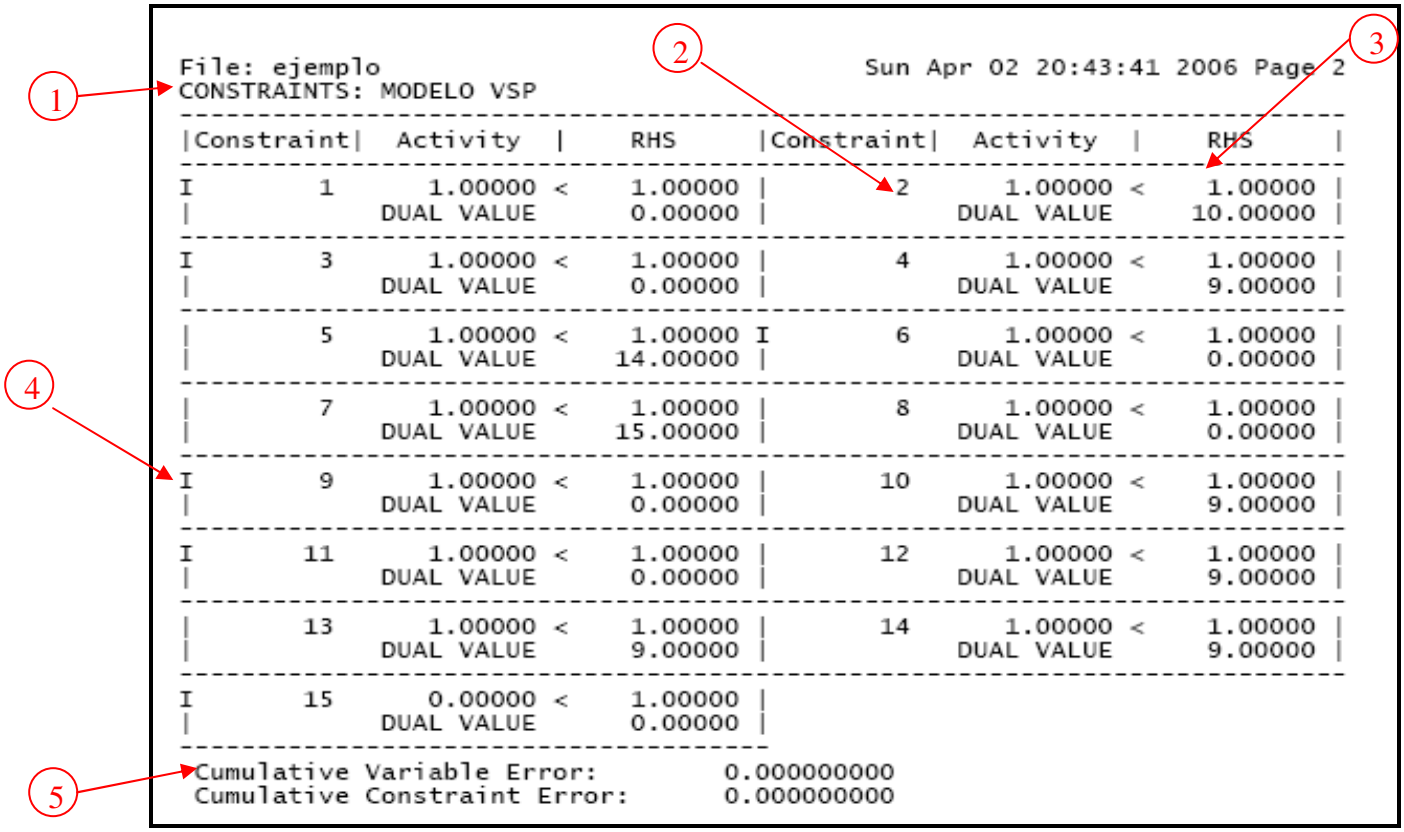
- 1) El nombre de archivo conteniendo la formulación problemática.
- 2) Aquí se refleja la fecha, hora y el número de página.
- 3) Vemos que el objetivo del problema es maximizar y el valor es de **93.00000**.

- 4) La variable x_2 tiene un valor de actividad de 1, un beneficio de 15.0 y un precio reducido de 0.0. Esto quiere decir que la Tarea 1 finalmente sí es procesada y su ejecución comienza en el instante t=1, siendo ésta llevada a cabo por el satélite 1

- 5) La **"I"** que precede al nombre de la variable indica que la variable pertenece a la base, es decir que la variable es básica. Una **" U "** nos indica que la variable es una cota superior.

INFORME DE LAS RESTRICCIONES DEL PROBLEMA.

El informe de las restricciones contiene la actividad de las restricciones basado en la actividad de las variables de decisión. Si nos referimos a nuestro ejemplo obtenemos los siguientes resultados:




```
File: ejemplo                                    Sun Apr 02 20:43:41 2006 Page 2
CONSTRAINTS: MODELO VSP
-------------------------------------------------------------------------------
|Constraint|  Activity  |     RHS     |Constraint|  Activity  |     RHS     |
-------------------------------------------------------------------------------
I         1      1.00000 <    1.00000 |         2      1.00000 <    1.00000 |
|             DUAL VALUE      0.00000 |             DUAL VALUE     10.00000 |
-------------------------------------------------------------------------------
I         3      1.00000 <    1.00000 |         4      1.00000 <    1.00000 |
|             DUAL VALUE      0.00000 |             DUAL VALUE      9.00000 |
-------------------------------------------------------------------------------
|         5      1.00000 <    1.00000 I         6      1.00000 <    1.00000 |
|             DUAL VALUE     14.00000 |             DUAL VALUE      0.00000 |
-------------------------------------------------------------------------------
|         7      1.00000 <    1.00000 |         8      1.00000 <    1.00000 |
|             DUAL VALUE     15.00000 |             DUAL VALUE      0.00000 |
-------------------------------------------------------------------------------
I         9      1.00000 <    1.00000 |        10      1.00000 <    1.00000 |
|             DUAL VALUE      0.00000 |             DUAL VALUE      9.00000 |
-------------------------------------------------------------------------------
I        11      1.00000 <    1.00000 |        12      1.00000 <    1.00000 |
|             DUAL VALUE      0.00000 |             DUAL VALUE      9.00000 |
-------------------------------------------------------------------------------
|        13      1.00000 <    1.00000 |        14      1.00000 <    1.00000 |
|             DUAL VALUE      9.00000 |             DUAL VALUE      9.00000 |
-------------------------------------------------------------------------------
I        15      0.00000 <    1.00000 |
|             DUAL VALUE      0.00000 |
------------------------------------
Cumulative Variable Error:         0.000000000
 Cumulative Constraint Error:      0.000000000
```

Figura 4-5. Formato con el que el programa nos muestra las restricciones del problema.

A continuación vamos a realizar una explicación de la información que podemos obtener:

- 1) El informe de las restricciones con título de **MODELO VSP**.
- 2) El valor de la restricción **"2"** es < de **"1"**.
- 3) El valor dual de la restricción **"2"** es de **"10"**.
- 4) La relación entre la **LHS** y la **RHS** es **<=**. La **" I "** que precede al nombre de la restricción nos indica que esta restricción forma parte de la base. Si en lugar de **"I "** apareciese una **" U "** nos informaría que la restricción es una cota superior.
- 5) Una vez que el **XA** resuelve un problema, todas las variables de decisión y las restricciones tienen ciertas características que han sido medidas. Si estas características medidas no se encuentran entorno a cero, entonces se produce la inestabilidad numérica. Para una mayor exactitud todos los cálculos se realizan con doble precisión, recurriendo para ello al sistema de tolerancia dinámica del **XA**. Es posible resolver problemas largos, pero los errores de computación ocurren cuando se están resolviendo problemas cuya estructura de lectura presenta dificultades numéricas. Cada error de computación es detectado y es recogido en **"Cumulative Variable and Cumulative Constraint Error"**.

4.3. RESOLUCIÓN DEL PROBLEMA OVISP MEDIANTE LIBRERÍAS XA.

Para una mayor comprensión del modo en que tratamos los datos de nuestro problema para que el XA los recoja de forma correcta, pasamos a continuación a explicar sobre el modelo ejemplo que hemos tomado, todos los tratamientos que recibe.

Nuestro ejemplo se compone de:

- Nº de tareas a ejecutar: 5 tareas en total; cada una de ellas con su ventana de instantes de comienzo, duración de su procesamiento y peso o beneficio por realización de la misma. También cabe mencionar que cada una de ellas pertenecerá a una Clase de Tarea, para más tarde poder ver la compatibilidad existente con las distintas Clases de Satélites.

- Número de satélites: 2.

- 3 Clases de Tareas.

- 2 Clases de Satélites.

En la Tabla 4-1 se recogen todas aquellas características que definen a cada uno de las 5 tareas de las que se compone el modelo.

	Tarea 1	Tarea 2	Tarea 3	Tarea 4	Tarea 5
Número de Tarea	1	2	3	4	5
Clase de Tarea	1	1	2	2	3
Duración	2	1	3	1	2
Ventana Inicio	[0,1]	[0,1,2]	[2,3]	[3,4,5]	[4,5]
Peso	15	10	27	18	23

Tabla 4-1. Información que caracteriza a cada una de las cinco tareas del ejemplo.

La Tabla 4-2 recoge la generación de las restricciones referentes a que las tareas tan sólo han de procesarse una sola vez, independientemente de que tarea pueda ser realizada por más de una clase de satélite.

Total Variables	Tarea	Variables afectadas
5	1	X1, X2
	2	X3, X4, X5
	3	X6, X7
	4	X8, X9, X10
	5	X11, X12, X13, X14

Tabla 4-2. Restricciones de procesamiento para cada una de la tareas.

Estableciendo en formato normal las distintas restricciones de la Tabla 4-2 que surgen (Restricciones tipo 3.2), quedan de la siguiente forma:

- $x_1 + x_2 \leq 1$ (4.1)
- $x_3 + x_4 + x_5 \leq 1$ (4.2)
- $x_6 + x_7 \leq 1$ (4.3)
- $x_8 + x_9 + x_{10} \leq 1$ (4.4)
- $x_{11} + x_{12} + x_{13} + x_{14} \leq 1$ (4.5)

} RESTRICCIONES (3.2) PARA CADA TAREA

Como ejemplo, la ecuación (4.5) obliga a que la Tarea 5, que aunque puede ser procesada por ambas clases de satélites (los dos satélites existentes), sólo deberá ser realizada por uno de ellos y en uno de los dos posibles instantes de comienzo posibles.

Por otro lado, en la Tabla 4-3 se muestra la matriz de compatibilidades que existen entre las distintas clases de satélites y tareas.

		Clase de Satélite	
		1	2
Clase de Tarea	1	X	-
	2	-	X
	3	X	X

Tabla 4-3. Matriz de compatibilidad entre Clases de Satélites y Clases de Tareas.

Es decir, la Clase de Tarea 1 es compatible con la Clase de Satélite 1 pero no lo es con la 2. O dicho de otro modo, Una tarea que pertenezca a la clase 1 tan sólo podrá ser procesada por un satélite que pertenezca a la clase de satélite 1. En cambio, un satélite de la clase 1 puede procesar tareas de la clase 1 y de la clase 3.

Las restricciones tipo (3.3) quedan, por tanto, del siguiente modo:

- $x_1 + x_2 \leq 1$ (4.6)
- $x_1 + x_2 + x_4 \leq 1$ (4.7)
- $x_2 + x_5 \leq 1$ (4.8)
- $x_{11} \leq 1$ (4.9)
- $x_{11} + x_{12} \leq 1$ (4.10)

RESTRICCIONES (3.3) PARA EL SATÉLITE 1

- $x_6 \leq 1$ (4.11)
- $x_6 + x_7 + x_8 \leq 1$ (4.12)
- $x_6 + x_7 + x_9 + x_{13} \leq 1$ (4.13)
- $x_7 + x_{10} + x_{13} + x_{14} \leq 1$ (4.14)
- $x_{14} \leq 1$ (4.15)

RESTRICCIONES (3.3) PARA EL SATÉLITE 2

Podemos por tanto concluir que las ecuaciones (4.1), (4.2), (4.3), (4.4), (4.5), (4.6), (4.7), (4.8), (4.9), (4.10), (4.11), (4.12), (4.13), (4.14) y (4.15) establecen las restricciones del modelo.

Mediante la realización de todos los pasos enumerados, podremos obtener todos los datos de entrada necesarios para poder crear el archivo de entrada específico con el que el XA pueda trabajar, tal y como queda reflejado a lo largo del presente apartado, obteniéndose finalmente la solución buscada.

5 RESOLUCIÓN DEL PROBLEMA OVISP MEDIANTE EXPLORACIÓN DIRIGIDA

En este capítulo trataremos de exponer de la forma más simple posible la técnica de resolución del problema planteado de los satélites del EOS mediante un algoritmo de Exploración Dirigida, que planteamos como alternativa cuando el tiempo de resolución mediante empleo directo de librerías XA es demasiado elevado.

Previamente definiremos el concepto de Heurística, así como también se describirán las características del algoritmo de Branch & Bound en que se basa dicho método propuesto como alternativa a las librerías XA.

5.1. MÉTODO DE EXPLORACIÓN DIRIGIDA BASADO EN BRANCH & BOUND.

El método de Branch & Bound es un método exacto de resolución de problemas combinatorios basado en un algoritmo de exploración dirigida de soluciones. Es obviamente un método de enumeración parcial de soluciones que garantiza un buen comportamiento en muchos problemas de optimización.

En este proyecto el algoritmo descrito y desarrollado no consiste exactamente un método de Branch & Bound, sino más bien, y como se comentó en capítulos anteriores, en un método aproximado consistente en una **Exploración Dirigida** de base heurística. Esto es así porque la solución alcanzada puede no ser la óptima del problema, siempre y cuando esté lo suficientemente cercana a la cota superior del problema. La característica de "*suficientemente cercana*" será definida por nosotros (o por el programador) mediante el parámetro denominado *Margen Permitido*, que no es otra cosa que el valor porcentual que permitimos que una solución entera del problema a resolver se aleje de la Cota Superior del mismo.

Otros métodos exactos de resolución de problemas combinatorios son:

Branch & Cut.

Programación Dinámica.

Métodos basados en grafos.

A continuación describiremos de forma general el concepto de Heurística, para una mayor comprensión por parte del lector de los términos empleados en este trabajo:

Heurísticas.

Tanto resultados teóricos como experimentales sugieren que siempre habrá modelos donde los métodos exactos presentan un mal comportamiento y son muy caros de realizar. Cuando esto sucede tendremos que recurrir a métodos casi-óptimos. A estos métodos aproximados se conocen con el nombre de Heurísticas. Una definición de una Heurística sería:

- "Procedimientos simples, normalmente guiados por el sentido común, los cuales logran buenas soluciones, no necesariamente óptimas, para problemas difíciles de forma rápida y sencilla."

Las Heurísticas son muy utilizadas en la práctica, pero su estudio se remonta a un corto espacio de tiempo.

La velocidad computacional no es el único motivo por el que son usadas. Existe un amplio abanico de razones (incluida la anterior) que justifican su uso, entre las que podemos destacar las siguientes:

1) Cuando la optimización requiere gran capacidad de memoria y tiempo de computación.

2) Cuando los datos son poco fiables y resulta poco rentable la optimización exacta.

3) La heurística es fácil de entender. Esto puede ser una razón de peso de su uso y tener una mayor acogida entre los usuarios no expertos.

4) La solución de un problema obtenida mediante una Heurística, proporciona una cota inferior del óptimo de la función objetivo en el caso de un modelo de Maximización. Esto facilita la búsqueda al encontrarse el óptimo entre el intervalo formado por ambas cotas (CS y CI).

Un inconveniente de las Heurísticas es que se modelan para un caso específico; esto no quita que no existan algunos principios generales. Heurísticas aplicadas a casos específicos han sido desarrolladas para un número de problemas de optimización combinatoria, en particular:

- Asignación cuadrática.
- Recocido Simulado.
- Algoritmo Greedy.
- Algoritmos genéticos.
- GRASP.
- Búsqueda Tabú.

5.2. CARACTERÍSTICAS DEL BRANCH & BOUND.

Como hemos comentado en el apartado anterior el algoritmo que se desarrolla y se describe como alternativa a las librerías XA se basa en uno de los métodos de Exploración Dirigida existentes más importantes, llamado en inglés método de Branch & Bound. La forma de representación gráfica del Branch & Bound es mediante un árbol de exploración.

El título del método en sí nos define en qué consiste la forma de proceder:

- **Branch:** Su traducción al castellano es *Ramificación*. Cada subproblema que generamos nace de un subproblema anterior.

- **Bound:** Su traducción al castellano es *Cota* o *Límite*. En este método tomamos dos tipos de cotas, utilizadas como limitantes de la solución óptima, las cuales pasamos a describir a continuación sucintamente, ya que con posterioridad lo haremos con más detalle:

    - Cota Superior (CS): Es el valor que limita la solución de cada subproblema, es decir, que la solución óptima de cada nodo creado será siempre inferior o igual a dicho valor. Su cálculo se lleva a cabo, como se verá más adelante, mediante la relajación del problema.

    - Cota Inferior (CI): Se trata de una solución admisible del subproblema. Dicha Cota Inferior tan sólo es calculada cuando la solución del problema no es entera (nos referimos al valor de la función objetivo) o cuando alguna de las variables de dicho problema toman valores igualmente no enteros. Es utilizada para descartar aquellas ramificaciones de subproblemas que tengan como CS un valor inferior al mayor valor de CI obtenido. Su cálculo se llevará a cabo, como se vera con posterioridad mediante una Heurística Greedy.

La **Exploración Dirigida** se caracteriza a través de varios bloques fundamentales, los cuales pasamos a enumerar y definir:

Relajación del Problema.

Como relajación del problema entendemos el cambiar variables enteras a continuas. De este modo conseguimos que la resolución del problema sea mucho más rápida. Tomamos la solución del problema relajado como Cota Superior del óptimo del problema con variables enteras.

Creación del nodo inicial

La solución relajada del problema original nos servirá, para nuestro algoritmo de Exploración Dirigida, como nodo inicial a partir del que crear el árbol de exploración. En este nodo inicial no tendremos la obligación de procesar ninguna tarea en concreto.

Partición del problema en subproblemas.

A partir de un nodo seleccionado de entre todos los nodos que estén activos se crean dos subproblemas o nodos hijos, cada uno identificado por la fijación de una variable que tiene que tomar forzosamente valor 0 y valor 1, respectivamente. Como veremos en el siguiente apartado, estos problemas no se generan aleatoriamente, esto es, que la variable seleccionada por la que ramificaremos no es elegida arbitrariamente, sino que dichos nodos hijos se crean al ir desarrollando el árbol de exploración a partir del problema original siguiendo unas pautas impuestas de antemano.

Resolución de los subproblemas.

La resolución de cada subproblema (nodo) relajado se realiza usando las librerías XA. Lo único que caracteriza a cada subproblema y lo hace único dentro del árbol de exploración es que a la hora de definir las restricciones, hay que incluir en dichas restricciones aquellas variables que se encuentren fijadas a valor 0 o a valor 1. Una vez resuelto el subproblema y, teniendo en cuenta que dentro del árbol de exploración representa una rama, se verá, según sea su solución, si será necesario podarlo o no.

Selección de subproblemas.

Una vez resueltos los nodos hijos del nodo seleccionado se verá qué nodos han de ser podados y cuáles pasarán a ser candidatos a ser ramificados. De entre todos los nodos que se encuentren activos se seleccionará uno por el que seguir ramificando, siguiendo igualmente unos criterios preestablecidos previamente.

5.3. RESOLUCIÓN DEL PROBLEMA OVISP MEDIANTE EXPLORACIÓN DIRIGIDA (BASADO EN BRANCH & BOUND).

El problema, inicialmente, es tratado del mismo modo que se explicó en la implementación del OVISP mediante el XA. La única diferencia son tres pasos que se realizan, previo a la ejecución del algoritmo basado en el Branch & Bound propiamente dicho, que hacen referencia a las siguientes cuestiones:

- Ordenamiento de todas las variables según correspondan a la realización de cada una de las tareas.

- Ordenamiento de las tareas, priorizándolas de forma que se tiene en cuenta el beneficio reportado por su ejecución, así como la duración del procesamiento de la misma. Esto lo haremos mediante un índice I que relacione estos dos últimos conceptos. Dicho índice I no es otra cosa sino la relación entre el beneficio que reporta una tarea si llega a ser realizada y el tiempo de ejecución, esto es, la duración de dicho procesamiento. Por tanto:

$$Indice = \frac{Beneficio}{Duración} \qquad (5.1)$$

Para el problema tratado en el caso de las librerías XA, y con el objetivo de no desviarnos de dicho ejemplo de aplicación práctica, el nodo inicial que nos servirá para crear el árbol de exploración estará formado por la solución del problema original sin más que convertir a continuas todas las variables del problema, estando el conjunto de variables definidas a 1 formado por el conjunto vacío:

- Variables Definidas a 1 = $\{\varnothing\}$ (5.2)

El árbol de exploración, para el caso que nos ocupa, estaría formado únicamente por el nodo inicial, pues al resolver dicho nodo obtenemos que todas sus variables son ya enteras, por lo que la solución obtenida en dicho nodo es la solución óptima del problema en cuestión.

Para un problema genérico en el que al resolver el nodo inicial se obtuvieran variables decimales (al menos una) el árbol de exploración quedaría del siguiente modo:

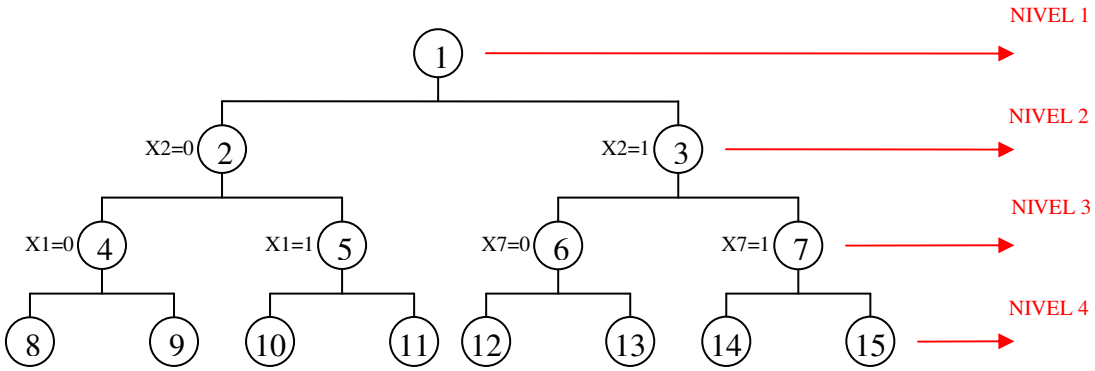


Figura 5-1. Ejemplo de árbol de exploración.

Las características más relevantes que podemos destacar son las siguientes:

- El nivel máximo al que se puede llevar el árbol es el número de variables del problema más uno.

- A partir del nivel dos, incluyéndolo, cada vez que se suba un nivel se ira fijando una variable más, ya sea a valor 0 o a valor 1.

- Los nodos de un nivel serán desarrollados a partir de los nodos del nivel anterior. Los nodos que sean desarrollados a partir de un nodo del nivel anterior recibirán las variables fijadas del nodo del que parten. De cada nodo padre se ramificaran dos hijos, correspondientes a los dos valores posibles que puede tomar la variable seleccionada (variable nueva a fijar).

A continuación pasemos a explicar las pautas que se siguen en la búsqueda de la solución entera óptima de nuestro problema o una solución entera que se aproxime lo suficientemente a la cota superior del problema. Los pasos a cumplimentar en el estudio de cada nodo son los siguientes:

- SELECCIONAR NODO.
- CREAR HIJOS.
- RESOLVER HIJOS CREADOS.
- COMPROBACIÓN DE PODA.
- ITERACIÓN DEL ALGORITMO.

Trataremos cada uno de ellos con más detalle en los siguientes cinco subapartados.

5.3.1. SELECCIONAR NODO.

De entre todos los nodos que se encuentren activos debemos elegir uno por el que tendremos que ramificar y obtener los dos nodos hijos. Los pasos para seleccionar el nodo a ramificar son los siguientes:

- Para todos los nodos creados hasta el momento se seleccionan aquellos que estén activos, los cuales obligatoriamente han de tener una, al menos, de sus variables con valor decimal (entre 0 y 1).
- De entre todos los nodos activos se escogerá aquel nodo que tenga la mayor Cota Superior de todas.

5.3.2. CREAR HIJOS.

Del nodo seleccionado se seleccionará una variable que tenga valor decimal. Por dicha variable **Xi** será por la que se ramificará y se obtendrán los dos hijos correspondientes:

- Hijo 1: **Xi = 0**
- Hijo 2: **Xi = 1**

Para cada hijo se procederá como sigue:

- Identificación del nodo, es decir:
    - Asignar número, el cual es el consecutivo del último nodo creado.
    - Indicar que el nodo es activo (por ahora).
- Se actualiza el número de nodos activos, indicando que existe un nodo más. Dicho nodo se incluirá en una lista donde se almacenan todos los nodos creados durante la ejecución del algoritmo, en la última posición de la misma.
- Se identifican las variables que se van a fijar y su valor; estas variables son:
    - Aquellas que recibo del nodo padre, del cual es engendrado nuestro nodo hijo.
    - La variable fijada en el nodo seleccionado, se le asigna el valor 0 ó 1 según corresponda (en la aplicación del algoritmo se toma, por regla general, que el primer hijo se corresponde con la asignación a valor 0 de la variable seleccionada, mientras que el segundo hijo se corresponde con la asignación a valor 1).

5.3.3. RESOLVER HIJOS CREADOS.

En este punto pasamos a resolver el problema usando para ello las librerías XA. La forma de resolverlo es la misma que la aplicada en el primer método de resolución de nuestro proyecto.

Además, si procede se calculará la Cota Inferior del nodo.

5.3.4. COMPROBACIÓN DE PODA.

En esta sección se estudia la posible poda de cada uno de los nodos activos y, por consiguiente, descartar su posterior ramificación. La poda puede ser debida a tres causas:

1. PODADO POR SOLUCIÓN NO FACTIBLE.

Si se da este caso el nodo es rechazado directamente.

2. PODADO POR SOLUCIÓN ENTERA.

En este caso el nodo ha alcanzado una solución admisible la cual se evalúa para ver si mejora la óptima actual. Si esto ocurriese, el nodo pasaría a ser el óptimo actual del problema.

Tanto si la solución del nodo mejora o no la óptima, el nodo es podado debido a que la solución no puede ser mejorada ya que es entera, por tener todas sus variables con valores enteros (0 ó 1).

3. PODADO POR MALA SOLUCIÓN.

Si la solución obtenida es no entera se plantean dos posibles casos:

- Si la Cota Superior del nodo se encuentra por debajo de la mejor solución actual (denominada Mejor Solución Entera), entonces podamos el nodo por tratarse de una mala solución.

- En otro caso, podaremos el nodo en estudio cuando su Cota Superior se encuentre por debajo de la Mejor Cota Inferior, esto es, la mayor Cota Inferior alcanzada hasta el momento.

5.3.5. ITERACIÓN DEL ALGORITMO.

Una vez se ha llevado a cabo el estudio de poda de todos los nodos activos se procederá a seleccionar un nuevo nodo para ramificar por él. Esto se lleva a cabo comprobando si existe al menos un nodo que no haya sido podado por ninguna de las restricciones anteriormente explicadas.

5.3.6. CÁLCULO DE LA COTA INFERIOR.

A continuación describimos la **"Heurística Greedy"** utilizada en el cálculo de una Cota Inferior para cada nodo del árbol de exploración. Las heurísticas greedy están basadas en una asignación voraz o avariciosa de las máquinas del problema a los trabajos, esto es, en nuestro caso en una asignación voraz o avariciosa de los satélites a las tareas a realizar.

La idea fundamental de la heurística empleada en este trabajo parte de la solución relajada de un nodo, en donde encontramos algunas variables con valor igual a cero, otras con valor igual a uno y, por último, encontramos también variables con valor decimal (0 < Valor de la variable < 1).

Así, partiendo de las variables que toman valor uno (es decir, que refleja que la tarea correspondiente a dicha variable es procesada por algún satélite compatible en la solución relajada del nodo en estudio), lo que hacemos es intentar procesar o "colocar" el mayor número de tareas posibles entre los huecos existentes de todos los satélites disponibles de cada tipo. Dicho de otro modo, partiendo de las variables con valor 1 en la solución relajada del nodo en estudio intentamos convertir a 1 al mayor número posible de las variables que

tengan valor 0 o valor decimal. Ello se hará, como comentamos anteriormente, priorizando las tareas según el índice Beneficio-Duración.

Por tanto, a partir de las tareas que tienen valor unidad en la solución relajada del nodo en estudio se comienza a intentar procesar todas las demás tareas no procesadas, siguiendo las siguientes pautas:

- Se selecciona la tarea de mayor Índice Beneficio-Duración que no sea procesada en la solución relajada del nodo en estudio.

- Una vez seleccionada dicha tarea vamos probando por los satélites disponibles si "cabe" entre los huecos dejados por las tareas que ya son procesadas en cada satélite en estudio. Esto lo hacemos comprobando cada una de las variables que corresponden a la tarea seleccionada una a una, hasta encontrar una de ellas que sí pueda ser convertida a una y además resultando una solución factible.

- Una vez encontrada dicha variable, si existiese, pasamos a intentar procesar la siguiente tarea de mayor índice I. Si no existiese ninguna variable correspondiente a la tarea seleccionada que pudiera ser convertida a 1 pasaríamos, de igual forma, a intentar procesar la siguiente tarea con mayor índice I.

- Una vez hallamos intentado convertir a 1 cada una de las variables correspondientes a cada una de las tareas, calcularemos la Cota Inferior del nodo en estudio simplemente sumando los beneficios individuales asociados a aquellas variables que tengan valor igual a 1.

La ejecución de la heurística se produce, para un nodo en cuestión, cuando tras haber resuelto el problema relajado característico del nodo (que no es más que el problema inicial relajado con las restricciones añadidas de determinadas variables definidas forzosamente a uno o a cero) se comprueba que las variables de dicho nodo no poseen todos sus valores enteros. Dicho de otro modo, la Cota Inferior de un nodo se calculará si existe, al menos, una variable de la solución del nodo que tenga un valor distinto a cero y a

uno. En el caso de que el nodo representara una solución no factible todas sus variables tendrían valores iguales a cero y, por consiguiente, tanto la Cota Superior como la Cota Inferior tomarían directamente valor igual a cero.

La Cota Inferior del caso tratado será utilizada para compararla con la mayor Cota Inferior obtenida durante la aplicación del algoritmo, quedándonos en todo momento con el máximo valor de la misma. Dicho valor lo emplearemos para confrontarlo con la Cota Superior calculada para un nodo genérico, de forma que podremos podar por dicho nodo cuando ocurra que la Cota Superior es mayor o igual que la mejor (mayor) Cota Superior obtenida durante la aplicación del algoritmo.

A partir de este momento se continúa ejecutando el algoritmo, operándose del mismo modo y con los mismos criterios, hasta que se produzca una salida del mismo ya sea porque se ha alcanzado la solución óptima o bien una solución que sea lo suficientemente buena. El grado de bondad de una solución será un parámetro que, como se comentó anteriormente, será definido por nosotros, y que se identificará con el porcentaje sobre la cota superior que queramos que alcance nuestra solución aproximada.

Mediante el método comentado anteriormente se pretende, como se ha dicho, calcular una Cota Inferior admisible que nos valdrá posteriormente para compararla con la Mejor Cota Inferior e intentar mejorarla.

A continuación pondremos un ejemplo de su cálculo.

Supondremos, al igual que el ejemplo del capítulo 3, que tenemos un total de 5 tareas a realizar y 2 satélites disponibles, cada uno de ellos perteneciendo a una clase de satélites distinta. Así, sean ahora dichas tareas requeridas, cada una con sus variables correspondientes:

	Tarea 1	Tarea 2	Tarea 3	Tarea 4	Tarea 5
Número de Tarea	T1	T2	T3	T4	T5
Clase de Tarea	1	1	2	2	3
Duración	5	4	6	3	4
Ventana Inicio	[0,1,2]	[1,2]	[3,4,5]	[2,3]	[4,5]
Peso	15	10	24	18	18
Índice Peso/Duración	3	2,5	4	6	4,5
Variables asignadas	X1, X2, X3	X4, X5	X6, X7, X8	X9, X10	X11, X12, X13, X14

Tabla 5-2. Ejemplo de tareas con petición de ser ejecutadas.

Por ejemplo, si resultara el siguiente árbol de exploración:

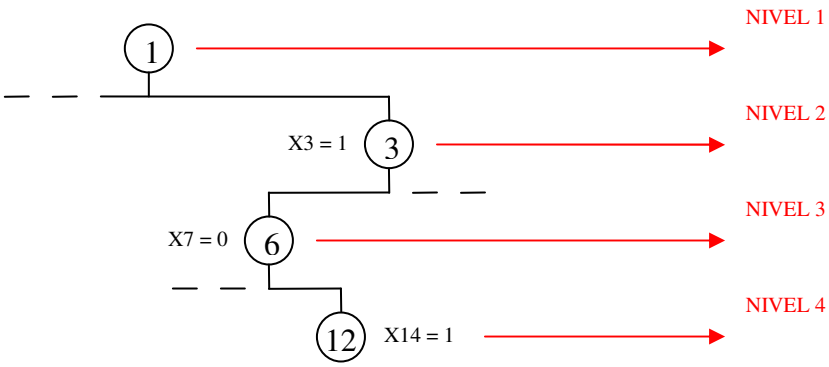


Figura 5-2. Ruta en el árbol de exploración.

En este ejemplo ilustrativo podemos ver que en el nodo 3 la variable X3 se encuentra fijada a 1, mientras que no existe ninguna variable fijada a 0. Por su parte en el nodo 6 se tiene fijada a 1 de nuevo la variable X3, por estar dicha variable fijada en su nodo padre, y además se tiene fijada a 0 la variable X7. Finalmente en el nodo 12 se tienen fijadas a 1 las variables X3 y X14, mientras que están fijadas a 0 de nuevo la variable X7. Todas estas restricciones habrán de ser recogidas en el modelo del algoritmo.

A continuación calcularemos la Cota Inferior para el Nodo 12 a modo de ejemplo clarificador. Para dicho nodo tenemos fijadas, como hemos dicho, las siguientes variables:

- $x_3 = 1$ (5.3)
- $x_{14} = 1$ (5.4)
- $x_7 = 0$ (5.5)

La restricción 5.3 equivale a que la Tarea1 es obligatoriamente procesada y además comienza su ejecución en el instante t=2 en el satélite 1. La restricción 5.4 equivale a que la Tarea 5 es obligatoriamente realizada y además comienza su ejecución en el instante t=5 en satélite 2. Finalmente, la restricción 5.5 equivale a que la Tarea 3 no comienza su procesamiento, si éste llegara a producirse, al menos en el instante t=4.

Lo primero que hay que hacer, antes de nada, es ordenar todas las tareas a realizar, en función al Índice Beneficio/Duración. Dicha ordenación nos conduce a:

	Tarea 4	Tarea 5	Tarea 3	Tarea 1	Tarea 2
Número de Tarea	4	5	3	1	2
Clase de Tarea	2	3	2	1	1
Índice Peso/Duración	6	4,5	4	3	2,5
Variables asignadas	X9, X10	X11, X12, X13, X14	X6, X7, X8	X1, X2, X3	X4, X5

Tabla 5-3. Tareas ordenadas según Índice Beneficio / Duración

Representamos ahora todos los instantes del horizonte de tiempo para los satélites 1 y 2:

SATÉLITE 1

Instante	1	2	3	4	5	6	7	8	9	10
Satélite 1										

SATÉLITE 2

Instante	1	2	3	4	5	6	7	8	9	10
Satélite 2										

Ahora obligaremos al procesamiento de las tareas fijadas en el nodo en estudio (nodo 12):

SATÉLITE 1

Instante	1	2	3	4	5	6	7	8	9	10
Satélite 1		T1	T1	T1	T1	T1				

SATÉLITE 2

Instante	1	2	3	4	5	6	7	8	9	10
Satélite 2					T5	T5	T5	T5		

A continuación trataremos de procesar tantas tareas pendientes de ser procesadas como sean posibles. Ello lo haremos seleccionando las tareas no procesadas de mayor a menor Índice Peso/Duración. Así, observando la Tabla 5-2 la primera tarea candidata a ser ejecutada es la Tarea 4. Al ser de la clase de tarea 2 habría de ser procesada en el satélite 2. Probamos a intentar convertir a 1 la variable X9, correspondiente al instante t=2:

SATÉLITE 2

Instante	1	2	3	4	5	6	7	8	9	10
Satélite 2		T4	T4	T4	T5	T5	T5	T5		

Vemos que sí es factible procesar la Tarea 4 en el satélite 2, comenzando su ejecución en el instante de tiempo t=2. Siguiendo con el procedimiento de cálculo de la Cota Inferior, la siguiente tarea no procesada y con mayor Índice es la Tarea 3. Probando con todas las variables que se corresponden con dicha Tarea (variables X6, X7 y X8) comprobaremos que no es posible procesar dicha tarea en el satélite 1 (el único compatible disponible). Igualmente, si probamos con la última tarea que nos resta, Tarea 2, cuyas variables son X4 y X5, vemos que tampoco puede llegar a ser procesada en el nodo que estamos tratando (nodo 12).

Por tanto la Cota Inferior que obtenemos para este nodo 12 no es otra que la que resulta de sumar los pesos correspondientes a aquellas tareas que logran ser ejecutadas. En este caso:

Tareas procesadas	T1	T4	T5
Peso	15	18	18

Así pues, la Cota Inferior para este nodo será:

$$Cota \quad Inferior = 15 + 18 + 18 = 51 \text{ u.b.}$$

, donde u.b. representa las *unidades de beneficio* obtenidas.

Una vez hayamos calculado la Cota Inferior, pasaremos a compararla con la Mejor Cota Inferior del momento, esto quiere decir, con la mayor Cota Inferior obtenida hasta el momento:

- Si la Cota Inferior obtenida es más restrictiva que la Mejor Cota Inferior del momento, entonces pasa a considerarse como nueva Mejor Cota Inferior.

- Si ocurre lo contrario, entonces no se efectúa ningún cambio manteniéndose la supremacía de la antigua Mejor Cota Inferior.

Tanto en un caso como en otro de los anteriormente explicados, el nodo que estamos estudiando pasa a formar parte de los nodos activos del problema. Una vez estudiados dichos nodos activos y podados los que correspondan, estos nodos serán ramificados, creando un nuevo nivel con nuevos nodos a estudiar.

6 DISEÑO FUNCIONAL

El siguiente apartado se compone de dos secciones, en ellas vamos a llevar a cabo la descripción de los distintos diagramas funcionales seguidos para llevar a cabo el posterior desarrollo de la resolución informática del problema OVISP mediante los dos métodos propuestos.

6.1. DISEÑO FUNCIONAL DE RESOLUCIÓN DEL MODELO OVISP CON LIBRERÍAS XA.

En esta sección vamos a detallar, para el caso del método de resolución mediante el algoritmo XA, cada uno de los distintos diagramas funcionales que constituyen la base para su posterior implementación informática.

A continuación pasamos a mostrar el diagrama de flujo del algoritmo:

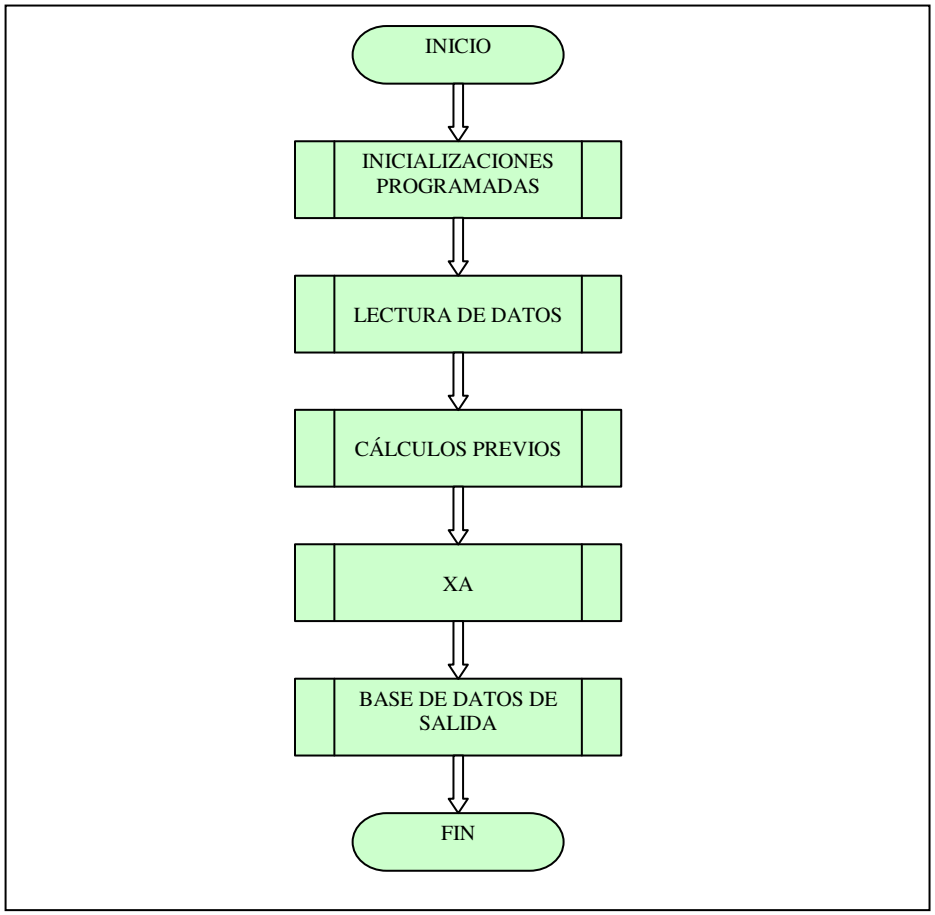


Figura 6-1. Diagrama de Flujo del programa.

Como podemos ver, el diagrama del método XA se compone de cinco módulos:

INICIALIZACIONES PROGRAMADAS

LECTURA DE DATOS

CALCULOS PREVIOS

XA

ESCRIBIR BASE DE DATOS DE SALIDA

En las próximas cinco subsecciones, vamos a describir cada uno de los cinco módulos que son los que componen el diagrama de flujo de nuestro algoritmo.

6.1.1. INICIALIZACIONES PROGRAMADAS.

En este módulo inicial del diseño funcional nos encargaremos de:

- Abrir la base de datos de los resultados para, una vez resuelto el problema, poder guardar los datos tomados como solución.

- Inicialización a cero de todas aquellas variables globales del problema que no lo hagan a lo largo del mismo.

- Obtener la dirección del archivo de texto del cual se obtienen los datos de entrada.

6.1.2. LECTURA DE DATOS.

En este segundo módulo se lleva a cabo la obtención de la información requerida por el programa (XA) referente al problema que se pretende solucionar. El procedimiento en la programación informática que da soporte a esta fase del diseño recibe el nombre de "**LeerDatos**".

Mediante este subprograma el algoritmo obtiene todos los datos necesarios para efectuar todas las operaciones programadas. Los datos que obtenemos se enumeran a continuación:

- Número de tareas a realizar.

- Ventana de posibles instantes de comienzo de cada una de las tareas a realizar, así como la duración de las mismas.

- Peso o beneficio obtenido por la ejecución de cada tarea, si ésta llega a ser ejecutada.

- Número de Satélites.

- Número de Clases de Tareas.
- Número de Clases de Satélites.
- Compatibilidad entre clases de cada tipo.

6.1.3. CÁLCULOS PREVIOS.

En este módulo se lleva a cabo, a partir de los datos recogidos en el anterior, la construcción del modelo matemático que será resuelto en el siguiente. Se compone de los siguientes procesos:

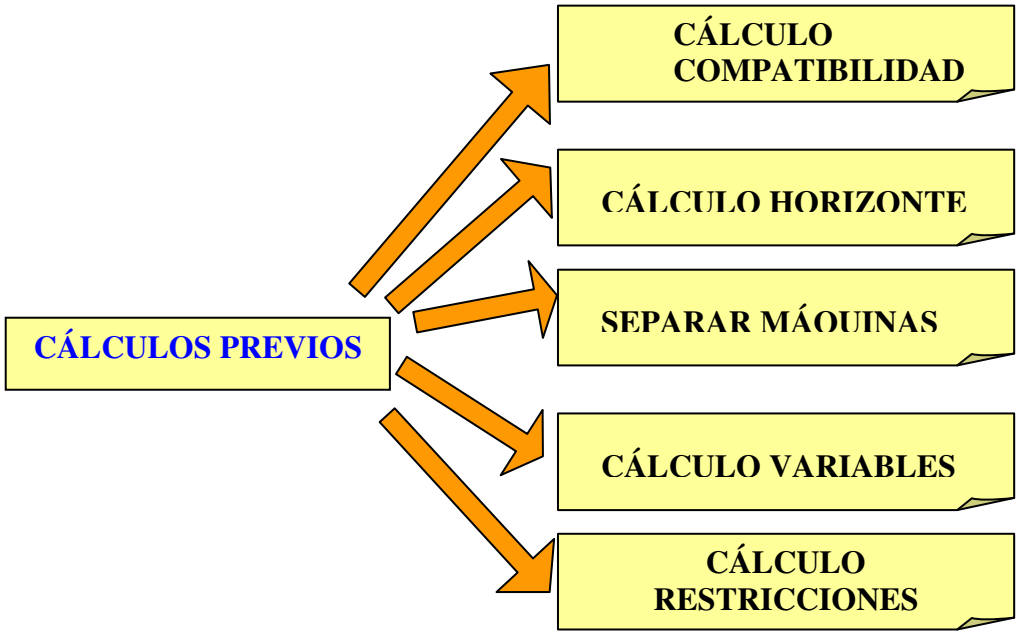


Figura 6-2. Procedimientos de que componen el módulo de Cálculos Previos.

En los cinco próximos puntos vamos a tratar con más detalle cada uno de estos procesos o procedimientos.

CÁLCULO COMPATIBILIDAD.

El subprograma informático que realiza este proceso lo hemos denominado en el programa como "**CalculoCompatibilidad**".

En este proceso es donde caracterizamos una matriz que recogerá la compatibilidad entre los satélites disponibles y las tareas con solicitud de ser realizadas en nuestra programación. Las componentes de esta matriz, tantas como clases de tareas haya, serán vectores que incluirán aquellas clases de satélites compatibles con dicha clase de tarea.

El diagrama de flujo representativo es el siguiente:

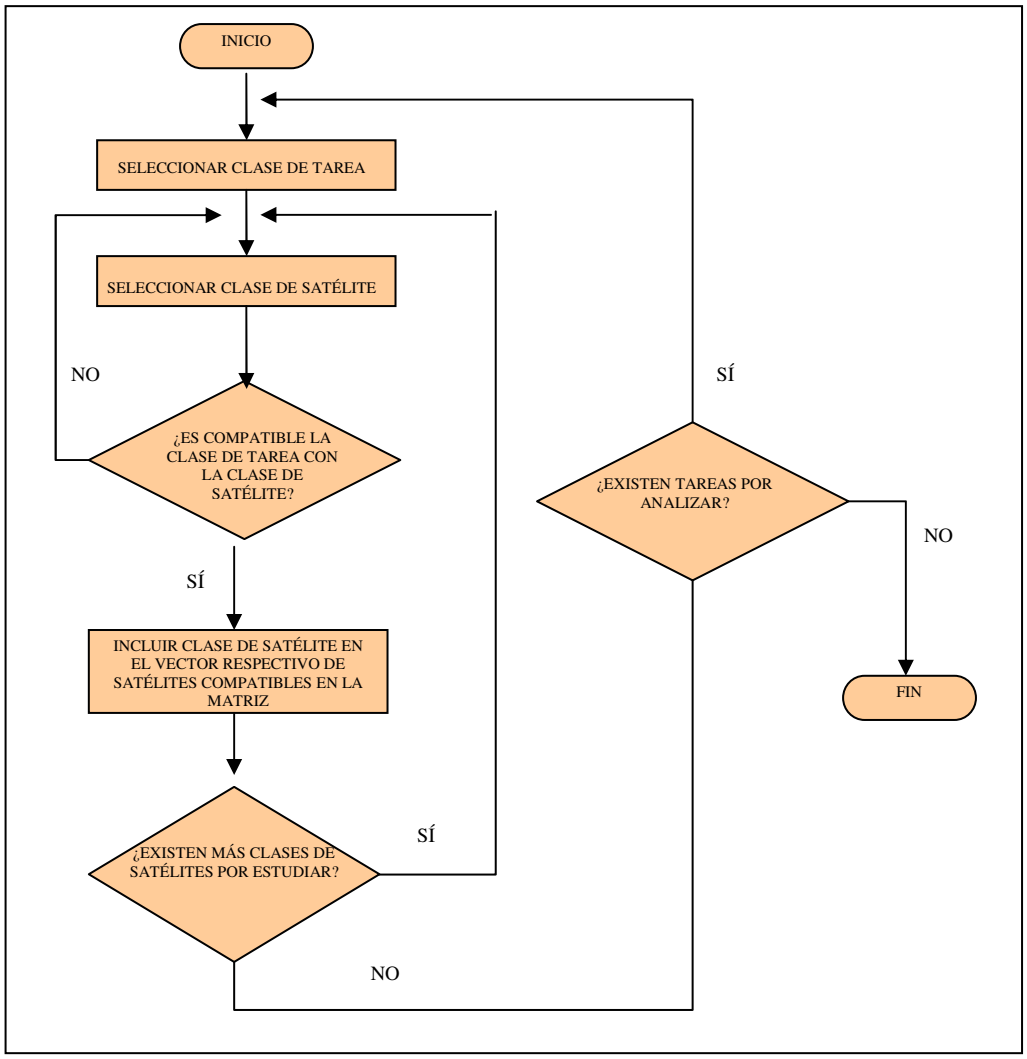


Figura 6-3. Diagrama de Flujo del módulo de Cálculo de Compatibilidad.

CÁLCULOS DE HORIZONTE.

El desarrollo del procedimiento informático que calcula el último instante posible de procesamiento de las tareas, también llamado **Horizonte,** recibe el nombre de **"CalculaHorizonte"**. El horizonte se calcula como la mayor de todas las cantidades que resultan de sumar cada posible instante de comienzo de ejecución de cada tarea más la duración de procesamiento de la misma.

A continuación exponemos el Diagrama de Flujo del programa:

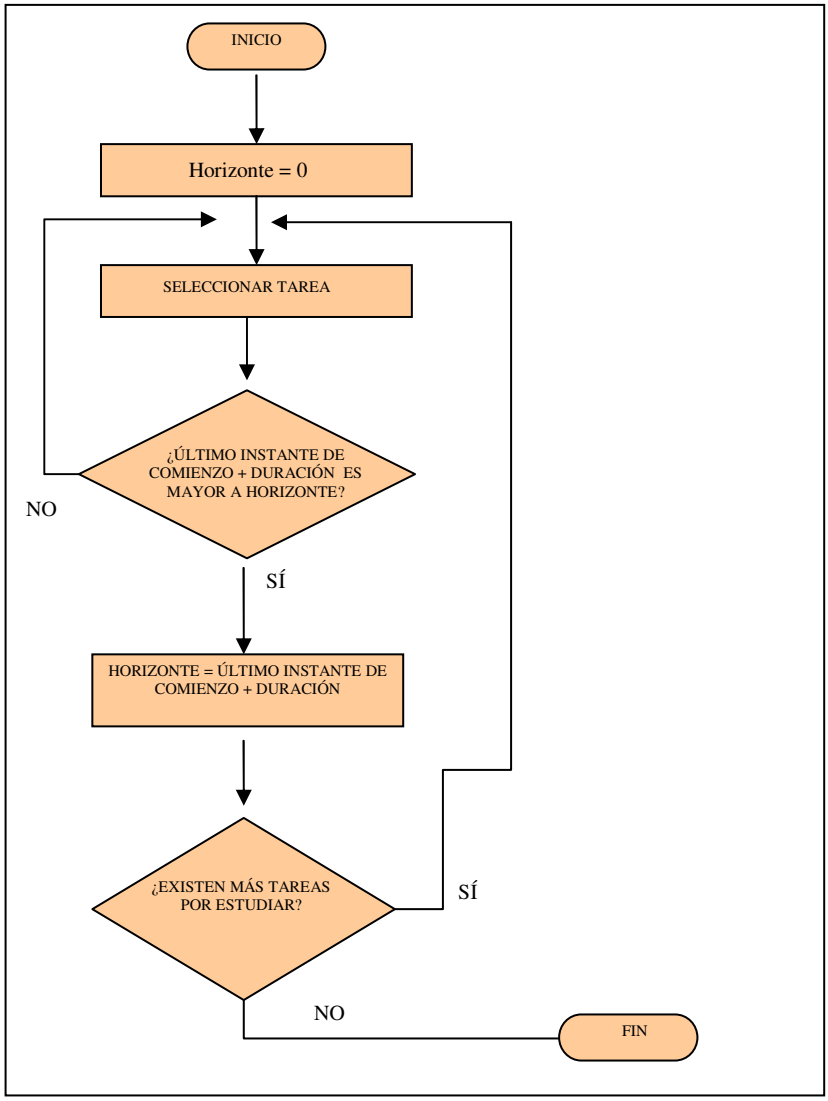


Figura 6-4. Diagrama de Flujo seguido para el cálculo del Horizonte de planificación.

SEPARAR MÁQUINAS

En este paso se busca la obtención del número exacto de satélites de cada clase del que disponemos para el procesamiento de las tareas, a partir del número total de satélites presentes. Así, partimos de una serie de premisas a tal efecto.

Éstas son las siguientes:

I. Si el número total de satélites disponibles (de todas las clases) resulta un múltiplo del número de clases de satélites existentes (sean por ejemplo 6 satélites y dos clases de satélites), entonces cada clase de satélite estará formada por un número n de satélites, idéntico para todas las clases de satélites (en el ejemplo anterior $n = 3$).

II. Si el número total de satélites disponibles (de todas las clases) no resulta un múltiplo del número de clases de satélites existentes (sean por ejemplo 7 satélites y dos clases de satélites), se procederá de la siguiente manera:

1) El número de satélites que sea idéntico al número entero y múltiplo (m) del número de clases de satélites se repartirá de la misma forma que en la premisa I.

2) El número de satélites que coincida con la diferencia entre el anterior número m y el número total de satélites disponibles (d) se repartirá, de forma sucesiva y estrictamente creciente, entre todas las clases de satélites, y empezando por la clase de satélites 1 (la primera). Así, volviendo al sencillo ejemplo anterior de 7 satélites y dos clases de satélites, resultará:

- 7 satélites y 2 clases de satélites
- $m = 6$ (2 x 3 = 6, entero y múltiplo de 2)
- $d = 7 - 6 = 1$
- SATÉLITES CLASE 1: [S1, S2, S3, S4]
- SATÉLITES CLASE 2: [S5, S6, S7]

CÁLCULO VARIABLES.

En este proceso interviene el subprograma **"CalcularVbles"**.

En este paso se obtienen las variables que van a intervenir en nuestro modelo. Cada una de ellas indica la actividad (tarea) que es procesada y con qué satélite se procesa de todos los que resultan compatibles con la clase respectiva de tarea en concreto. Esta designación viene especificada por:

- "x_{ijq}": Indica si la tarea j se realiza, o no, mediante un satélite compatible, en concreto el satélite q, comenzando su procesamiento en el instante i. Se trata de variable de decisión binara en la que :
  - "$x_{ijq}=1$": La tarea i se procesa mediante el satélite j, que es compatible con la clase de tarea de la tarea i.
  - "$x_{ijq}=0$": La tarea i no se procesa mediante el satélite j.

Como se comentó en el apartado 3.2 operativamente iremos asignando variables con subíndices estrictamente crecientes (1, 2, 3,…), controlando en todo momento a qué tarea pertenece dicha variable, con qué satélite se corresponde, instante de comienzo, etc.

EL diagrama de flujo del procedimiento en cuestión que nos calcula las variables del modelo es el siguiente:

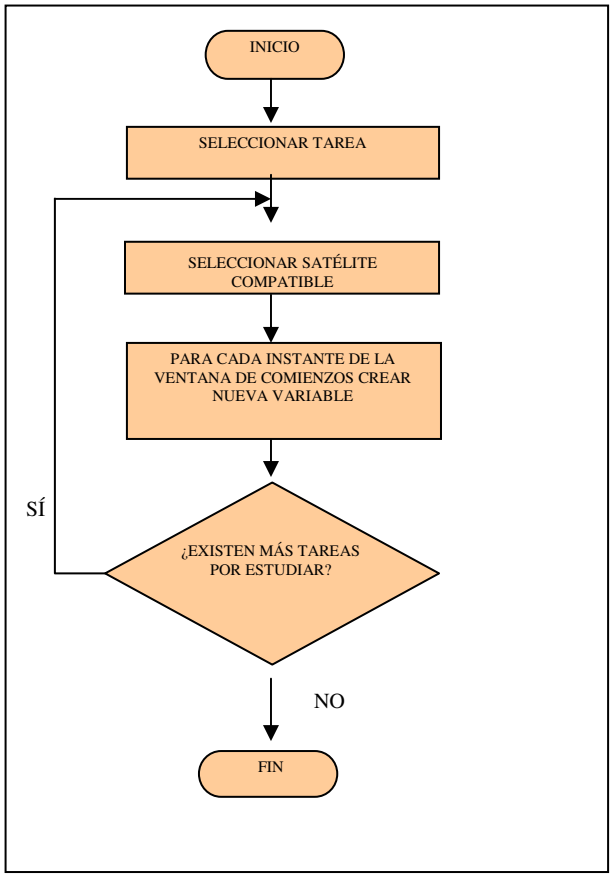


Figura 6-5. Diagrama de Flujo seguido para el cálculo de las variables.

CÁLCULO RESTRICCIONES.

Una vez establecidas las variables del modelo obtenemos también el grupo de restricciones del mismo. El procedimiento del programa implementado recibe el nombre de "**CalcularRestricciones**".

Estas restricciones serán de dos tipos:

1. Aquellas que indican que cada tarea sólo puede procesarse, si esto llega a ocurrir, como máximo una sola vez y por un solo satélite, es decir:

- $\sum_{i=1}^{Sj} \sum_{q \in C_a} x_{ijq} \leq 1$; Para cada tarea $j = 1...J$

2. Aquellas que indican que, en cada instante del horizonte de planificación, un satélite puede procesar a la vez una sola tarea. Recordamos la expresión matemática de esto:

- $\sum_{j=1}^{J} x_{ijq} \leq 1$; Para cada satélite $q \in C_a$ y cada instante

6.1.4. XA.

Este módulo es el cuerpo de todo el modelo, en él se lleva a cabo la resolución del problema que estamos planteando. Se compone de tres partes mediante las que se recogen, de la forma adecuada, los datos de entrada, para su posterior estudio y obtención de la solución óptima. A continuación pasamos a describir cada una de las tres partes con mayor profundidad:

Crear File Format Lenguaje de entrada para Xa.

Esta primera parte la realiza el subprograma **"CrearArchivoEntrada"**.

Mediante este subprograma llevamos a cabo la presentación, en un archivo de texto, del modelo a resolver, en un formato formal denominado "File Format Lenguaje". Este Archivo será el que se usará como entrada para el XA.

Ejecución Del XA.

En esta segunda parte del presente módulo se hace uso del subprograma **"EjecutarXA"**.

Se lleva a cabo la resolución de forma exacta del modelo matemático mediante el programa, basado en la plataforma MS_2.

Lectura de los Resultados del XA.

Por último en la tercera parte, se hace uso del subprograma **"LeerResultados"**.

Se realiza la recogida y su posterior almacenamiento, en unas variables específicas, de los resultados obtenidos del XA .

6.1.5. ESCRIBIR BASE DE DATOS DE SALIDA.

En este último módulo se recurre al subprograma **"EscribirenBD"**.

En él se lleva a cabo el almacenamiento de los resultados obtenidos en una base de datos, de la que en un comienzo recogimos su ubicación.

En concreto, los datos de salida que se han decidido mostrar son los siguientes:

- Nombre del fichero.
- Amplitud máxima de la ventana de comienzos de las tareas del problema.
- Grado de solapamiento de las tareas.
- Número de tareas.
- Número de clases de tareas.
- Número de satélites.
- Número de clases de satélites.
- Número de variables creadas en el problema.
- Número de restricciones creadas en el problema.

- Óptimo del problema relajado (valor no entero).
- Óptimo del problema (valor entero).
- Número de tareas que son procesadas en el óptimo entero del problema
- Tiempo invertido en la resolución del problema mediante librerías del XA.
- Número de iteraciones realizadas por el XA al resolver el problema.

6.2. DISEÑO FUNCIONAL DE RESOLUCIÓN DEL MODELO OVISP CON EL MÉTODO DE EXPLORACIÓN DIRIGIDA.

En esta sección vamos a detallar, para el caso del método de resolución de Exploración Dirigida basado en un método de Branch & Bound, cada uno de los distintos diagramas funcionales que han constituido la base para su posterior implementación informática.

El diagrama de flujo que presenta este método es el que pasamos a continuación a mostrar:

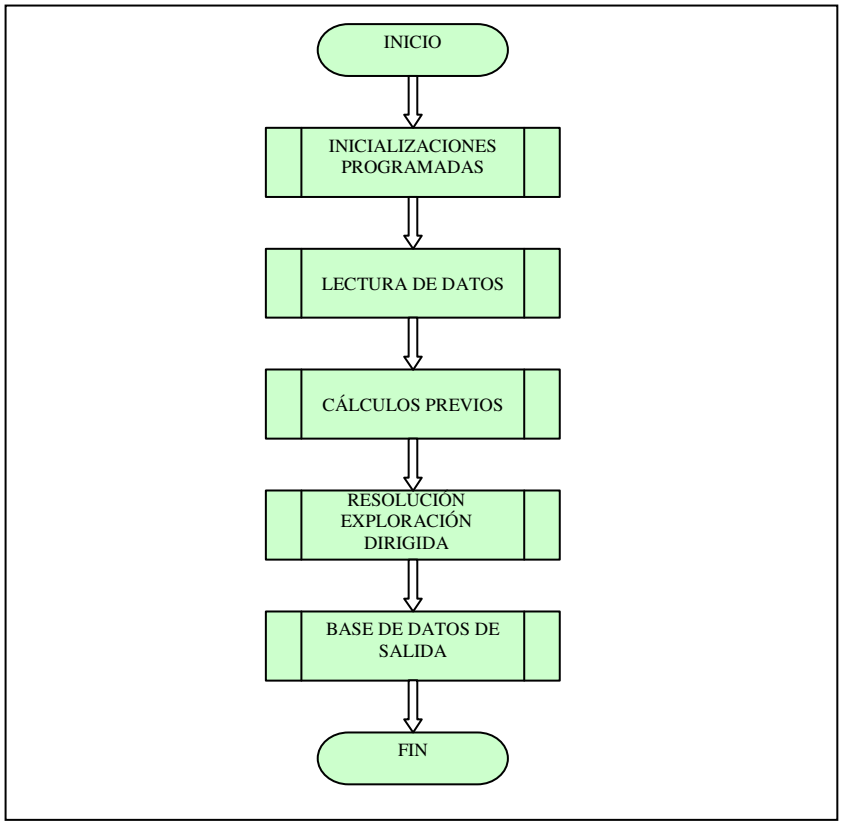


Figura 6-6. Diagrama de Flujo del programa.

Como podemos ver el diagrama del método basado en Branch & Bound es análogo al del XA y se compone también de cinco módulos:

INICIALIZACIONES PROGRAMADAS.

LECTURA DE DATOS.

CALCULOS PREVIOS.

RESOLUCIÓN EXPLORACIÓN DIRIGIDA.

ESCRIBIR BASE DE DATOS DE SALIDA.

Procedemos a continuación a tratar, en secciones separadas, cada uno de los cinco módulos comentados.

6.2.1. INICIALIZACIONES PROGRAMADAS.

En este módulo inicial del diseño funcional nos encargaremos de:

- Abrir la base de datos de los resultados para, una vez resuelto el problema, poder guardar los datos tomados como solución.

- Inicialización a cero de todas aquellas variables globales del problema que no lo hagan a lo largo del mismo.

- Obtener la dirección del archivo de texto del cual se obtienen los datos de entrada.

6.2.2. LECTURA DE DATOS.

En este segundo módulo se lleva a cabo la obtención de la información requerida por el programa (algoritmo implementado de Exploración Dirigida) referente al problema que se pretende solucionar. El procedimiento en la programación informática que da soporte a esta fase del diseño recibe el nombre de "**LeerDatos**".

Mediante este subprograma el algoritmo obtiene todos los datos necesarios para efectuar todas las operaciones programadas. Los datos que obtenemos se enumeran a continuación:

- Número de tareas a procesar.

- Ventana de posibles instantes de comienzo de cada una de las tareas a procesar, así como la duración de las mismas.

- Peso o beneficio obtenido por la realización de cada tarea, si ésta llega a ser ejecutada.

- Número de Satélites.

- Número de Clases de Tareas.
- Número de Clases de Satélites.
- Compatibilidad entre clases de cada tipo.

6.2.3. CÁLCULOS PREVIOS.

En este módulo se lleva a cabo, a partir de los datos recogidos en el anterior, la construcción del modelo matemático que será resuelto en el siguiente. Se compone de los siguientes procesos:

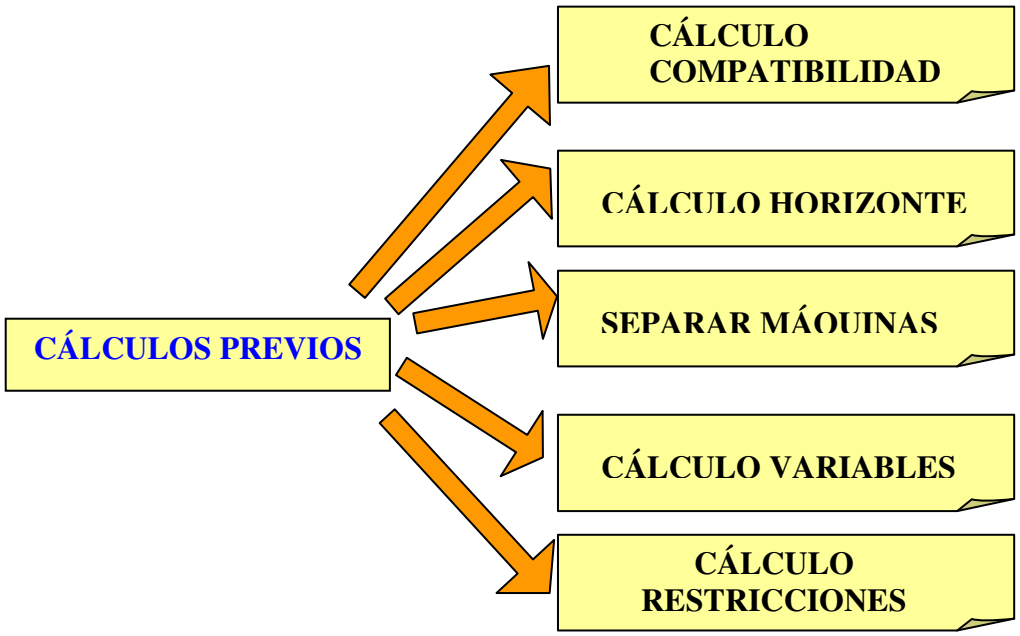


Figura 6-7. Procedimientos de que componen el módulo de Cálculos Previos.

En los cinco próximos puntos vamos a tratar con más detalle cada uno de estos procesos o procedimientos.

CÁLCULO COMPATIBILIDAD.

El subprograma informático que realiza este proceso lo hemos denominado en el programa como "**CalculoCompatibilidad**".

En este proceso es donde caracterizamos una matriz que recogerá la compatibilidad entre los satélites disponibles y las tareas con solicitud de ser procesadas en nuestra programación. Las componentes de esta matriz, tantas como clases de tareas haya, serán vectores que incluirán aquellas clases de satélites compatibles con dicha clase de tarea.

El diagrama de flujo representativo es el siguiente:

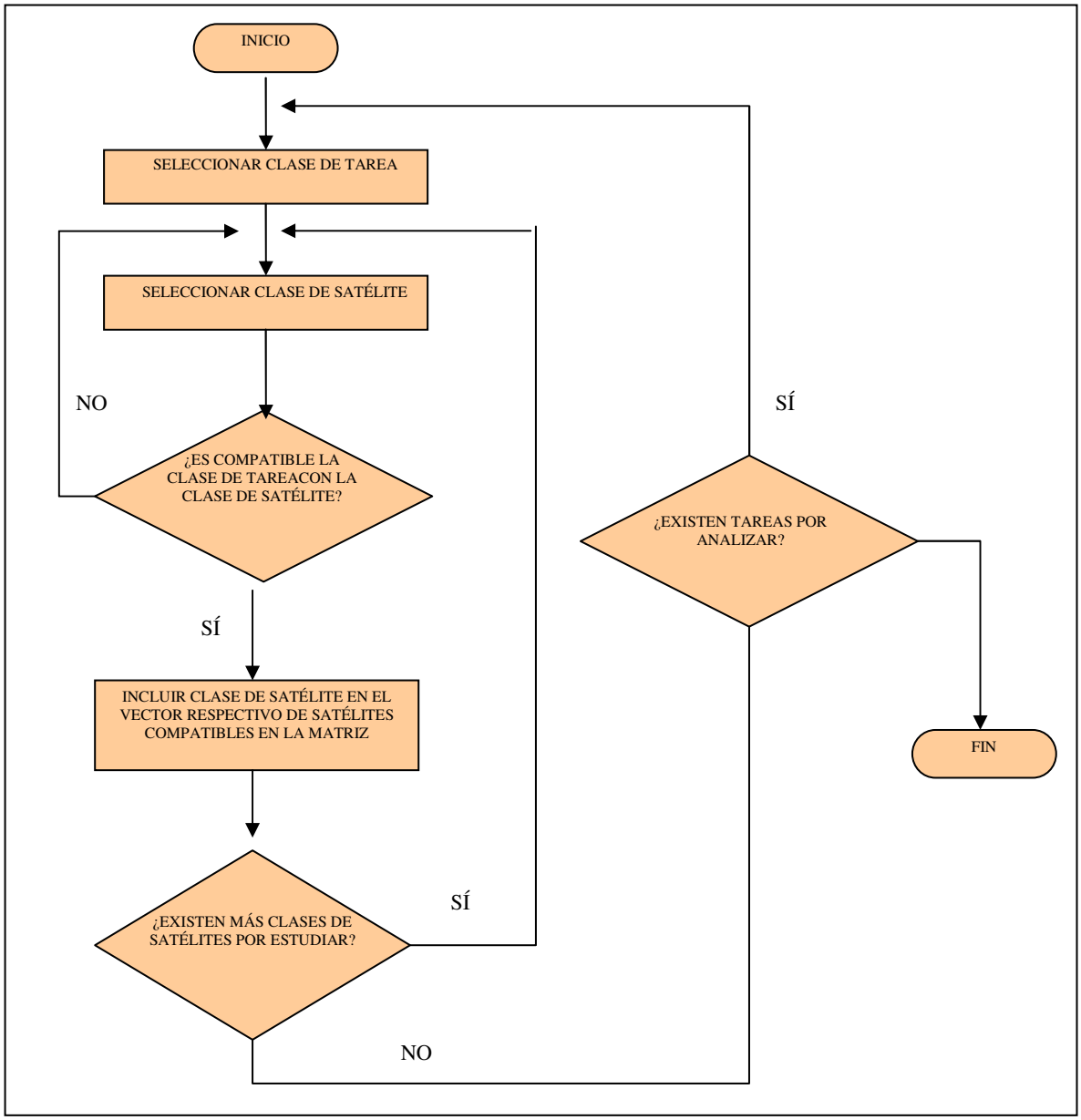


Figura 6-8. Diagrama de Flujo del módulo de Cálculo de Compatibilidad.

CÁLCULOS DE HORIZONTE.

El desarrollo del procedimiento informático que calcula el último instante posible de procesamiento de las tareas, también llamado **Horizonte**, recibe el nombre de "**CalculaHorizonte**". El horizonte se calcula como la mayor de todas las cantidades que resultan de sumar cada posible instante de comienzo de procesamiento de cada tarea más la duración del procesamiento de la misma.

A continuación exponemos el Diagrama de Flujo del programa:

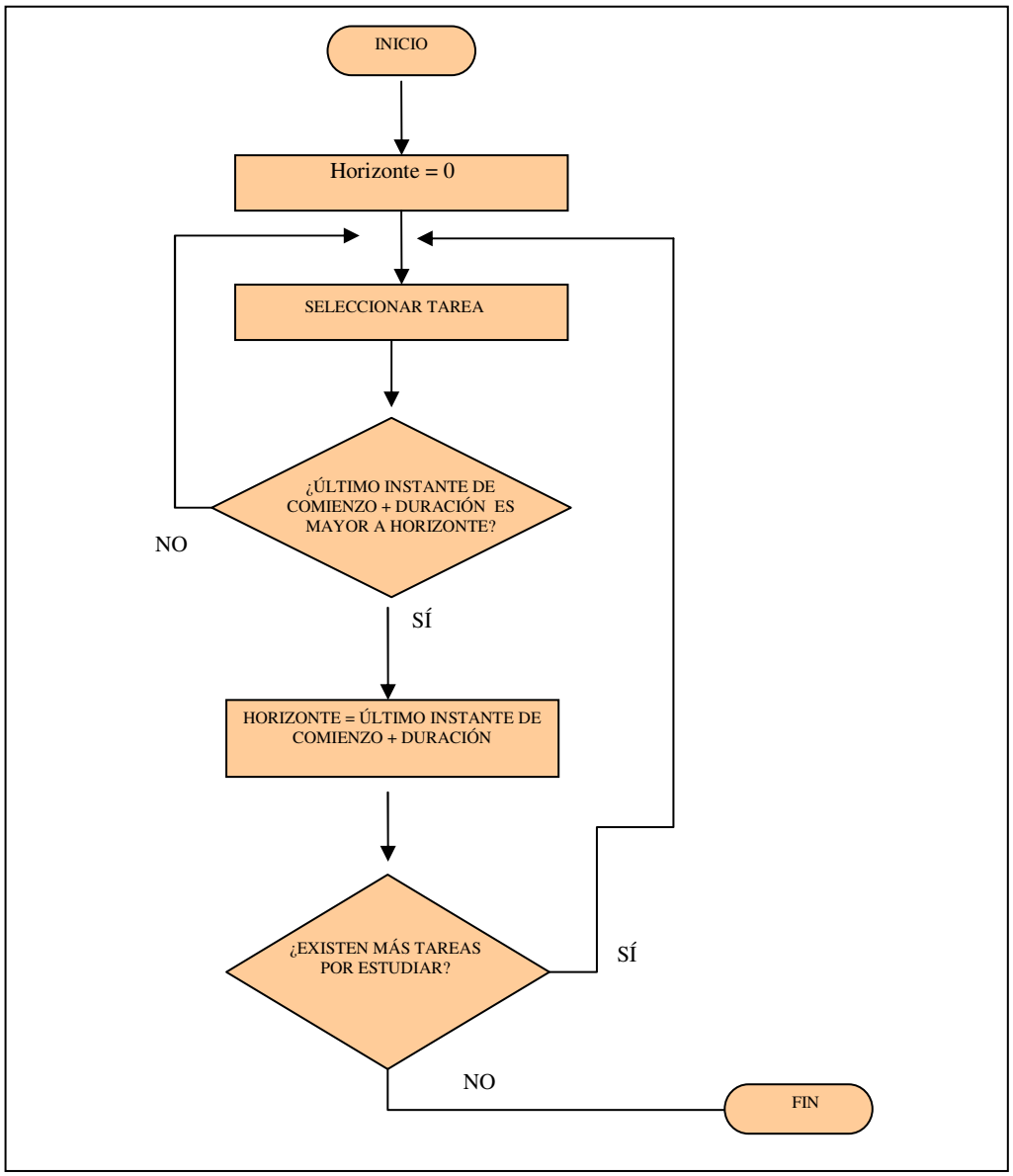


Figura 6-9. Diagrama de Flujo seguido para el cálculo del Horizonte de planificación.

SEPARAR MÁQUINAS

En este paso se busca la obtención del número exacto de satélites de cada clase del que disponemos para la ejecución de las tareas, a partir del número total de satélites presentes. Así, partimos de una serie de premisas a tal efecto.

Éstas son las siguientes:

I. Si el número total de satélites disponibles (de todas las clases) resulta un múltiplo del número de clases de satélites existentes (sean por ejemplo 6 satélites y dos clases de satélites), entonces cada clase de satélite estará formada por un número n de satélites, idéntico para todas las clases de satélites (en el ejemplo anterior $n = 3$).

II. Si el número total de satélites disponibles (de todas las clases) no resulta un múltiplo del número de clases de satélites existentes (sean por ejemplo 7 satélites y dos clases de satélites), se procederá de la siguiente manera:

1) El número de satélites que sea idéntico al número entero y múltiplo (m) del número de clases de satélites se repartirá de la misma forma que en la premisa I.

2) El número de satélites que coincida con la diferencia entre el anterior número m y el número total de satélites disponibles (d) se repartirá, de forma sucesiva y estrictamente creciente, entre todas las clases de satélites, y empezando por la clase de satélites 1 (la primera). Así, volviendo al sencillo ejemplo anterior de 7 satélites y dos clases de satélites, resultará:

- 7 satélites y 2 clases de satélites
- $m = 6$ (2 x 3 = 6, entero y múltiplo de 2)
- $d = 7 - 6 = 1$
- SATÉLITES CLASE 1: [S1, S2, S3, S4]
- SATÉLITES CLASE 2: [S5, S6, S7]

CÁLCULO VARIABLES.

En este proceso interviene el subprograma **"CalcularVbles"**.

En este paso se obtienen las variables que van a intervenir en nuestro modelo. Cada una de ellas indica la actividad (tarea) que es procesada y con qué satélite se realiza de

todos los que resultan compatibles con la clase respectiva de la tarea en cuestión. Esta designación viene especificada por:

- "x_{ijq}": Indica si la tarea j se realiza, o no, mediante un satélite compatible, en concreto el satélite q, comenzando su procesamiento en el instante i. Se trata de variable de decisión binara en la que :
  - $*$"$x_{ijq}=1$": La tarea i se procesa mediante el satélite j, que es compatible con la Clase de Tarea de la tarea i.
  - $*$"$x_{ijq}=0$": La tarea i no se procesa mediante el satélite j.

Como se comentó en el apartado 3.2 operativamente iremos asignando variables con subíndices estrictamente crecientes (1, 2, 3,…), controlando en todo momento a qué tarea pertenece dicha variable, con qué satélite se corresponde, instante de comienzo, etc.

EL diagrama de flujo del procedimiento en cuestión que nos calcula las variables del modelo es el siguiente:

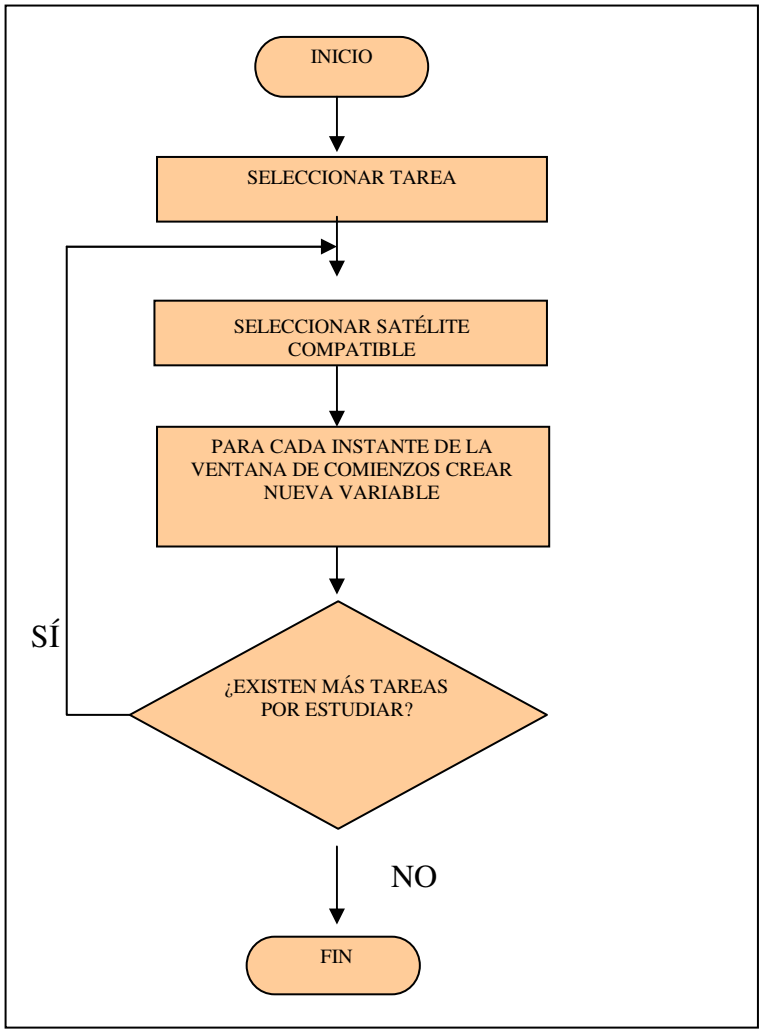


Figura 6-10. Diagrama de Flujo seguido para el cálculo de las variables.

CÁLCULO RESTRICCIONES.

Una vez establecidas las variables del modelo obtenemos también el grupo de restricciones del mismo. El procedimiento del programa implementado recibe el nombre de "**CalcularRestricciones**".

Estas restricciones serán de dos tipos:

1. Aquellas que indican que cada tarea sólo puede procesarse, si esto llega a ocurrir, como máximo una sola vez y por un sólo satélite, es decir:

- $\sum_{i=1}^{Sj} \sum_{q \in C_a} x_{ijq} \leq 1$; Para cada tarea $j = 1...J$

2. Aquellas que indican que, en cada instante del horizonte de planificación, un satélite puede procesar a la vez una sola tarea. Recordamos la expresión matemática de esto:

$$\sum_{j=1}^{J} x_{ijq} \leq 1$$; Para cada satélite $q \in C_a$ y cada instante.

6.2.4. RESOLUCIÓN EXPLORACIÓN DIRIGIDA.

Este módulo es el cuerpo de todo el modelo del algoritmo de Exploración Dirigida de base heurística y que está basado en un Branch & Bound, en él se genera el árbol de exploración del problema que estamos resolviendo, y el estudio de cada una de las ramas del mismo. El módulo presenta la estructura de decisión interna, la cual viene definida por el diagrama de flujo que pasamos a mostrar a continuación:

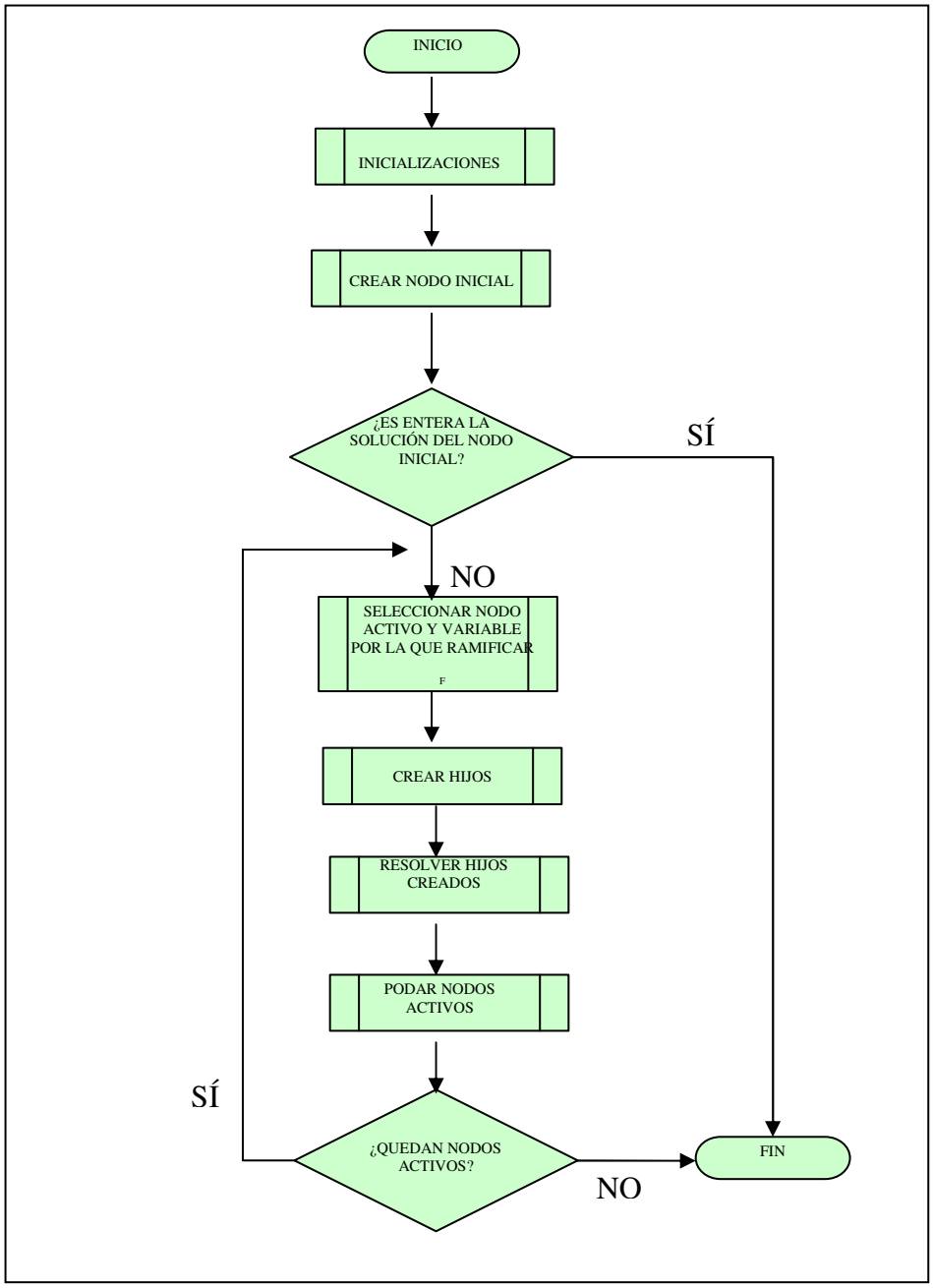


Figura 6-11. Diagrama de Flujo del módulo Resolución Exploración Dirigida.

Como podemos observar el diagrama de flujo esta integrado por los seis submódulos que se enumeran a continuación:

- INICIALIZACIONES
- CREAR NODO INICIAL

- SELECCIONAR NODO
- CREAR HIJOS
- RESOLVER HIJOS CREADOS
- PODAR NODOS ACTIVOS (COMPROBACIÓN DE PODA)

En los próximos seis apartados se explica en qué consiste cada uno de ellos y el modo en que operan.

6.2.4.1. INICIALIZACIONES.

Este primer submódulo se encarga de inicializar los siguientes parámetros:

- Margen Permitido, medido sobre el máximo valor de la función objetivo del problema relajado (cota superior de problema) y que servirá como criterio de parada del algoritmo. Habrá de introducir, pues, y a criterio del usuario final de este método de programación implementado, el máximo margen permitido (por ejemplo $M\arg en\_Permitido = 5\%$).
- Inicializar a "0" el contador de nodos creados.
- Inicializar a "0" el número de iteraciones empleadas.
- Inicializar a "0" la mejor solución entera obtenida, esto es, borrar de memoria el nodo que contiene la mejor solución con valores enteros de las variables.
- Inicializar a "0" el número de nodos activos.

Además, en este submódulo se realizan otras operaciones iniciales que son necesarias, como las siguientes:

- Resolver el problema relajado para ver cuál es la cota máxima que puede alcanzar la función objetivo del problema resuelto.

- Ordenar y agrupar las variables del problema por tarea. Esto lo realizaremos mediante el procedimiento **"OrdenaVariablesPorTrabajo".**

- Ordenar y agrupar las tareas del problema, y en consecuencia las variables asociadas a esas tareas, según el Índice Beneficio-Duración. Esto nos servirá posteriormente para seleccionar la variable por la que ramificaremos. Este paso será realizado en el procedimiento llamado **"OrdenaVariablesPorTrabajo"**.

- Crear los modelos **ModeloHijo1** y **ModeloHijo2**, que son simplemente estructuras de datos idénticas al modelo inicial del problema, llamado **ModeloEntrada**, y que nos servirán para resolver cada uno de los hijos que se crearán como consecuencia de la ramificación por un nodo.

En relación con el primero de los puntos anteriormente mencionados, referido al margen permitido sobre la cota superior del problema relajado para activar el criterio de parada de nuestro algoritmo implementado de Exploración Dirigida, en el siguiente capítulo del presente proyecto (Capítulo 7) se hará hincapié acerca de su importancia, si la tuviera o no, para acortar los tiempos de resolución de los problemas. Allí se verá si al disminuir dicho margen permitido los tiempos de resolución de los problemas tienden o no a aumentar.

Una vez realizado este primer procedimiento, ya estamos preparados para poder afrontar la creación del árbol de exploración.

6.2.4.2. CREAR NODO INICIAL.

En este apartado del programa implementado se busca la obtención del nodo inicial del árbol de exploración y, si procede, comenzar a partir de este nodo calculado la ramificación y, consecuentemente, la creación de nuevos niveles superiores.

El ***Nodo Inicial*** será obtenido a partir de los valores de las variables de la solución del problema original relajado. Esta problema original relajado se resolvió en al anterior apartado de **INICIALIZACIONES**.

Al crear el ***Nodo Inicial*** se inicializará un vector, llamado ***Listanodos***, donde se irán almacenando todos los nodos que se vayan creando en la aplicación de nuestro algoritmo de Exploración Dirigida. Así, este primer nodo inicial ocupará, pues, el primer lugar o componente de ***Listanodos***.

Por otra parte, en este ***Nodo Inicial***, se procederá como sigue:

- Aquellas variables que toman valor "1" en la solución de dicho ***Nodo Inicial*** pasarán a formar parte del conjunto de variables definidas a "1" de este y sucesivos nodos creados a partir de este ***Nodo Inicial***. Este conjunto (vector) de variables definidas a "1", propio de cada nodo, recibe el nombre de ***Variables_a_1***.

- Aquellas variables que toman valor "0" en la solución de dicho ***Nodo Inicial*** pasarán a formar parte del conjunto de variables definidas a "0" de este y sucesivos nodos creados a partir de este ***Nodo Inicial***. Este conjunto (vector) de variables definidas a "1", propio de cada nodo, recibe el nombre de ***Variables_a_0***.

Estas inicializaciones de variables definidas a "0" y a "1" se realizan a fin de acelerar el proceso de obtención de una solución óptima o cercana a la óptima.

6.2.4.3. SELECCIONAR NODO.

En este apartado se procede como ya se indicó en el Capítulo 5. Así pues, de entre todos los nodos que se encuentren activos debemos elegir uno por el que tendremos que

ramificar y obtener los futuros dos nodos hijos. Recordamos cuáles eran los pasos para seleccionar el nodo a ramificar son los siguientes:

- Para todos los nodos creados hasta el momento se seleccionan aquellos que estén activos, los cuales obligatoriamente han de tener una, al menos, de sus variables con valor decimal (entre 0 y 1).

- De entre todos los nodos activos se escogerá aquel nodo que tenga la mayor Cota Superior de todas.

6.2.4.4. CREAR HIJOS.

Recordamos ahora, igualmente del Capítulo 5 cuáles serán las pautas a la hora de crear los hijos a partir del nodo seleccionado:

- Del nodo seleccionado se seleccionará una variable que tenga valor decimal.

- Por dicha variable **Xi** será por la que se ramificará y se obtendrán los dos hijos correspondientes:

  - Hijo 1: **Xi = 0**

  - Hijo 2: **Xi = 1**

Para cada hijo se procederá como sigue:

- Identificación del nodo, es decir:

  - Asignar número, el cual es el consecutivo del último nodo creado.

  - Indicar que el nodo es activo (por ahora).

- Se actualiza el número de nodos activos, indicando que existe un nodo más. Dicho nodo se incluirá en una lista donde se almacenan todos los nodos creados durante la ejecución del algoritmo, en la última posición de la misma.

- Se identifican las variables que se van a fijar y su valor; éstas son:

    - Aquellas que recibe del nodo padre, del cual es engendrado nuestro nodo hijo.

    - La variable fijada en el nodo seleccionado, se le asigna el valor 0 ó 1 según corresponda (en la aplicación del algoritmo se toma, por regla general, que el primer hijo se corresponde con la asignación a valor 0 de la variable seleccionada, mientras que el segundo hijo se corresponde con la asignación a valor 1).

6.2.4.5. RESOLVER HIJOS CREADOS.

El siguiente paso consiste en resolver cada uno de los hijos creados usando para ello las librerías XA. Los pasos que hay que dar en este apartado son los siguientes:

- Preparar el modelo del hijo creado (ModeloHijo1 y ModeloHijo2, según corresponda) para su resolución. EL procedimiento recibe el nombre de **"PrepararModeloXA"**.

- Resolver el nodo de forma relajada, esto es, haciendo todas sus variables continuas. El procedimiento recibe el nombre de **"ResolverXA_Pbrelajado"**.

- Comprobación de si todas las variables son enteras o no; si no son enteras, se procederá a calcular la Cota Inferior de la solución entera óptima del nodo y de la rama que pudiera "colgar" de él. Si resulta una solución entera,

comprobaremos si dicha solución mejora la **Mejor Solución Entera** alcanzada hasta el momento

- Para los futuros hijos, si éstos llegaran a crearse y a fin de acelerar el proceso de alcance del óptimo del problema, se actualizará el conjunto de variables definidas a "1", añadiendo aquellas variables de la solución calculada de ModeloHijo (1 ó 2, según corresponda) que toman valor "1".

La forma de aplicar el comentado procedimiento **"ResolverXA_Pbrelajado"** es la misma que la aplicada en el primer método de resolución de nuestro proyecto, esto quiere decir, que mediante las **Librerías XA**. El diagrama de flujo que representa a este procedimiento "**ResolverXA_Pbrelajado"** es el siguiente:

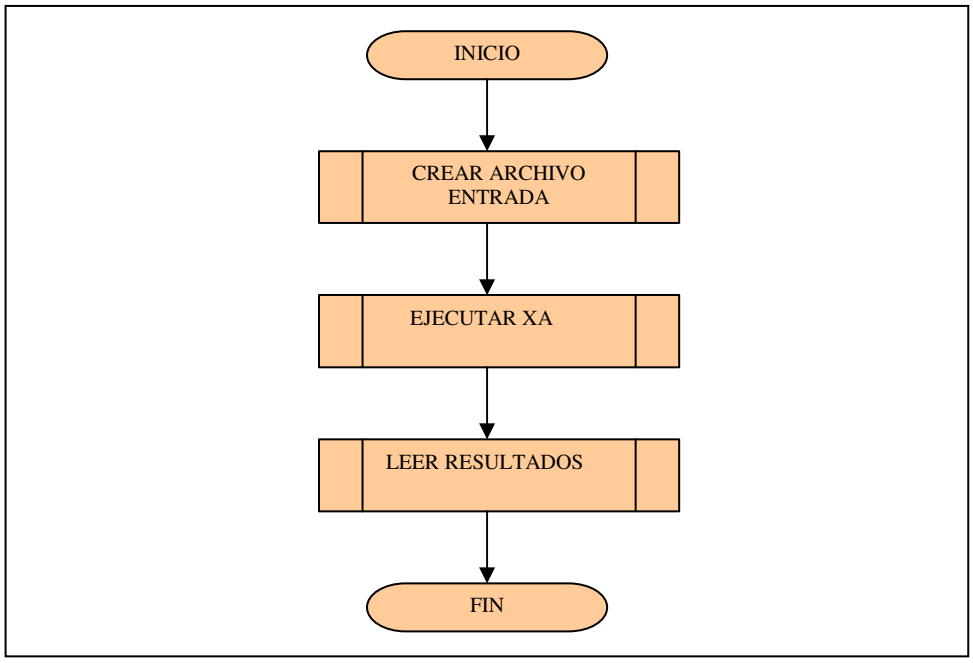


Figura 6-12. Pasos a seguir en la resolución de un nodo.

Como podemos observar, en el diagrama, este procedimiento "**ResolverXA_Pbrelajado**" se encuentra constituido por cuatro procesos, que son los siguientes:

- CREAR ARCHIVO DE ENTRADA.
- EJECUTAR XA.
- LEER RESULTADOS.
- CALCULAR SOLUCIÓN.

A continuación vamos a dedicar un apartado a cada uno de los cuatro procedimientos que componen el submódulo, en los que definiremos en qué consiste cada uno de ellos, y el modo que tienen de operar.

CREAR ARCHIVO DE ENTRADA.

El subprograma informático que lleva el desarrollo de este procedimiento tiene por nombre **"CrearArchivoEntrada"**.

Mediante este subprograma llevamos a cabo la presentación, en un archivo de texto, del modelo a resolver, en un formato formal denominado "File Format Lenguaje". Este archivo será el que se usara como entrada para el XA.

A diferencia del utilizado en el primer método del presente proyecto (en el cual se resolvería aplicando directamente el XA), en éste las variables de decisión pueden ser no enteras.

EJECUTAR XA.

Este segundo procedimiento viene recogido por un subprograma informático, el cual lleva por nombre **"EjecutarXA"**.

En esta parte es donde se lleva a cabo la resolución, de forma exacta, del modelo matemático mediante el programa, basado en la plataforma MS_2.

No presenta ninguna diferencia con el usado en el primer método de resolución del proyecto.

LEER RESULTADOS.

El subprograma informático que da soporte a este procedimiento recibe el nombre de **"LeerResultados"**.

Aquí lo que se realiza es la recogida, y posterior almacenamiento, en unas variables específicas, de los resultados obtenidos del XA.

6.2.4.6. COMPROBACIÓN DE PODA.

Recordamos que en esta sección se estudia la posible poda de cada uno de los nodos activos y, por consiguiente, descartar su posterior ramificación. La poda puede ser debida a tres causas:

1. PODADO POR SOLUCIÓN NO FACTIBLE.

Si se da este caso el nodo es rechazado directamente.

2. PODADO POR SOLUCIÓN ENTERA.

En este caso el nodo ha alcanzado una solución admisible la cual se evalúa para ver si mejora la óptima actual. Si esto ocurriese, el nodo pasaría a ser el óptimo actual del problema.

Tanto si la solución del nodo mejora o no la óptima, el nodo es podado debido a que la solución no puede ser mejorada ya que es entera, por tener todas sus variables con valores enteros (0 ó 1).

3. PODADO POR MALA SOLUCIÓN.

Si la solución obtenida es no entera se plantean dos posibles casos:

- Si la Cota Superior del nodo se encuentra por debajo de la mejor solución actual (denominada Mejor Solución Entera), entonces podamos el nodo por tratarse de una mala solución.

- En otro caso, podaremos el nodo en estudio cuando su Cota Superior se encuentre por debajo de la Mejor Cota Inferior, esto es, la mayor Cota Inferior alcanzada hasta el momento.

A continuación, pasamos a mostrar el diagrama de flujo que representa la toma de decisiones para que se cumpla alguna de las tres causas de poda anteriores.

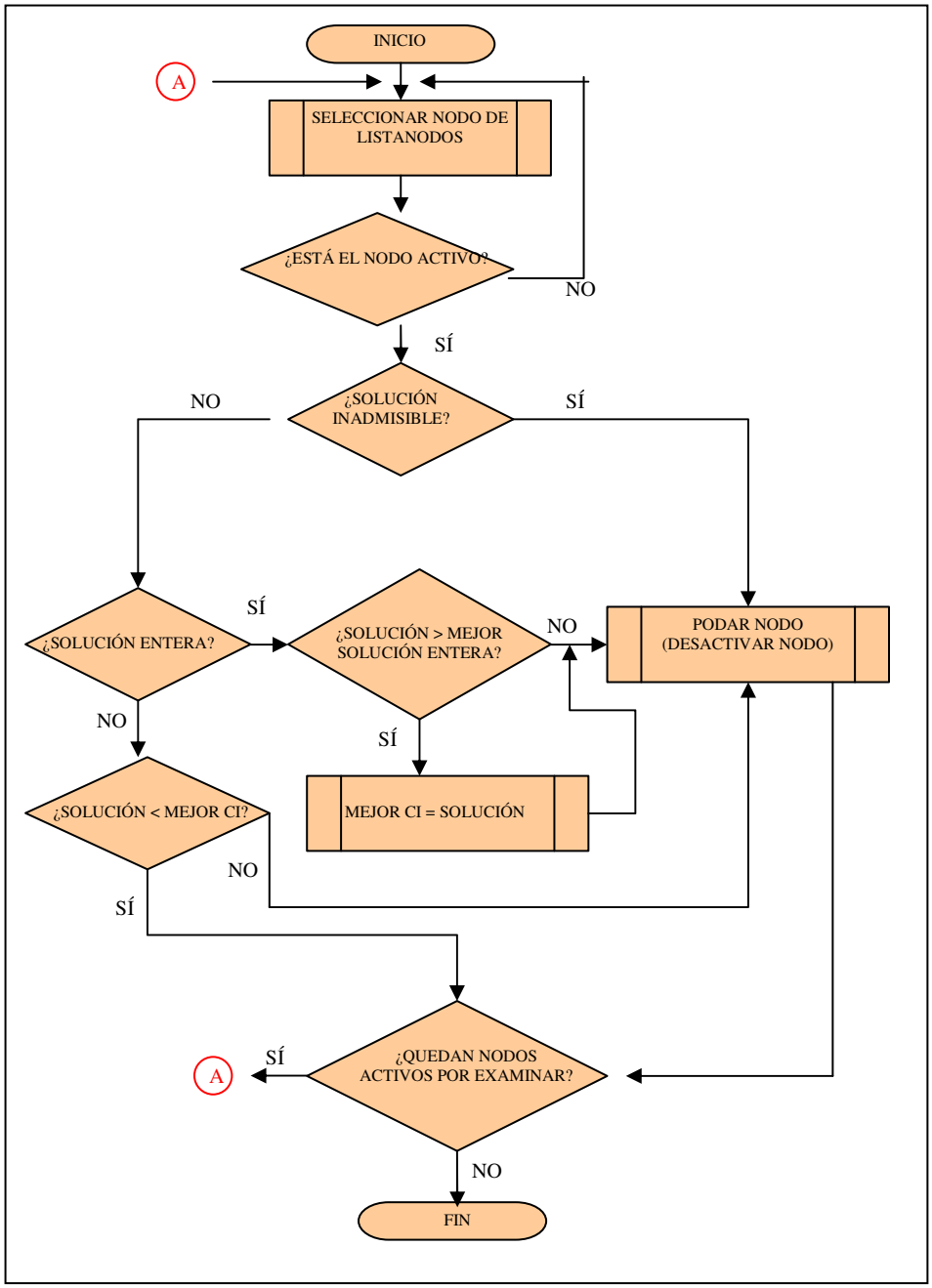


Figura 6-13. Diagrama de Flujo del procedimiento PodarListaNodos.

6.2.4.7. ITERACIÓN DEL ALGORITMO.

Una vez se ha llevado a cabo el estudio de poda de todos los nodos activos se procederá a seleccionar un nuevo nodo para ramificar por él. Esto se lleva a cabo

comprobando si existe al menos un nodo que no haya sido podado por ninguna de las restricciones anteriormente explicadas.

6.2.5. ESCRIBIR BASE DE DATOS DE SALIDA.

En este último módulo se recurre al subprograma **"EscribirenBD"**.

En él se lleva a cabo el almacenamiento de los resultados obtenidos en una base de datos, de la que en un comienzo recogimos su ubicación.

En concreto, los datos de salida que se han decidido mostrar son los siguientes:

- Nombre del fichero.
- Amplitud máxima de la ventana de comienzos de las tareas del problema.
- Grado de solapamiento de las tareas.
- Número de tareas.
- Número de Clases de Tareas.
- Número de satélites.
- Número de Clases de Satélites.
- Número de variables creadas en el problema.
- Número de restricciones creadas en el problema.
- Óptimo del problema relajado (valor no entero).
- Óptimo del problema (valor entero).

- Número de tareas que son procesadas en el óptimo entero del problema.
- Tiempo invertido en la resolución del problema mediante librerías del XA.
- Número de iteraciones realizadas por el XA al resolver el problema.
- Margen Permitido (%) sobre la cota superior para el criterio de parada.

7 RESULTADOS COMPUTACIONALES

Esta sección la vamos a dividir en dos partes. En la primera parte vamos a explicar el método utilizado para la generación de la batería de problemas con los que más tarde experimentaremos. En la segunda parte pasamos a comparar tanto la eficiencia del método exacto mediante **librerías de optimización XA** con los resultados obtenidos con nuestro método de **Exploración Dirigida**, consistente en un método aproximado de exploración dirigida y que está basado en un Branch & Bound, midiendo la evolución del tiempo de ejecución de dicho algoritmo Branch & Bound al variar el margen que se permite, medido éste sobre la cota superior del problema original, para activar el criterio de parada del algoritmo implementado.

7.1. GENERACIÓN DE PROBLEMAS.

Para llevar a cabo un buen estudio es necesario disponer de una batería de problemas que contenga una amplia gama de situaciones posibles. De este modo vamos a obtener un comportamiento del algoritmo más general. Por otro lado, al no dirigir el estudio hacia casos particulares las conclusiones que saquemos a partir de los resultados obtenidos tendrán un fundamento más fuerte.

El método usado para la creación de la batería de problemas fue desarrollado por Kroon et al (1.995). A continuación procedemos a la explicación detallada de dicho método.

Los problemas generados para la posterior experimentación con el algoritmo se obtuvieron de modo aleatorio a partir de la modificación del ratio a priori de utilización ρ diseñado por Kroon et al (1995) para la generación de problemas VSP.

El ratio ρ se trata de un indicador de la carga de trabajo esperada por unidad de capacidad en el espacio de trabajo. Se define como:

$$\rho = \frac{\text{carga de trabajo esperada (uds. tiempo)}}{\text{espacio total de trabajo (uds. tiempo)}} = \frac{n \times \frac{1}{15} D}{T}$$

Ecuación 7-1. Ratio de utilización.

En donde:

- D indica la duración máxima de la realización de una tarea.
- T representa la longitud del horizonte de planificación.
- n denota el número de tareas.

En este ratio, cabe destacar que la carga de trabajo es independiente del número de satélites, lo cual repercute en que el promedio de tareas procesadas depende del número de satélites que se han utilizado.

Para la generación de la batería se han tomado tres valores para el ratio:

- Ratio para baja utilización (ρ=1.5): corresponde a una carga de trabajo de 0.8
- Ratio para utilización media (ρ=3): corresponde a una carga de trabajo de 1.4
- Ratio para utilización alta (ρ =6): corresponde a una carga de trabajo de 2.0

El horizonte de planificación considerado es de T = 1000 unidades de tiempo e instancias con n = 25, n = 50, n = 75, n = 100 y n = 200 tareas. El parámetro *D* es calculado de acuerdo a los valores asignados en la fórmula del ratio.

El Número de Satélites disponibles estudiados varían desde m = 4, m = 8 y m = 16.

Respecto al número de Clases de Tareas y Clases de Satélites, tres diferentes matrices *L* irreducibles fueron usadas en los experimentos, con dimensiones L_{3x2}, L_{3x3} y L_{3x4}. Es decir, se modelan problemas con 2, 3 y 4 Clases de Satélites y problemas únicamente con 3 Clases de Tareas, debido a la enorme complejidad de resolución que supondrían problemas de valores mayores para dicho parámetro Clases de Tareas.

A continuación mostramos una tabla en la que se expresa las combinaciones (*n*, *L*) consideradas para cada uno de los valores designados para ρ en la generación de problemas.

	L_{3x2}	L_{3x3}	L_{3x4}
n=25	1	0	0
n=50	1	1	0
n=75	1	1	1
n=100	1	1	1
n=200	1	1	1

Tabla 7-1. Combinaciones (n, L) consideradas.

Cada una de las tareas es aleatoriamente asignada a una Clase de Tarea.

Para cada una de las tareas J_j, el tiempo de proceso t_j es obtenido aleatoriamente dentro del intervalo (0,*D*), y su instante de inicio s_j es generado igualmente de forma aleatoria dentro del intervalo (0, T-t_j).

Por otra parte el peso w_j se obtiene también de forma aleatoria en el intervalo comprendido entre 0 y 100.

Así pues, y partiendo de las tres combinaciones del ratio ρ consideradas así como de las combinaciones de la Tabla 7-1, se generan un total de 60 problemas, cantidad suficiente para recoger los distintos casos posibles. La resolución de estos 60 problemas es lo que nos servirá para poder comparar las dos formas, vistas en este proyecto, de atacar el problema de planificación de tareas de los satélites del observación terrestre.

De los 60 problemas resueltos, 20 de ellos corresponden a cada uno de los tres ratios de utilización empleados y a su vez, repartidos a su vez entre los 5 tamaños distintos de Números de Tareas, a razón de 4 problemas para cada tamaño estudiado. Además, el estudio de la Exploración Dirigida se realizará para un valor del Margen Permitido (MP) del 3%, medido éste, y como se comentó anteriormente, sobre la cota superior global de cada problema. Comprobaremos, posteriormente, la influencia de este parámetro, si la tuviera, en la rapidez con la que el algoritmo de Exploración Dirigida alcanza la solución, exacta o aproximada, del problema en cuestión.

7.2. RESUMEN DE RESULTADOS.

En este apartado vamos a llevar a cabo dos estudios:

- En una primera parte realizaremos la comparación de los resultados obtenidos mediante la resolución de los problemas con librerías de optimización XA y con la Exploración Dirigida. Esta comparación nos va a permitir tener una idea del nivel de calidad de las soluciones obtenidas mediante la Exploración Dirigida que proponemos.
- En la segunda parte del apartado veremos cómo evoluciona el tiempo de ejecución del algoritmo de Exploración Dirigida al variar el Número de Clases de Satélites.

El estudio experimental fue llevado a cabo en un Intel Pentium M Centrino 1,73 Ghz.

Es necesario decir, además, que ambos métodos de resolución presentados en este proyecto (Librerías XA y Exploración Dirigida basada en Branch & Bound) fueron implementados en lenguaje de programación Visual Basic.

Podemos observar en las soluciones que la resolución de los problemas mediante **librerías XA** resulta más costosa, computacionalmente hablando, y compleja a medida que aumentan el Número de Clases de Satélites, el Número de Tareas, el Número de Satélites y el Grado de Solapamiento.

7.2.1. COMPARACIÓN DE TIEMPOS ENTRE LÍBRERIAS XA Y EXPLORACIÓN DIRIGIDA.

La rapidez de computación de la **Exploración Dirigida** reside en que los niveles requeridos de estudio del árbol de exploración son de bajo nivel. Los tiempos que aparecen en las tablas vienen referidos en CPU-segundos y las celdas en la que aparece la palabra "**sobrecarga**" o bien "**> XX h**" nos vienen a indicar que el tiempo de resolución sobrepasaba en todos los casos las "XX" horas o bien se producía sobrecarga de memoria. Estos fallos de resolución se han dado en los casos de 100 y 200 tareas, así como respectivamente 3 y 4 Clases de Satélites.

A continuación vamos a exponer, en tres tablas una para cada uno de los tres grados de solapamiento, los tiempos invertidos para resolver los problemas usando **Librerías XA** y el método de **Exploración Dirigida** basado en Branch & Bound. Igualmente presentaremos la función objetivo alcanzada mediante XA, así como el error cometido en la solución de la Exploración Dirigida.

Para cada problema resuelto indicaremos los siguientes parámetros:

- **Nº Tareas**: Número de tareas a procesar.
- **Nº Sát**: Número total de Satélites disponibles.
- **CT**: Número de Clases de Tareas.
- **CS**: Número de Clases de Satélites.
- **V**: Número de Variables del problema.

- **R**: Número de Restricciones del problema.

- **XA**: Representa el tiempo invertido en la resolución, de forma exacta, del problema en cuestión usando el método XA.

- **E.D.**: Representa el tiempo invertido en la resolución del problema en cuestión usando el método de Exploración Dirigida (ED), para un ***Margen Permitido*** del 3 %.

- **FO**: es el valor que alcanza la Función Objetivo para cada problema (u.b.).

- **Error**: es el error cometido (%), medido sobre la solución óptima alcanzada por el XA, en la solución obtenida con la Exploración Dirigida.

Las tres tablas generadas son las siguientes, una por cada ratio de utilización estudiado:

Ratio para baja utilización (ρ=1.5)

Nº Tareas	Nº Sát	CT	CS	V	R	XA		E.D.	
						Tiempo	FO	Tiempo	Error
25	4	3	2	212	399	00:00:02	4388	00:00:01	0,00%
25	4	3	2	250	415	00:00:03	5978	00:00:01	0,00%
25	4	3	2	452	405	00:00:04	7427	00:00:05	0,00%
25	4	3	2	342	421	00:00:03	7027	00:00:02	0,00%
50	8	3	3	957	826	00:01:46	21043	00:03:24	0,12%
50	8	3	3	961	820	00:00:05	21651	00:00:57	0,67%
50	8	3	3	968	790	00:00:13	20674	00:00:41	1,01%
50	4	3	2	434	444	00:00:03	12391	00:00:01	0,00%
75	8	3	3	1347	845	00:03:18	33751	00:02:44	0,03%
75	8	3	3	1548	854	00:00:10	29522	00:01:00	0,00%
75	4	3	2	1160	469	00:03:43	22385	01:05:13	1,26%
75	8	3	3	1425	865	00:00:32	35217	00:00:38	0,01%
100	4	3	2	910	494	00:00:31	27705	00:07:01	0,01%
100	16	3	4	3380	1664	00:16:15	49698	00:01:16	0,00%
100	16	3	4	3208	1632	00:08:22	47600	00:01:26	0,00%
100	4	3	2	1752	492	00:22:54	32505	01:00:48	0,77%
200	4	3	2	1828	586	00:01:28	56192	00:15:09	0,25%
200	16	3	4	6512	1784	01:03:10	98459	00:05:31	0,00%
200	16	3	4	11220	1780	03:46:15	107541	00:24:46	0,00%
200	16	3	4	11716	1813	>12 h	-	00:21:33	-

Tabla 7-2. Tiempos de ejecución para ratio de utilización 1,5.

Podemos observar a la vista de los resultados que el método de Exploración Dirigida, que aunque mejora al XA en algunos casos, funciona, de forma general, peor que el XA.

Este empeoramiento general que presenta el método de Exploración Dirigida para pequeños valores del parámetro Número de Tareas es debido a la pérdida excesiva de tiempo que se produce cuando se llevan a cabo cada una de las llamadas a las librerías XA desde el algoritmo de exploración dirigida implementado. Este efecto se ha tenido en cuenta, de forma que la columna correspondiente a la Exploración Dirigida recoge los tiempos reales de resolución de cada problema descontando un tiempo por cada una de las llamadas al software XA, necesarias para la resolución de los nodos del árbol de exploración asociado a cada problema. EL tiempo descontado considerado, tomado éste por debajo del tiempo real perdido en cada llamada, ha sido de tres décimas de segundo por llamada al XA o nodo resuelto. Este descuento de tiempo podría evitarse sin más que mediante el empleo de un software de mayor potencia. Dicho de otro modo, sería necesario emplear un programa que para la resolución de un nodo en cuestión no requiera, como nos ha ocurrido con el software empleado en este proyecto, de la carga completa del modelo original asociado al problema a resolver (variables y restricciones). Un ejemplo del software al que nos referimos podría ser el caso de las librerías CPLEX. Además, es preciso decir que el tiempo de almacenaje de datos comentado alcanza grandes valores para problemas de gran tamaño (100 y 200 tareas). Aún así se ha tomado como criterio el descontar siempre el mismo tiempo por cada nodo resuelto.

Observando la tabla 7-2 podemos hacer otra consideración, que se refiere a que en 10 de las 19 soluciones de los problemas (un problema no se cuenta ya que el XA no es capaz de resolverlo) el método de Exploración Dirigida consigue hallar la solución óptima de los respectivos problemas, lo que equivale a una optimalidad del 52,63 %.

Ratio para media utilización (ρ=3)

Nº Tareas	Nº Sát	CT	CS	V	R	XA		E.D.	
						Tiempo	FO	Tiempo	Error
25	4	3	2	212	423	00:00:02	6211	00:00:01	0,00%
25	4	3	2	262	423	00:00:03	6551	00:00:01	0,00%
25	4	3	2	318	419	00:00:04	4807	00:00:09	0,08%
25	4	3	2	408	435	00:00:04	4233	00:00:03	0,00%
50	4	3	2	762	442	00:00:06	10075	00:00:30	0,00%
50	8	3	3	879	820	00:00:08	15523	00:00:13	0,00%
50	8	3	3	1554	822	00:00:10	14511	00:01:09	0,00%
50	8	3	3	1747	820	00:04:07	14441	00:00:59	0,61%
75	4	3	2	718	461	00:00:04	17376	00:00:24	0,67%
75	8	3	3	2482	832	01:08:06	22420	02:06:00	1,26%
75	16	3	4	2232	1643	00:01:54	33776	00:01:18	0,00%
75	16	3	4	4164	1662	03:00:00	31870	00:05:11	0,00%
100	4	3	2	958	486	00:00:05	20867	00:00:10	0,00%
100	8	3	3	3291	886	05:18:02	39554	02:46:12	0,27%
100	16	3	4	3408	1664	06:43:17	44842	03:59:12	0,16%
100	16	3	4	6280	1628	sobrecarga	-	00:27:14	-
200	4	3	2	1954	596	00:22:58	43097	01:45:53	0,28%
200	4	3	2	1916	594	00:19:31	44184	02:11:46	0,68%
200	8	3	3	6668	1719	sobrecarga	-	sobrecarga	-
200	16	3	4	6720	1723	sobrecarga	-	04:30:29	-

Tabla 7-3. Tiempos de ejecución para ratio de utilización 3.

En el caso de ratio de utilización 3 los resultados vienen a indicar que el método Exploración Dirigida mejora al XA principalmente en los casos de valores altos de los parámetros **Números de Tareas** y **Clases de Satélites**, salvo algunas excepciones, en que nuestro algoritmo de Exploración Dirigida tarda demasiado.

Comprobamos también que el ratio de utilización no influye en los tiempos de ejecución para el algoritmo de Exploración Dirigida implementado.

En lo que se refiere a la optimalidad obtenida en esta tabla de resultados, resulta del 52,94 % (9 soluciones óptimas de 17 comparadas), valor prácticamente idéntico al obtenido anteriormente.

Ratio para alta utilización (ρ=6)

Nº Tareas	Nº Sát	CT	CS	V	R	XA		E.D.	
						Tiempo	FO	Tiempo	Error
25	4	3	2	204	415	00:00:03	4076	00:00:01	0,00%
25	4	3	2	288	419	00:00:04	5126	00:00:02	0,00%
25	4	3	2	478	437	00:00:04	5162	00:00:03	0,00%
25	4	3	2	446	429	00:00:04	5867	00:00:02	0,00%
50	4	3	2	424	450	00:00:03	9312	00:00:13	1,07%
50	8	3	3	965	844	00:00:06	12854	00:00:04	0,00%
50	8	3	3	1854	879	00:00:11	11412	00:00:08	0,00%
50	8	3	3	1746	869	00:00:10	11880	00:00:07	0,00%
75	16	3	4	2372	1651	00:00:15	23221	00:00:15	0,00%
75	16	3	4	2692	1692	00:05:23	33237	00:01:47	0,00%
75	16	3	4	4380	1603	00:02:06	26571	00:01:36	0,00%
75	16	3	4	4388	1631	00:00:45	25256	00:00:38	0,00%
100	4	3	2	962	494	00:00:10	16231	00:00:34	0,00%
100	8	3	3	3064	891	00:19:31	29043	00:05:09	0,50%
100	16	3	4	6008	1668	07:28:38	38320	00:09:26	0,00%
100	16	3	4	5048	1676	06:32:11	35768	04:41:56	0,13%
200	4	3	2	1850	596	00:00:23	36521	00:05:53	0,13%
200	8	3	3	6217	1710	sobrecarga	-	sobrecarga	-
200	8	3	3	6654	1780	sobrecarga	-	sobrecarga	-
200	16	3	4	11492	1895	sobrecarga	-	sobrecarga	-

Tabla 7-4. Tiempos de ejecución para ratio de utilización 6.

Por último para un ratio de utilización 6 los resultados son claramente superiores, en lo que se refiere al menor tiempo de resolución de los problemas estudiados, para el algoritmo de Exploración Dirigida implementado. Puede comprobarse, sin más que observar la tabla anterior (Tabla 7-4), que en sólamente 3 ocasiones, de las 20 presentadas, el método XA ha sido superior al de Exploración Dirigida.

También hay que destacar, como se comentó en un comienzo, que las librerías XA no son capaces de resolver la mayoría de los problemas con 200 tareas y 4 Clases de Satélites en unos tiempos razonables, o a veces ni siquiera son capaces de resolver dichos problemas, En este aspecto el algoritmo de Exploración Dirigida mejora sensiblemente a las librerías XA, si bien es verdad que en algunas ocasiones el método de Exploración Dirigida tampoco es capaz de resolver algunos de los problemas de gran tamaño.

Volvemos a comprobar que el ratio de utilización no influye en los tiempos de ejecución para el algoritmo de Exploración Dirigida implementado.

En lo que se refiere a la optimalidad obtenida en esta tabla de resultados, resulta del 76,47 % (13 soluciones óptimas de 17 comparadas), valor muy superior al obtenido anteriormente para los demás ratios de utilización.

Cabe destacar que se resolvieron los mismos problemas para unos valores del **Margen Permitido** del 6 % y del 10 %, consiguiendo exactamente los mismos tiempos que los obtenidos para el valor de 3 % presentado anteriormente. Esto quiere decir que cuando el método presentado como alternativa al XA obtiene una solución que es entera dicha solución siempre (o al menos en los problemas resueltos) se encontrará bastante cercana al óptimo del problema, esto es, a una distancia menor del 3 % respecto a la cota superior del problema. Basta con comprobarlo observando los errores cometidos en cada una de las tres tablas anteriores.

Como hemos podido observar, a la vista de los resultados anteriores, el método XA es mejor que el método de Exploración Dirigida para aquéllos problemas que tengan un número pequeño de Clases de Satélites, pocas tareas a realizar y además un grado de solapamiento medio-bajo. Por contra observamos también que el método de Exploración Dirigida funciona mejor para problemas con gran número de Clases de Satélites y muchas tareas a realizar. Por tanto, podemos sacar en conclusión que un buen indicador podría ser el número de Clases de Satélites.

A la vista de lo anterior, la comparativa entre los dos métodos vamos a llevarla a cabo atendiendo primero al número de Clases de Satélites que intervengan, para cada cantidad de tareas implicadas. Para dicho estudio, no tomaremos en consideración el ratio de utilización, generando una nueva tabla (una por cada respectivo estudio) que será la media entre las tres tablas anteriores, agrupando entre sí los problemas por Clases de Satélites.

7.2.2. INFLUENCIA DEL NÚMERO DE CLASES DE SATÉLITES

La tabla correspondiente a la media de las 3 tablas anteriores, ordenada en función del número de Clases de Satélites y usando una única columna que represente al método de Exploración Dirigida sin tener en cuenta el parámetro MP (Margen Permitido), ya que vimos que su influencia resultaba despreciable, es la tabla siguiente:

Nº Tareas	CS = 2		CS = 3		CS= 4	
	XA	ED	XA	ED	XA	ED
25	3,30	2,80	-	-	-	-
50	4,00	14,60	46,22	51,33	-	-
75	113,50	1964,50	1073,50	1945,75	1903,83	107,50
100	355,00	1025,75	10126,50	5140,50	15224,60	6399,20
200	166,25	3880,25	-	-	20188,33	1036,66

Tabla 7-5. Tiempos medios de ejecución a partir de los tres ratios de utilización (segundos).

Además, en esta tabla no han sido incluidos aquellos problemas que no ha sido posible su resolución, ya sea por parte del método XA o bien por la Exploración Dirigida.

Para una mejor visualización de la comparativa entre los dos métodos, a partir de la tabla anterior vamos a generar tres gráficas una para cada número de Clases de Satélites:

- 2 Clases de Satélites.
- 3 Clases de Satélites.
- 4 Clases de Satélites.

En ellas, para un mismo Número de Tareas **NT** (eje X), obtenemos los tiempos de resolución empleados en los dos métodos (eje Y). En las gráficas se han representado en color azul los resultados obtenidos con las librerías XA y en color rojo los resultados obtenidos mediante la Exploración Dirigida. Las gráficas obtenidas para los casos comentados son las siguientes:

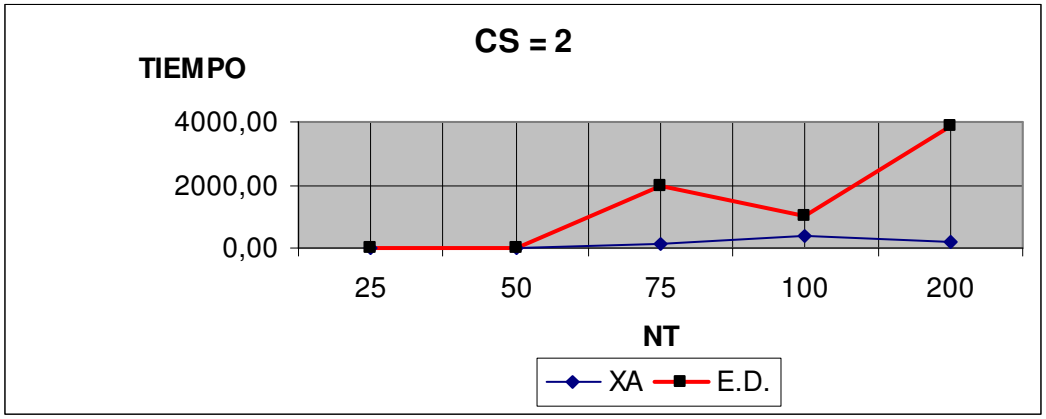


Figura 7-1. Representación de los tiempos de ejecución para 2 Clases de Satélites (seg).

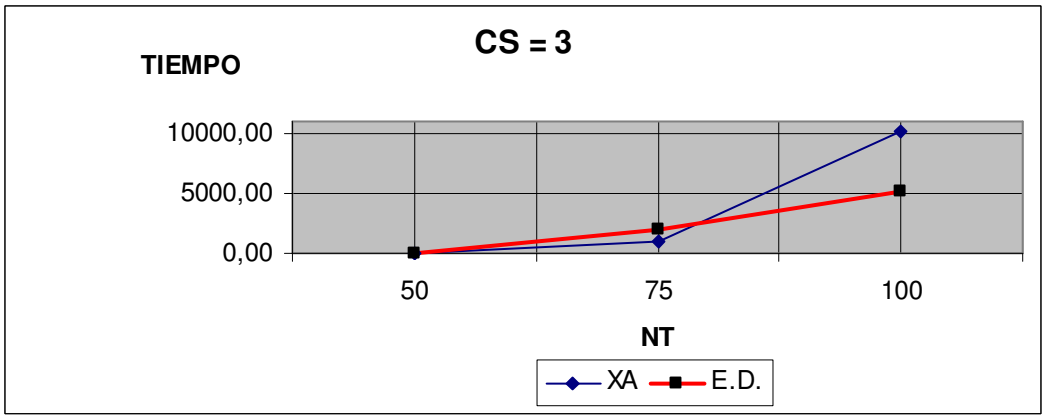


Figura 7-2. Representación de los tiempos de ejecución para 3 Clases de Satélites (seg).

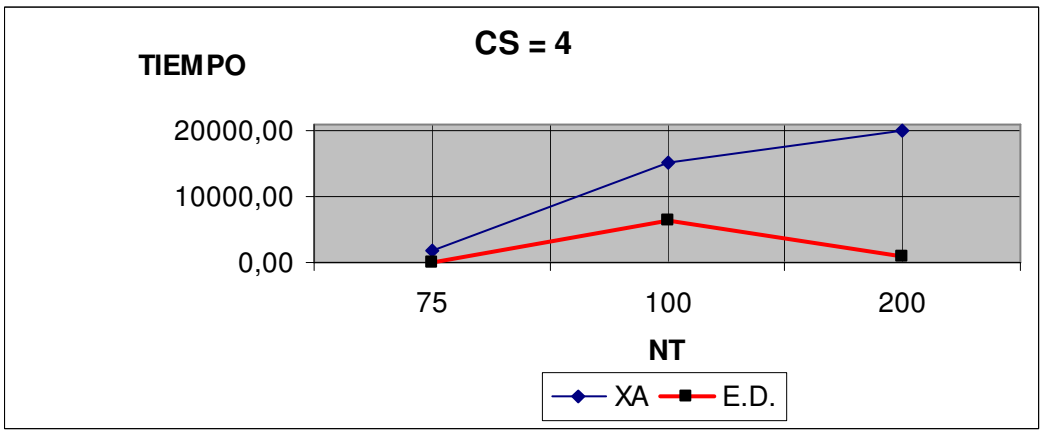


Figura 7-3. Representación de los tiempos de ejecución para 4 Clases de Satélites (seg).

Podemos observar en las figuras anteriores que el método XA mejora al en todos los casos al algoritmo de Exploración Dirigida para el caso de 2 Clases de Satélites. Así pues, y a la vista de los resultados, podemos concluir que para 2 Clases de Satélites el método de Exploración Dirigida es muy inferior tanto para problemas de baja densidad de tareas como para los de alta.

Para el caso de 3 Clases de Satélites, su representación deja claro que la Exploración Dirigida tiene un mejor comportamiento para tamaños superiores a 75 tareas y su curva presenta una evolución lineal, a diferencia de la del XA que presenta una concavidad.

A la vista de la representación gráfica, para 4 Clases de Satélites la Exploración Dirigida es siempre mejor al XA.

8 SUMARIO Y CONCLUSIONES

En el presente proyecto se ha realizado un estudio de un problema de planificación de tareas con restricciones de tiempo sobre un conjunto de satélites, optimizando la asignación de los mismos a las tareas a procesar para así maximizar el beneficio total obtenido.

La sistemática llevada a cabo ha sido el estudio y la resolución del problema **Operational Variable Interval Job Scheduling Problem** (OVISP) aplicado al caso de los satélites del EOS; para ello se ha realizado, en primer lugar, una clasificación de los problemas de planificación, ayudándonos para ello de ejemplos ilustrativos. En segundo lugar, se ha analizado el problema OVISP, considerado de clase NP, y se han tomado como técnicas para su resolución un método exacto (**librerías de optimización XA**) y un método aproximado (**Exploración Dirigida de carácter heurístico**), los cuales han sido explicados, y se ha realizado una implementación sobre el problema en concreto.

Posteriormente, en base a los resultados obtenidos al evaluar todos los problemas con ambos métodos, se han analizado los tiempos de resolución alcanzados atendiendo al grado de dificultad y al tamaño de los problemas, así como la optimalidad alcanzada por el método de Exploración Dirigida.

Las conclusiones más importantes tras el estudio experimental son las siguientes:

1) Por un lado destacar la capacidad del algoritmo de Exploración Dirigida en resolver la mayoría de los problemas de hasta 4 Clases de Satélites en tiempos razonables, aspecto éste del que escasea la resolución utilizando las librerías XA.

2) Por otra parte, otra conclusión clara es que para problemas que tengan pocas clases de máquinas (Clases de satélites), el método de resolución del software XA es sensiblemente superior al algoritmo de Exploración Dirigida descrito en este proyecto.

3) De otro lado, el algoritmo de Exploración Dirigida compite en tiempo con XA para problemas de tamaño medio (en lo que se refiere al número de clases de máquinas), ya que en esos casos es igual de factible el uso de cualquiera de los dos métodos. Dicho de otro modo, en estos casos ambos métodos son equiparables, y podrían usarse indistintamente uno u otro para la resolución de un problema.

4) Por contra, para problemas de gran tamaño (número de clases de máquinas) el método de Exploración Dirigida responde sensiblemente mejor que el XA, para cualquier cantidad de trabajos o tareas. Parece lógico pensar, pues, que si continuara aumentando el tamaño del problema, tanto en número de tareas a procesar como en número de clases (tanto de tareas como de satélites), únicamente el algoritmo de Exploración Dirigida podrá resolver el problema.

5) En lo que se refiere a la optimalidad, es necesario indicar que la resolución mediante librerías de optimización XA proporciona siempre soluciones óptimas para cada problema resuelto, si bien es verdad que los elevados tiempos de computación al incrementarse el número de tareas y clases de máquinas, hacen que dichas librerías XA sean tan sólo recomendables para problemas con reducido número de tareas y clases de satélites disponibles. Por el contrario nuestro método de Exploración Dirigida, aunque sus soluciones aportadas no sean forzosamente óptimas, invierte en la resolución unos tiempos aceptables, tanto en casos de gran número de tareas como de elevado ratio de utilización, encontrando la misma solución óptima entera o, en su defecto, una solución aproximada que está lo suficientemente cercana a la cota superior del problema continuo. Recordamos que el grado de cercanía a dicha cota superior es un parámetro (Margen Permitido) que, como se explicó en los capítulos 6 y 7, será una decisión que tomaremos nosotros.

Cabría volver a destacar la enorme complejidad del problema resuelto, considerado de Clase NP, así como también la dificultad de resolución mediante la Exploración Dirigida debida al software empleado. El uso de un software más potente reduciría en gran medida los tiempos de resolución de los problemas.

Para terminar, sería correcto decir, observando los resultados obtenidos en la resolución de la batería de los problemas, que existen ciertas configuraciones de los problemas que hacen que el método de Exploración Dirigida presentado en este proyecto resuelva dichos problemas en un tiempo sensiblemente superior al que emplean las librerías XA. Ello hace pensar en que el algoritmo de Exploración Dirigida expuesto pueda ser mejorado, a fin de obtener mejores resultados.

BIBLIOGRAFIA

- Arkin, E.M., Silverberg, E.L. (1987). Scheduling jobs with fixed starting and finishing times. Discrete Applied Mathematics, Vol. 18, pp. 1-8.

- Gabrel, V. (1995). Scheduling job within time windows on identical parallel machines: New model and algorithms. European Journal of Operational Research, Vol. 83, pp. 320-329.

- Potin, P. 1998. "End-To-End Planning Approach for Earth Observation Mission Exploitation", SpaceOps 1998, 1-5 June, Tokyo, Japan.

- Kolen, A.W.J., Kroon, L.G. (1991). On the computational complexity of (maximum) class scheduling. European Journal of Operational Research, Vol. 54, pp. 23-38.

- Kolen, A.W.J., Kroon, L.G. (1993). On the computational complexity of (Maximum) Shift Class Scheduling. European Journal of Operational Research, Vol. 64, pp. 138-151.

- Kroon, L.G., Salomon, M., Van Wassenhove L. N. (1995). Exact and approximation algorithms for the operational fixed interval scheduling problem. European Journal of Operational Research , Vol. 82, pp. 190-205.

- Wolfe, W.J., Sorensen, S.E. Three scheduling algorithms applied to Earth Observing systems domain. Management Science 2000, Vol. 46

- "A Comparison of Techniques for Scheduling Earth Observing Satellites", Al Globus, James Crawford (NASA Ames), Jason Lohn, Anna Pryor (NASA Ames)

- "Scheduling Earth Observing Fleets Using Evolutionary Algorithms", Al Globus, James Crawford, Jason Lohn and Robert Morris

- "Daily Imaging Scheduling of an Earth Observation Satellite", Wei-Cheng Lin, Da-Yin Liao, Member, IEEE, Chung-Yang Liu, and Yong-Yao Lee

- Bases de datos con Visual Basic 6 / Jeffrey P. McManus, Madrid [etc.] : Prentice Hall, 1999

- Microsoft Visual Basic 6.0 : manual del programador / Microsoft Corporation, Madrid [etc.] : McGraw-Hill, 1998

Sevilla, a 23 de Mayo de 2006