PROGRAMACIÓN POR RESTRICCIONES PARA SOLUCIONAR PROBLEMAS DE PLANIFICACIÓN

Daniel Díaz Araya, Francisco S. Ibáñez y Raymundo Q. Forradellas
Departamento e Instituto de Informática
Universidad Nacional de San Juan,
Cereceto y Meglioli (5400), San Juan, Argentina
e-mail: {ddiaz, fibanez, kike}@iinfo.unsj.edu.ar

RESUMEN

En los últimos años, las técnicas de resolución de problemas mediante el uso de restricciones, ha cobrado mucho interés dentro del área de la Inteligencia Artificial. En este trabajo se proponen nuevas técnicas para resolver problemas de scheduling, las cuales son susceptibles de ser aplicadas usando una herramienta basada en restricciones.

El problema consiste en producir una cierta cantidad de productos, para lo cual necesitan realizarse determinadas tareas, en un orden explícito. Para realizar estas tareas, existen ***n*** máquinas que pueden realizar algunas o todas las tareas, con distintos ritmos de producción (cantidad de productos realizados por unidad de tiempo). En este planteamiento se tiene en cuenta además el tiempo de preparación de las máquinas (setup). La entrada del problema la constituye el tamaño del lote de productos que se desea producir, la cantidad de tareas, la cantidad de máquinas, y las características de las mismas (ritmo de producción, tareas que realiza, tiempos de preparación, etc.). La salida la constituye un diagrama de Gantt que describe para cada máquina, los intervalos de tiempo en que se realizan las tareas, de modo tal que se obtenga el lote de productos deseado, optimizando el tiempo de producción.

Finalmente se muestran resultados que permiten evaluar las técnicas propuestas.

Palabras Clave: Scheduling - Inteligencia Artificial - Programación por restricciones - Programación Orientada a Objetos.

1. INTRODUCCION

La globalización ha producido cambios espectaculares en la participación de los mercados, así, surge como una agresiva competencia internacional basada en costos, calidad, plazos de entrega, innovación, flexibilidad y servicio al cliente. Para sobrevivir y tener éxito ahora y en el futuro, en las organizaciones se deben poner en práctica las nuevas tecnologías que están surgiendo.

Este trabajo presenta nuevas técnicas basadas en restricciones provenientes de la Inteligencia Artificial que ayudan a solucionar problemas en el proceso de planificación de la producción, explícitamente en el área de asignación de recursos y scheduling de tareas. Para demostrar la aplicabilidad y viabilidad de tales técnicas se desarrolla un programa para resolver un problema de scheduling complejo.

1.1 Herramientas para Programar con Restricciones

Existen fundamentalmente dos paradigmas para abordar problemas con restricciones. Por un lado uno basado en la programación lógica [Jaffa, 87] y otro basado en programación orientada a objetos[ILOG,Sol-U].

Los paradigmas basados en restricciones permiten trabajar con distintos dominios asociados a las variables. En este trabajo utilizaremos los dominios finitos [Hente,89],[Ibañez,94] ,[Forrade,94], [Rueda,95].

En esta propuesta utilizaremos el paradigma orientado a objetos. Las herramientas concretas en que se desarrollaron las técnicas presentadas en este trabajo son las siguientes: ILOG SOLVER [ILOG,Sol-U], [ILOG,Sol-R]. ILOG SCHEDULE. ILOG,Sch-U],[ILOG,Sch-R],

ILOG SOLVER es una biblioteca de C++ para solucionar problemas en muchos dominios. En dominios tan diferentes como Asignación de Recursos, planificación de actividades, organización del personal, corte de materiales, optimización de mezclas, etc. **ILOG SCHEDULE,** esta diseñada como una extensión de la biblioteca SOLVER y provee clases derivadas de las clases de SOLVER que expresan restricciones típicas en problemas de asignación de recursos y scheduling.

La separación de la representación del problema, de los mecanismos utilizados para resolverlo, es fundamental en la programación por restricciones y estas herramientas se basan en esta separación para abordar un problema. En SOLVER Y SCHEDULE, la parte desconocida del problema se representa con ***variables restringidas.*** Se define como variable restringida a una variable que sólo puede tomar valores desde un conjunto de valores denominado dominio**.** De este modo, la representación del problema consiste en la declaración de las variables y la colocación de las restricciones sobre ellas. La resolución de un problema consiste en encontrar un valor para cada variable de modo que se satisfagan todas las restricciones. Debido a que es posible que exista más de una solución para un problema dado, la elección de una solución determinada, normalmente se determina de acuerdo a un criterio de optimización. Para capturar la representación del problema, estas herramientas utilizan la programación orientada a objetos que provee C++, mediante el uso de clases.

1.2 La Aplicación

La aplicación que se ha desarrollado trata con un proceso de producción que consta de 4 tareas, T0, T1, T2, T3 para elaborar un producto. Estas tareas cambian el estado del producto. En una primera instancia existe sólo la materia prima, la misma es procesada por la Tarea T0, cuyo resultado es el producto en estado E0. Este producto comúnmente se lo denomina producto semielaborado. Luego al aplicar la tarea T1 pasa al estado E1, y así sucesivamente hasta que el producto alcanza el estado E3, donde se obtiene el producto terminado. Para realizar estas tareas existen *n* máquinas que pueden realizar algunas o todas las tareas, ellas pueden realizar las tareas a distintos ritmos de producción. Se define como ritmo de producción

de una máquina, para una tarea determinada, la cantidad de productos para esa tarea que puede realizar por unidad de tiempo. Si una máquina realiza más de una tarea, la máquina necesita una preparación para que pueda realizar determinada tarea. Esta preparación insume cierto tiempo, el cual es denominado tiempo de preparación (setup).

El Objetivo del problema es minimizar el tiempo de manufactura para cada tarea, de modo de obtener una optimización del tiempo de manufactura del producto terminado.

La entrada para el problema la constituye el tamaño del lote de productos que se necesitan producir, y las características de las tareas y de las máquinas (ritmo de producción, tiempo de setup, etc).

La salida la constituye una secuencia de operaciones, en donde una **operación** es el trabajo que lleva a cabo determinada **máquina** realizando una determinada **tarea**, en la cual se indican, además de la máquina y la tarea, el tiempo o duración de la misma, el punto temporal inicial y final, y el tipo y cantidad de insumo que la misma requiere. El tipo de insumo requerido está determinado por la tarea a realizarse. Por ejemplo, para realizar la tarea T1 el tipo de insumo serán productos semielaborados en estado E0, etc. La salida deberá determinar en qué puntos temporales y con qué insumos necesita cargarse determinada máquina para que realice determinada tarea.

1.3 Técnicas desarrolladas para abordar el problema

La complejidad principal del problema planteado, es que, la cantidad de operaciones (que utilizan determinados recursos) no se conoce a priori, y es la propia dinámica de la resolución quien las determina.

Para poder abordar este problema, se proponen dos modelos, basados en objetos, uno para representar la estructura estática del problema y otro para describir la estructura dinámica. Que permitirá comprender, entre otras cosas, cómo se generan dinámicamente las operaciones en el proceso de encontrar una solución, que sea óptima según un determinado criterio de optimización y que verifique todas las restricciones. El modelo para representar la estructura dinámica incluye además nuevas técnicas para determinar los tiempos de comienzo y fin de las operaciones, en función de los tiempos de setup y de los ritmos de producción de las distintas máquinas.

2. Descripción completa del problema

La aplicación que se ha desarrollado trata con un proceso de producción que consta 4 tareas, A0, A1, A2, A3 para realizar un producto. Estas tareas confieren al producto cuatro estados distintos. En una primera instancia existe sólo la materia prima, la misma es procesada por la Tarea A0, cuyo resultado es el producto en estado E0, este producto comunmente se lo denomina producto semielaborado. Luego al aplicar la tarea A1 pasa al estado E1, y así sucesivamente hasta que el producto alcanza el estado E3, donde se obtiene el producto terminado. Para realizar estas tareas existen *n* máquinas que pueden realizar algunas o todas las tareas, ellas pueden realizar las tareas a distintos ritmos de producción. Si una máquina realiza más de una tarea, la máquina necesita una preparación para que pueda realizar determinada tarea, esta preparación insume cierto tiempo, el cual es denominado tiempo de preparación. La entrada para el problema la constituye el tamaño del lote de productos que se necesitan producir. La salida la constituye una secuencia de operaciones en la cual se indica, la máquina que realiza tal operación, el tiempo o duración de la misma, el punto temporal de comienzo y final de la operación y el tipo y cantidad de insumo que la misma requiere. Esta salida determina en qué puntos temporales y con qué insumo necesita cargarse determinada máquina para que realice determinada tarea.

2.1.1 Objetivo

Minimizar el tiempo de manufactura para cada tarea, para de esta manera intentar obtener una optimización del tiempo de manufactura del producto terminado.

2.1.2 Descripción

Existen n máquinas con distintas capacidades. Las tareas que realizan estas máquinas son:

Tarea A0.
Tarea A1.
Tarea A2.
Tarea A3.

2.1.3 Consideraciones Generales

Se deben tener en cuenta que no todas las máquinas pueden realizar todas estas tareas.

- ✓ El orden de ejecución de las tareas para obtener un producto es A0 - A1 - A2 - A3.
- ✓ Existen máquinas que realizan las mismas tareas pero con distinto ritmo de producción.
- ✓ Una máquina que es capaz de realizar por ejemplo las tareas A1, A2 y A3 puede por necesidades de producción, ser programada para realizar por ejemplo solamente la tarea A1.
- ✓ Las tareas para producir un producto se deben hacer en el orden descripto anteriormente, es decir no se puede realizar la tarea A1 sin haber realizado antes la tarea A0.
- ✓ Existe un tiempo de preparación para cada máquina, el cual consiste en el tiempo necesario para preparar la máquina para una nueva tarea.
- ✓ Una máquina se puede preparar para ejecutar sólo una tarea en un mismo instante de tiempo.
- ✓ El proceso de producción para producir un producto es secuencial, pero las tareas que se aplican sobre las distintos estados del producto no los son, o lo que es lo mismo pueden existir concurrencia en las tareas. Mientras una máquina esta realizando la tarea A0, en ese mismo instante otra puede estar realizando la tarea A1.
- ✓ Los requerimientos de cada tarea son las máquinas y los insumos que ellas necesitan, estos son para la tarea T0, materia prima, para la tarea T1, producto en estado E0, para la tarea T2, producto en estado E1, para la tarea T3, producto en estado E2.

2.1.4 Consideraciones sobre la complejidad

Existe una explosión combinatoria sobre el número de posibles planes dependiendo de las características de las máquinas disponibles. Esto hace necesario modelar el problema de manera tal que el mismo pueda armar un plan de acuerdo con las máquinas disponibles en ambiente.

Para un mismo plan puede existir una cantidad grande de soluciones, la cual es generada por una explosión combinatoria de los distintos caminos para alcanzar una solución. Seleccionar una solución será función de la heurística elegida para tal fin.

Consideraciones especiales merecen la disponibilidad de los insumos en todo instante de tiempo, al igual que la disponibilidad de las máquinas, ellos son determinantes en la planificación de cualquier tarea, como así también la cantidad de productos producidos por cada una de las tareas en un determinado instante de tiempo. Otro factor que agrega complejidad al problema son los tiempos de preparación de las máquinas, los cuales determinarán o no que una máquina realice o no determinada tarea.

El solucionador del problema actuará creando operaciones, que requieren insumos y máquinas, y luego deberá encontrar el tiempo de duración para cada operación que minimice la duración de la tarea. Estas duraciones generaran una gran explosión combinatoria que producirá asignaciones en el tiempo de las máquinas a las distintas tareas. Así en determinado punto del tiempo deberán existir máquinas que estén asignadas a tareas y otras que no.

2.1.5 Entrada

La entrada para el problema queda determinada por un conjunto M ={M1, M2, ..., M_n}de máquinas con distintas características, y el tamaño del lote a producir: Tlote.

Una máquina queda caracterizada por los siguiente conjunto de atributos:

Atributos={ Tareas que Realiza,
Ritmo de producción para cada tarea que realiza,
Tiempo de preparación,
Tarea actual para la cual esta preparada actualmente,
Nombre}

2.1.6 Salida

La salida será un diagrama de Gantt, que mostrará las distintas operaciones de una máquina en el espacio temporal. De esta manera, en el eje vertical estarán ubicadas las distintas máquinas, y en el eje horizontal el tiempo.

3. Modelo para el Problema

La siguiente figura (1) muestra un diagrama de categoría de clases para la solución del problema.

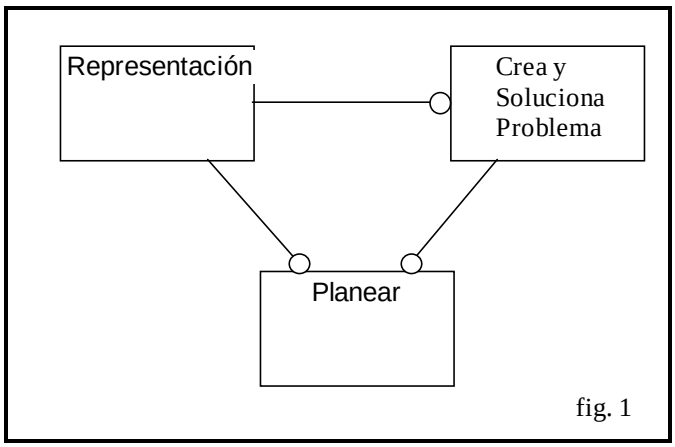

fig. 1

La categoría de clases ***Planear*** agrupa todas las clases que tienen que ver con la interface del usuario, la base de datos, conversiones, tratamiento de error etc.

La categoría de clases ***Representación*** agrupa todas las clases que tienen que ver con la representación de un problema dado. Debemos recalcar que un problema de este tipo queda bien definido sólo cuando se conoce la cantidad precisa de instancias de estas clases y esto se puede establecer sólo en tiempo de ejecución pues es el usuario quien completará la definición del problema a resolver haciendo uso de la interface visual. Por lo anterior podemos decir que las clases de ***Representación*** definen una cantidad suficientemente grande de problemas y es el usuario quien decide cual de ellos resolver.

La categoría de clases ***Crea y Soluciona Problema*** contiene las clases de objetos con comportamientos que permiten completar la definición del problema con los datos que ingresa el usuario mediante la interface visual. Esto se realiza creando las correspondientes instancias

de las clases pertenecientes a Representación. También posee clases de objetos que permiten encontrar una solución para el problema.

3.1 La Categoría de Clases, Representación

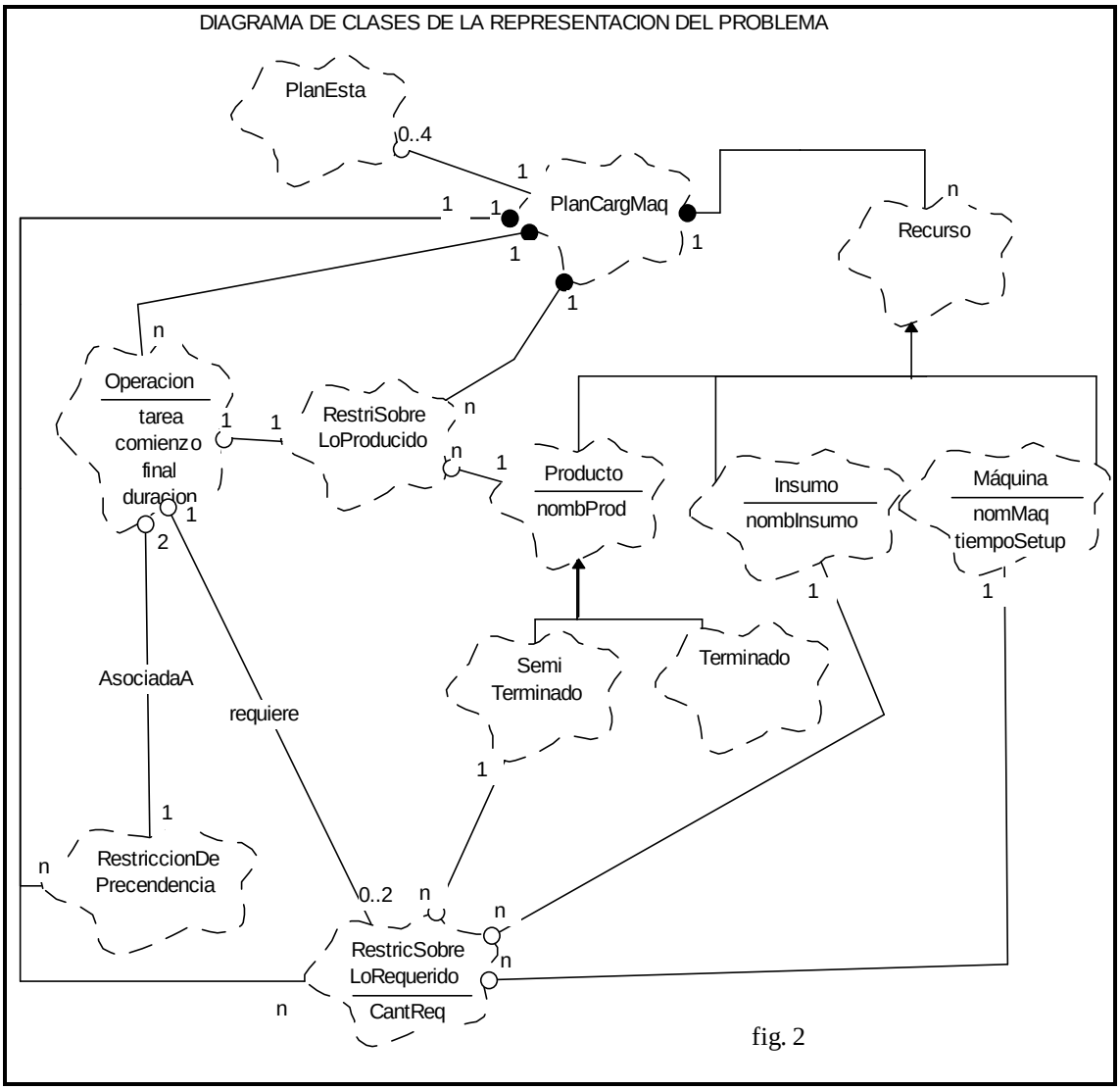

fig. 2

La figura 2 muestra la categoría de clases ***Representación***, la notación utilizada es la de [Booch, 91]. La clase PlanCargMaq es la depositaria de todo el plan de carga para las máquinas. La misma esta compuesta por recursos, operaciones , restricciones sobre los que se debe producir, restricciones de precedencia entre operaciones, restricciones sobre lo requerido, un recurso puede ser una Máquina, un Insumo ó un Producto, este ultimo puede ser un producto semiterminado o terminado. Una operación tiene cuatro atributos principales, la tarea que realiza (T0, T1, T2, T3), comienzo indica el punto temporal donde comienza la operación, final indica un punto temporal donde termina la operación, duración indica la duración de la misma. Una operación necesita ciertos recursos para realizar una tarea. Por ejemplo, para obtener n productos semielaborados en estado E0, se necesita una maquina que realice la tarea T0 (recurso del tipo máquina). Si se desean obtener n productos semielaborados en estado E1, se necesita una máquina que realice la tarea T1 y n productos en estado E0. Estas necesidades se modelan por medio de restricciones sobre lo requerido. La cantidad de productos que se necesita producir, como la cantidad que produce una máquina por ciclo se modelan por medio de restricciones sobre lo producido. Estos tipos de restricciones ligarán los recursos a la

operación durante todo el intervalo en que transcurra la operación. Las restricciones de precedencia indican que una operación debe realizarse antes o después de otra.

A manera de clarificar la descripción anterior mostramos a continuación un diagrama de Gantt que es la salida de un problema

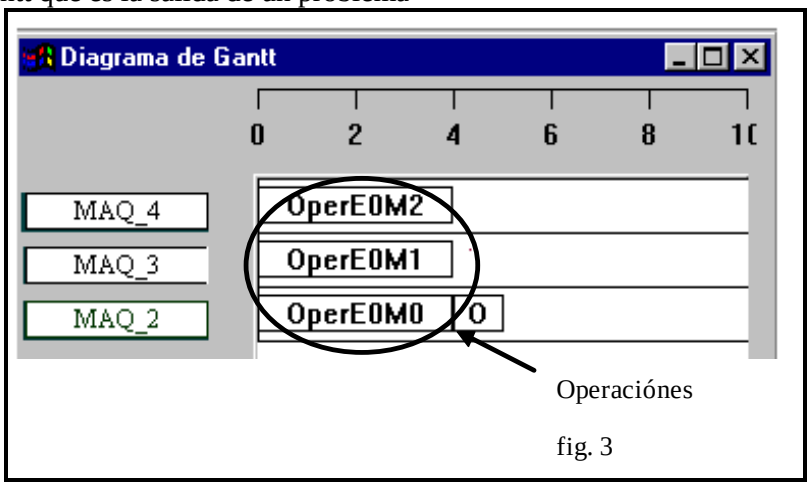

fig. 3

En el gráfico de Gantt se puede observar una planificación para obtener 37 productos en el estado E0, todas las máquinas pueden realizar la tarea T0. Los ritmos de producción de MAQ_4, MAQ_3 y MAQ_2 son, respectivamente 4,3 y 2. La operación OperE0M2 que es realizada por la máquina MAQ_4, tiene una duración de 4 ciclos o unidades de tiempo, comienza en el ciclo 0, termina en el ciclo 4. Como MAQ_4 tiene un ritmo de producción de 4 unidades por ciclo, la operación debe producir 4 unidades por ciclos. Esta es una restricción del tipo restricción sobre lo que se necesita producir. La operación antes de comenzar necesita materia prima, pero, a los efectos de simplificar el modelo, la consideramos infinita y consecuentemente no es necesario considerarla. También necesita la disponibilidad de la máquina MAQ_4 en el intervalo 0-4, estas son restricciones sobre lo requerido.

El plan de carga para las máquinas lo componen las 4 operaciones que son ejecutadas por las tres máquinas. La cantidad de producto que se obtiene es la suma de lo producido por cada operación. Si OperE0M2 produce un total de 16 productos y OperE0M1 produce 12 y las restantes operaciones realizadas por la máquina MAQ_2, producen 8 y 1, tenemos que se producen 37 unidades de productos.

3.2 La Categoría de Clases Crea y Soluciona Problemas

Como se mencionó anteriormente la categoría de clases ***Representación*** nos aporta sólo el esqueleto del plan de carga, pero ellas no determinan, qué cantidad de productos producir, qué cantidad de máquinas existe, y por lo tanto tampoco se sabe que cuántas operaciones tendrá el plan, ni las restricciones asociadas que posee cada operación. En otras palabras, la representación no define completamente el plan sino que sólo provee un "**template**" (esqueleto) para el plan que hay que cargar con datos que ingresa el usuario. Cuando este template esté cargado se debe proceder a buscar una solución. Estas dos tareas son llevadas a cabo por los objetos cuyas clases se definen en ***Crea y Soluciona Problema***. Cabe mencionar que esta estr1uctura de clases se utiliza para solucionar problemas que no presentan tareas concurrentes. Si se desean 37 productos terminados, primero se realiza la tarea T0 para obtener 37 productos en estado E0, luego se realiza la tarea T1 para obtener 37 productos en estado E1, y así sucesivamente hasta obtener 37 productos terminados (en estado E3). En la figura 2 se puede observar también que cada clase deriva de IlcGoal. Esta es una clase que nos permite hacer búsquedas y utilizar el backtraking para deshacer acciones.

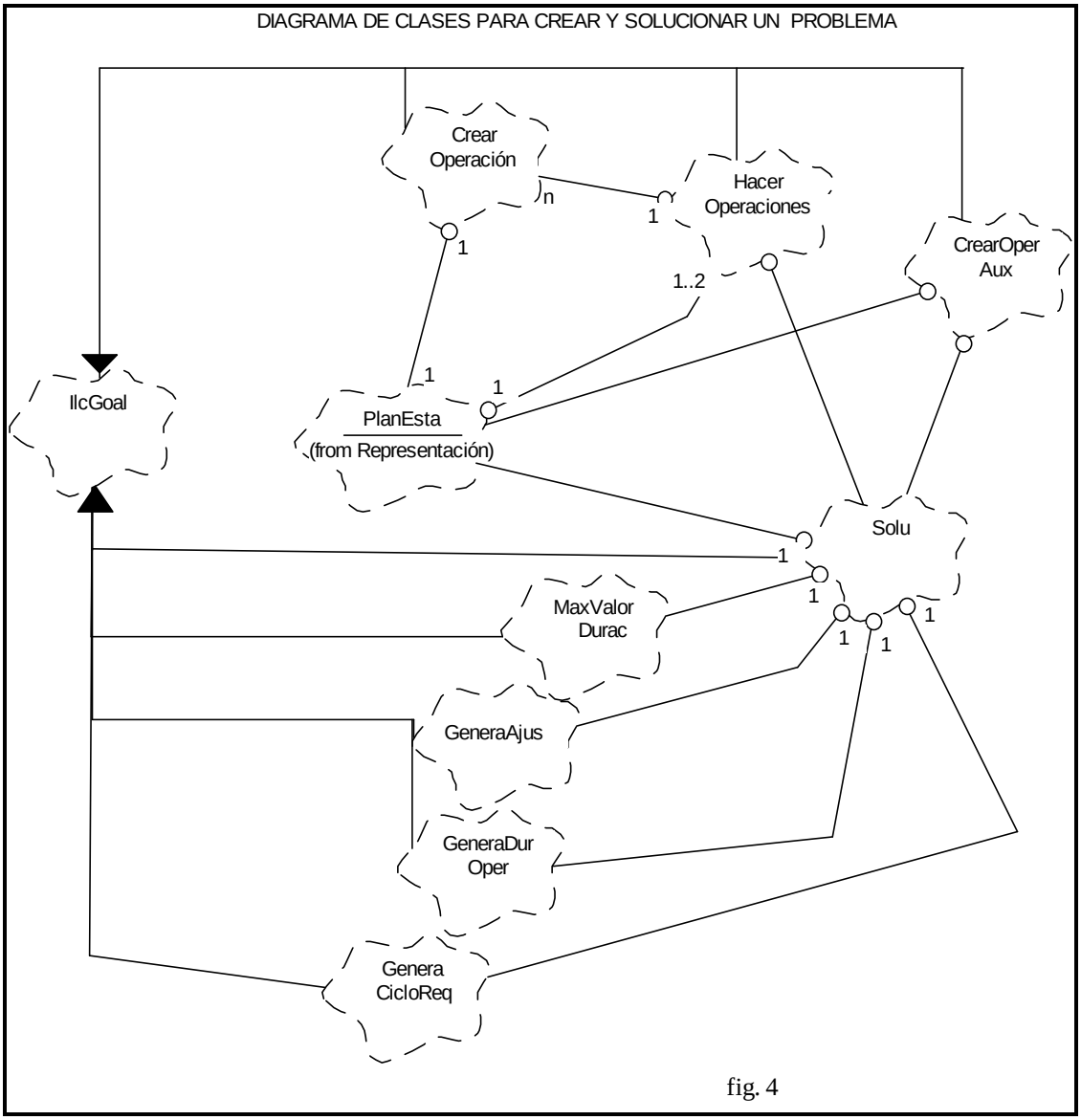

fig. 4

3.2.1 Descripción resumida de las clases más importantes.

PlanEsta	**Hereda:** No tiene padre **usa:** HacerOperaciones, PlanCargMaq, Máquina[1],Solu **Objetivo:** Obtiene Plan a realizar, selecciona las máquinas, conserva algunas variables globales necesarias en la implementación, llama a la goal Hacer Actividades para crear operaciones y luego busca una solución que cumpla con el objetivo
HacerOperaciones	**Hereda:** IlcGoal **usa:** CreaOperación, PlanEsta **Objetivo:** Crear todas las operación de PlanCargMaq para la tarea.

[1] La clase máquina no se ha referenciado en el diagrama de clase Crea y Soluciona Problema, por motivos de simplicidad pero es evidente que se la necesita.

CrearOperaciones	**Hereda:** IlcGoal **usa:** PlanEsta, operación **Objetivo:** Crear la operación de PlanCargMaq y coloca las restricciones sobre la operación, actualiza los atributos de PlanEsta
Solu	**Hereda:** IlcGoal **usa:** PlanEsta, MaxValorFinTarea, GeneraAjus, GeneraDurOper, GeneraCicloReq, CreaOper **Objetivo:** Busca una solución que permita producir los n productos de una determinada tarea en el menor tiempo posible
MaxValorFinTarea	**Hereda:** IlcGoal **usa:** variable dominio que representa a finTarea **Objetivo:** Instancia la var. dominio con su límite superior.
GeneraAjus	**Hereda:** IlcGoal **usa:** arreglo de variables dominio que representa a los ajustes **Objetivo:** Instancia cada var. dominio del arreglo con su límite inferior.
GeneraDurOper	**Hereda:** IlcGoal **usa:** arreglo de variables dominio que representa a las finTareaiones **Objetivo:** Instancia cada var. dominio del arreglo con su límite inferior.
GeneraCicloReq	**Hereda:** IlcGoal **usa:** arreglo de variables dominio que representa a los ciclos de finalización de cada operación **Objetivo:** Instancia cada var. dominio del arreglo con su límite inferior.

4. Implementación

Para la etapa de implementación se han desarrollado dos prototipos. El primero, tuvo como objetivo, probar la viabilidad de la implementación del modelo con las herramientas mencionadas. Este prototipo esta preparado para resolver solamente problemas pequeños.

El segundo prototipo, el cual se encuentra en desarrollo, tiene como objetivo, resolver problemas con datos de entrada de gran tamaño. El mismo deberá resolver problemas cuyo tamaño de lote (de productos a elaborar), este en el orden de los 10000 productos y que los mismos se puedan planificar hasta con 20 máquinas.

4.1 Resultados

Usando el **primer prototipo** y los siguientes datos de entrada

Para un lote de 100 productos y cuatro tareas, usando 3 máquinas MAQ_2, MAQ_3, MAQ_4, que realizan todas las tareas con los siguientes ritmos de producción. MAQ_2 ,realiza las tareas a un ritmo de producción de 2 productos por unidad de tiempo (RP_MAQ_2 = 2). MAQ_3 lo hace a 3 productos por unidad de tiempo y MAQ_4 lo hace a 4 productos por unidad de tiempo. El tiempo de preparación es igual a 2 , según puede verse ne la Figura 5.

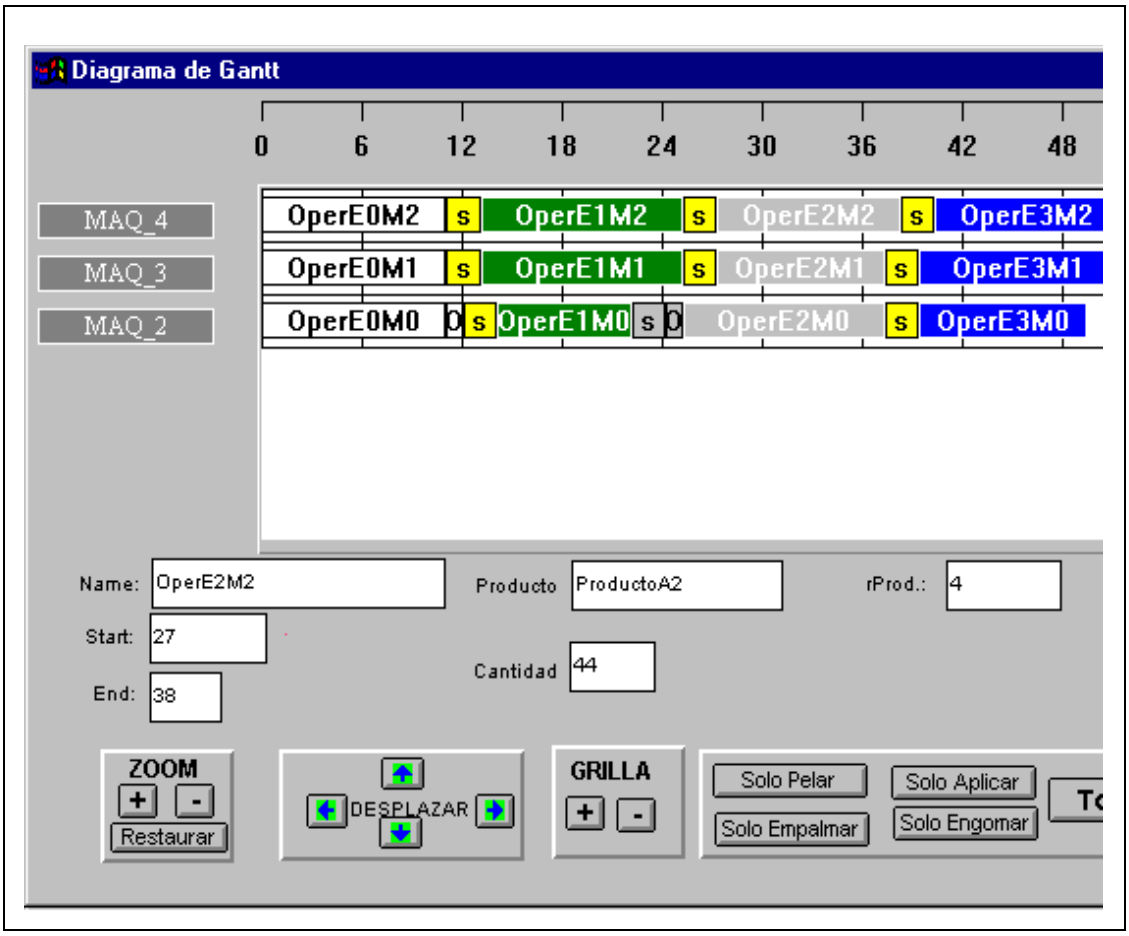

Figura 5: Diagrama de Gantt para el Ejemplo

La estadistica producida por la ejecución del programa es la siguiente.

Number of fails	: 406
Number of choice points	: 1147
Number of variables	: 417
Number of constraints	: 141
Reversible stack (bytes)	: 8060
Solver heap (bytes)	: 148760
And stack (bytes)	: 4040
Or stack (bytes)	: 4040
Constraint queue (bytes)	: 4068
Total memory used (bytes)	: 168968
Total CPU time since IlcInit	: 7.454

En la estadística se observa un tiempo de ejecución de 7 segundos y 406 fallas, y 141 restricciones.

Con el segundo prototipo

Para un lote de 10000 productos y cuatro tareas, usando 8 máquinas que realizan todas las tareas pero a distintos ritmos de producción, y tiempo de setup igual a cero. Se obtuvo un tiempo de ejecución de 104 segundos y producieron 3680 fallas, y 224 restricciones.

Number of fails	: 3680
Number of choice points	: 38169
Number of variables	: 789
Number of constraints	: 224

Reversible stack (bytes) : 32180
Solver heap (bytes) : 289460
And stack (bytes) : 4040
Or stack (bytes) : 4040
Constraint queue (bytes) : 4068
Total memory used (bytes) : 333788
Total CPU time since IlcInit : 103.998

5. Conclusiones

La aplicación a problemas de scheduling de técnicas provenientes de la Inteligencia Artificial, como la Programación por Restricciones, es una alternativa viable, y además eficiente, para resolver problemas específicos.

La problemática que aborda este proyecto es similar a la mayoría de los problemas industriales en los que es necesario optimizar la planificación de tareas para alcanzar los mejores niveles de productividad. El tiempo de desarrollo, no solo se reduce en la primera fase del diseño. sino también en sucesivas modificaciones. Además el tratamiento de problemas en forma dinámica permite una planificación sensible a los cambios o fallas que puedan producirse en un proceso productivo.

6. Referencias

[**Booch, 91**] Grady Booch, Object-Orient Design with Aplications, Benjamin/Cummings, Redwood City, Calif., 1991.

[**Forrade,94**] R.Forradellas, "Un Modelo para la Resolución de Sistemas Dinámicos con Restricciones en el marco CLP", Tesis Doctoral, Universidad Politécnica de Madrid, España, 1994

[**Hente,89**] P. Van Hentenryck, "Constraint Satisfaction in Logic Programming" MIT Press, 1989.

[**Ibañez,94**] F. Ibañez, "CLP(Temp), Integración de Restricciones Temporales Métricas y Simbólicas, en el Marco CLP", Tesis Doctoral, Universidad Politécnica de Valencia, España, 1994.

[ILOG,Sch-R] "ILOG SCHEDULE- Reference Manual Version 2.0", Ilog, France, 1995.

[ILOG,Sch-U] "ILOG SCHEDULE - User Manual Version 2.0", Ilog, France, 1995.

[ILOG,Sol-R] "ILOG SOLVER - Reference Manual Version 3.0", Ilog, France, 1995.

[ILOG,Sol-U] "ILOG SOLVER - User Manual Versión 3.0", Ilog, France, 1995.

[Jaffa, 87] J. Jaffar and J-L. Lassez. Constraint Logic Programming, ACM, 1987.

[**Rueda,95**] Rueda L. Klenzi R. Gutierrez L. Ibañez F. Forradellas R., "Tratamiento de Problemas de Combinatoria Discreta mediante el Paradigma CLP", 2das. Jornadas de Inteligencia Artificial, Universidad Nacional del Sur, Bahía Blanca, 1995.