INSTITUTO UNIVERSITARIO AERONÁUTICO

Facultad Ciencias de la Administración

Tesis de grado Ingeniería de Sistemas

Optimizar

Solución de problemas de planificación con un algoritmo determinista

Autora: Patricia Soledad Simonelli

Tutora: Brenda Meloni

Córdoba, 2013

Declaración de derechos de autor

A la memoria de René Simonelli,
por haberme transmitido
sus valores morales,
la perseverancia y
la constante alegría
en el alcance
de toda meta.

Agradecimientos

Decía Cortázar que las palabras nunca alcanzan cuando lo que hay que decir desborda el alma, pero haré el intento de volcar mi inmensa gratitud a todos los que me ayudaron a alcanzar este logro personal que sin dudas es compartido, porque no hubiera sido posible sin el apoyo constate, presente y a la distancia de mucha gente.

En primer lugar a mis padres Nanci y Héctor que fueron las bases sólidas en las que apoyarme cuando todo parecía ser complejo y lejano, y a mi hermana Natalia que con su cariño y su carácter me dió los empujones necesarios para no sucumbir cuando la carga parecía muy pesada.

A Leonardo, por su infinita paciencia cuando estaba un poco ausente, o cuando las situaciones se complicaban, y por alentarme siempre a conseguir lo que me hace feliz.

A mis abuelos, mis tíos y mis primos que me contuvieron, apoyaron y se alegraron de cada logro conseguido, al igual que mis amigos, que con su cariño, su confianza y su alegría, fueron un soporte inconmensurable.

A mis compañeros del IUA, que aún en la distancia, estuvieron siempre presentes apoyándome y alentándome a continuar, porque a pesar de que esta modalidad de estudio no me permitió tener un "curso", pude sentir el apoyo y la solidaridad de compañeros que se encontraban en los lugares más remotos.

A mi profesora tutora Brenda Meloni por su profesionalidad y calidez, por allanarme este complejo camino de la elaboración de la tesis, preocupándose por mi avance y alentándome a continuar, también a la profesora Natalia Mira por su atención y apoyo, porque colaboró mucho en mis últimos años de carrera cuando la finalización del ciclo parecía compleja y lejana.

Y por último a todo el Instituto Universitario Aeronáutico porque cuando ingresé algunas personas dudaban de la modalidad de estudio elegida, pero puedo dar fe que en esta casa de estudios el compromiso de todos los involucrados es real y el éxito depende de la fuerza de voluntad que uno ponga en el alcance del objetivo, porque desde la institución las garantías están dadas para conseguir lo que uno se proponga.

Abstract:

La optimización industrial es un tema ampliamente abordado por diversas investigaciones con incumbencias interdisciplinarias. A lo largo de estos últimos 50 años pueden observarse grandes avances en la planificación y optimización del uso de recursos, esto se ha dado en gran parte por la utilización de herramientas computacionales, en la actualidad, se encuentran en estudio la aplicación de métodos relacionados a la lógica computacional compleja para obtener mejores resultados. En la presente investigación se implementará un nuevo método algorítmico para obtener otra alternativa a la solución de los problemas de Job Shop Scheduling.

El algoritmo es el planteado para la resolución de los rompecabezas Sudoku, este tipo de juego es uno de los 21 problemas de Karp NP-completo (problemas computacionales famosos que tratan sobre combinatoria y teoría de grafos), al igual que los problemas de planificación, debido a que se los considera como problemas de satisfacción de restricciones k-SAT puede plantearse una analogía entre ambos.

Palabras claves:

Planificación, optimización, algoritmo determinista, lógica computacional

The industrial optimization is a topic widely discussed by interdisciplinary researches, over the last 50 years can be seen great advances in planning and optimizing the use of resources, but there are still some drawbacks related to the complex computational logic. The present research is to implement a new algorithmic method to give an alternative to solving the problems of job shop Scheduling.

This algorithm is proposed for solving Sudoku puzzles, this type of game is one of Karp's 21 NP-problems completely (they are famous computational problems dealing with combinatory and graph theory), as well as planning issues, because they are considered as constraint satisfaction problems k-SAT can consider an analogy between the two.

Keywords:

Planning, optimization, deterministic algorithm, computational logic

Índice

Índice de ilustraciones

Índice de tablas

Prólogo

"The whole of science is nothing more than a refinement of everyday thinking."

- Albert Einstein

La tecnología, al servicio del hombre, dá respuesta a diferentes circunstancias que se presentan día a día o extraordinariamente. Ante este planteo surge un interrogante ¿se ha alcanzado el límite en los avances tecnológicos que tienen repercusión en lo cotidiano? Sin dudas no es así, mancomunadamente ingenieros e investigadores ponen sus conocimientos a disposición y los integran de modo tal que surgen nuevas inquietudes y respuestas, que si bien pueden ser aplicados en ámbitos complejos, también impactan en el quehacer diario.

Basado en los interrogantes comienza la inquietud de investigar si existen mejoras aplicables a la programación de tareas, tema de gran interés para las materias que lo estudian y que se considera puede continuar evolucionando, de modo tal que genere grandes ventajas para las empresas que optimicen su producción haciendo uso de la lógica computacional, la matemática y la estadística.

Las maneras distintas de generar valor productivo, y ofrecer mercadería altamente planificada generan diferenciaciones con la competencia y posibilitan que los márgenes con los que se lleva adelante el proceso de producción sean superiores, permitiendo a la empresa ser más flexible con los constantes cambios que los nuevos modelos económicos demandan.

Dando los primeros pasos de la investigación surge como objetivo analizar las reglas y algoritmos que pueden optimizar el proceso de planificación y programación de la producción, ordenar las tareas que desarrollan empresas reales y que generan distintas alternativas productivas. Considerando esta información será posible aplicar la simulación con datos reales, de modo que puedan obtenerse reducciones y ordenamientos para maximizar los beneficios.

La planificación puede entenderse como la satisfacción de restricciones (más o menos variables), y como tal puede ser planteada como problemas factibles de ser resueltos por algoritmos (obteniendo los mejores resultados con la realización de pasos sucesivos).

A partir de la investigación de esquemas, procesos, modelos, algoritmos y fórmulas, existe un trabajo reciente publicado (2012) en la revista de divulgación científica Scientific Reports donde se plantea que los rompecabezas Sudoku pueden ser considerados problemas

de satisfacción de restricciones booleanas, y para resolverlos plantea un nuevo algoritmo que en primer lugar realiza una medición de la dificultad clasificándolo en niveles y posteriormente obtiene los valores que dan respuesta a todas las condiciones iniciales del Sudoku y se llega así a la resolución.

Ante este trabajo surge la posibilidad real de plantear una analogía entre las condiciones iniciales dadas del Sudoku y las restricciones que se presentan en el proceso de planificación de la producción.

Con los conocimientos adquiridos durante la formación académica es posible realizar un tratamiento óptimo de la información obtenida y plantear modelos computacionales que den respuesta a la problemática planteada en el próximo punto.

Con la investigación se obtendrán importantes conclusiones sobre la posibilidad de resolver planificaciones complejas mediante el uso de un nuevo algoritmo implementado en una computadora. Dejando abierta la posibilidad de que esta estructura de resolución pueda ser implementada en distintos ámbitos, no sólo industriales, y genere nuevas formas de resolver problemas conocidos.

1. Introducción

1. Introducción

El fin de este trabajo será adaptar el algoritmo determinista de Mària Ercsey-Ravasz y Zoltán Toroczkai publicado por la revista de divulgación científica Science Report en el artículo: “The chaos within Sudoku” (2012).

El algoritmo descripto es utilizado para resolver Sudokus complejos, el problema pasa por una fase de codificación a Forma Normal Conjuntiva (CNF) y luego se simula un sistema de ecuaciones, analizando la evolución de la solución ante distintos valores posibles, hasta hallar el resultado final que satisfaga todas las restricciones. El solucionador planteado por los autores pasa por momentos dónde la trayectoria resolutiva es “caótica” y mientras mayor sea el tiempo que el sistema permanece en ese estado más compleja será encontrar la solución.

El sistema dinámico propuesto por los autores incorpora la configuración de números de partida en el tablero y evoluciona desde una condición inicial aleatoria. Si la solución existe y es única, la evolución en tiempo de este sistema acabará alcanzándola.

En el artículo publicado queda descripta la aplicación del algoritmo a la resolución de Sudokus, fue demostrado por varios autores que este juego es un problema del tipo NP-Completo, uno de ellos fue Takayuki Yato en 2003. Además dadas sus características y debido a que puede ser expresado en forma normal conjuntiva, también se clasifica como SAT (problema de Satisfactibilidad Booleana, identificados por Cook en 1971), así lo describieron Lynce y Ouaknine en su trabajo “Sudoku as a SAT problema” (2006).

Existen varios ejemplos de problemas de satisfactibilidad booleana como: N-damas, Sudoku, Coloreado de mapas, equivalencias de circuitos combinacionales, programación y asignación de tareas. Esta última situación es la que se va a analizar en el presente trabajo, buscando en primer lugar explicar su grado de dificultad y su relevancia en la planificación organizacional, para luego aplicar el solucionador presentado por Ercsey-Ravasz y Toroczkai a la resolución del mismo.

La planificación de tareas organizacionales (Job Shop Scheduling) es un problema planteado desde hace casi medio siglo, ya que influye en todo el proceso productivo y en los planes operativos y estratégicos de la organización, las necesidades de materia prima, los tiempos de entrega y abastecimiento, pero también sirve para llevar a cabo controles de desviaciones en los tiempos pautados, de este modo al observar fluctuaciones en las esti-

maciones, (ya sea retrasos en lo pautado o “tiempos muertos”) se podrán tomar medidas correctivas para evitar implicaciones monetarias que afectarán la viabilidad de la organización.

Debido a que el Job Shop Scheduling también es del tipo NP-Completo cuando el número de máquinas es mayor o igual a 2 (Conway, Maxwell y Miller 1967), el cálculo de la secuencia óptima es extremadamente complejo (en especial cuando se presentan varias operaciones), es debido a su carácter combinatorio que dificulta la búsqueda de la solución, y que genera que la relación entre tamaño del problema y el tiempo de solución no sea lineal, suponiendo que al aumentar la complejidad el tiempo de resolución se dispara y los algoritmos planteados pierden su eficiencia.

Por estas razones se está en una permanente búsqueda de un algoritmo que optimice la resolución de la planificación, y que permita un cálculo más rápido, y además algún indicador de la dificultad de ese cálculo.

Caracterizar los sistemas caóticos es difícil, por la irregularidad de su comportamiento, por lo que lo habitual es introducir parámetros que caractericen el comportamiento a largo tiempo, ellos han decidido introducir un parámetro κ que mide la duración del transitorio caótico (que acaba desapareciendo porque el sistema dinámico acaba convergiendo a la solución exacta).

En definitiva esta investigación brinda una nueva manera de llevar a cabo la planificación de operaciones productivas, siendo este método eficiente cuando la complejidad imposibilita la solución utilizando modelos sencillos.

1.1 Objetivo general

Resolver la planificación de la producción midiendo su dificultad y agilizándola cuando se presente una problemática compleja, determinando la optimización computacional de los trabajos que se constituyen dentro de un proceso productivo, mediante un algoritmo que reduzca la longitud total del proceso y maximice los beneficios.

1.2 Objetivos particulares

- Optimizar la planificación de la producción mediante la codificación del problema y la posterior aplicación de un solucionador dinámico de tiempo continuo.

- Enmarcar la planificación dentro de los problemas NP complejos.
- Expresar el problema de Planificación de la producción como un SAT (Problema de Satisfacción de Restricciones).
- Codificar el problema planteado como SAT en forma normal conjuntiva (CNF).
- Plantear un algoritmo determinista de tiempo continuo que resuelva los SAT.
- Proporcionar un método para calcular el ratio de dificultad del problema de planificación.
- Crear un software que implemente la codificación y el algoritmo determinista, teniendo como ingreso los datos acerca del número de trabajos, de máquinas y los correspondientes tiempos de procesamiento de las actividades, para obtener un modelo óptimo de producción.
- Verificar la validez y el rendimiento del método propuesto.

1. Introducción

2 Marco contextual

La presente investigación se desarrolla en el marco de la optimización del proceso de planificación organizacional, específicamente en el contexto de la programación de operaciones considerando las tareas y máquinas involucradas en el proceso productivo.

Es decir que el enfoque utilizado para llevar a cabo el proyecto fue establecer un modelo de toma de decisiones que optimice la distribución y elaboración de un modelo eficaz para dar soporte a los planificadores y simplificar así la extremadamente compleja labor de secuenciar tareas, esperando obtener los mejores resultados.

En la actualidad se han ido desarrollando diferentes modelos que van desde reglas de despacho no refinadas, hasta técnicas de cuello de botella y algoritmos genéticos, los últimos 50 años han sido de avance permanente en torno a la optimización de la planificación, y mancomunadamente investigadores, científicos de gestión y expertos en producción han volcado su conocimiento en pos de obtener metodologías que mejoren en tiempo y calidad de decisión la consecución de soluciones para el problema. (“Scheduling”, Michael Pinedo, 2012)

Han surgido nuevos enfoques para intentar dar respuesta a la problemática planteada, tales como las redes neuronales, la computación evolutiva, la investigación en campos biológicos, genéticos y neurofísicos. Estos avances acrecientan aún más la idea de una problemática capaz de ser abordada multidisciplinarmente para obtener los mejores resultados.

Debe destacarse la intrínseca relación entre la planificación y la productividad, entendida esta última como la efectividad en la relación entre la producción obtenida y los recursos utilizados para obtenerla. Gerentes y responsables de planificación hacen énfasis a la hora de tomar la decisión de planificación de tareas (actividad inicial y sumamente prioritaria para obtener éxito operacional) en el aumento de la productividad, para ello es necesario el mejor uso de los recursos disponibles en la organización eliminando tiempos de espera innecesarios y evitando la sobrecarga de las máquinas, que genera horas extras de producción. (“Planificación eficiente”, Silva, 2007)

Para realizar correctamente el planeamiento operativo es necesario coordinar y conducir todas las operaciones del proceso considerando diversos factores, tales como el plazo, los recursos, los objetivos y necesidades a satisfacer, la filosofía de la organización y la flexibilidad necesaria para ajustarse a las contingencias. Una vez elaborado el plan por los responsables el sector de producción deberá llevar adelante la fabricación siguiendo sin desv-

íos el orden de las tareas, y serán los encargados del control que deberán cotejar el normal accionar de los empleados del nivel operativo con el plan de programación de tareas y operaciones realizado por los del nivel estratégico y táctico, para que en forma oportuna se realice la corrección de desvíos que posibiliten el alcance del objetivo general de la organización.

En el contexto de la optimización del proceso que utilizan los planificadores para secuenciar las tareas y determinar el uso de las máquinas se selecciona como objeto de estudio el Job Shop Scheduling, que se trata de un problema de optimización en la cual los recursos (máquinas) y operaciones (que componen cada trabajo) son asignados idealmente. El mismo es planteado como un problema de optimización combinatoria debido a que involucra elecciones entre alternativas discretas, buscando la solución óptima. Debido a la naturaleza del problema es extremadamente complejo encontrar la solución computacionalmente, perteneciendo estos a la clase NP-compleja, es decir que el tiempo de cómputo que se requiere para resolverlo se incrementa conforme crece el tamaño del problema, presentando una dependencia funcional tal que no admite ser acotada por un polinomio. ("Estado del arte del Job Shop Scheduling", Peña y Zumelzu, 2006).

Por lo expuesto es que se ha seleccionado como objeto de estudio, buscando proporcionar un nuevo método que posibilite hallar la mejor planificación para las tareas de una organización y para ello se decide utilizar un enfoque que plantea el Job Shop Scheduling como un problema de Satisfactibilidad Booleana y que luego de ser convertido a una representación en Forma Normal Conjuntiva es evaluado por un algoritmo determinista (puede ser definido de forma matemática explícita) de tiempo continuo (definido para todos los valores entre $-\infty$ e ∞) para obtener la mejor planificación de la producción.

Es un factor condicionante en la elaboración de esta investigación la naturaleza compleja del problema a analizar, y por tanto es necesario profundizar en distintos campos del conocimiento, desde la Gestión y Administración de tareas, pasando por la investigación de operaciones, la matemática compleja, hasta conocimientos físicos involucrados con ña Teoría del Caos. Esto se debe a que el algoritmo a utilizar para obtener la solución, presentado en el artículo "Chaos within Sudoku", publicado en agosto de 2012 por Mària Ercsey-Ravasz y Zoltán Toroczkai de las universidades Royal Babes de Rumania y Notre Dame de EEUU, también es multidisciplinario. En este artículo se aborda la resolución del popular rompecabezas Sudoku planteado como un problema de Satisfacción de Restricciones Boo-

2. Marco contextual

leanas utilizando un sistema dinámico continuo (conjunto de ecuaciones diferenciales ordinarias) notando que la dificultad que en ellos se observa tiene relación con una trayectoria de resolución que pasa un tiempo transitorio por el caos determinista.

Cabe destacar que aún no se han publicado trabajos que relacionen este algoritmo de resolución de Sudokus con la planificación de tareas industriales, pero sí existen otros que realizan abordajes utilizando por ejemplo redes neuronales o modelos biológicos para hallar los mejores resultados.

En la presente investigación se establece una relación entre el modelo dinámico de tiempo continuo introducido por Ercsey-Ravasz y Toroczkai y se determina que debido a que tanto la resolución de Sudokus como el Job Shop Scheduling se enmarcan y pueden ser expresados como problemas SAT (satisfacción de restricciones booleanas), perteneciendo por esto a la clase de problemas NP-complejos, pueden ser resueltos por el algoritmo propuesto por los autores.

Posteriormente se realiza un análisis acerca de la validez y el rendimiento del método propuesto en comparación con alguno de los ya mencionados.

2. Marco contextual

3 Marco teórico

En este punto se presenta la descripción del proceso de planificación, posteriormente se exponen sus características como problema NP completo, luego se estudian los SAT, el Sudoku y la investigación de Toroczkai y Ercsey-Ravasz.

Por último se realiza una explicación de la aplicación del solucionador propuesto al problema de la planificación.

3.1 Planificación de la producción

La planificación de la producción o Job Shop Scheduling trata acerca de las acciones y de sus efectos, es un concepto ligado a la toma decisiones en ámbitos relativos al control de la producción, en que las decisiones a tomar se centran en la asignación de recursos y la optimización de funciones, derivándose su complejidad en factores tales como restricciones en los abastecimientos, costos y tiempos de entrega. (Ander Egg - 1991)

En 1994 Pinedo definió a la programación de tareas como un problema de optimización combinatoria cuya función es la asignación de los recursos limitados a tareas a lo largo del tiempo, con la finalidad de optimizar los objetivos.

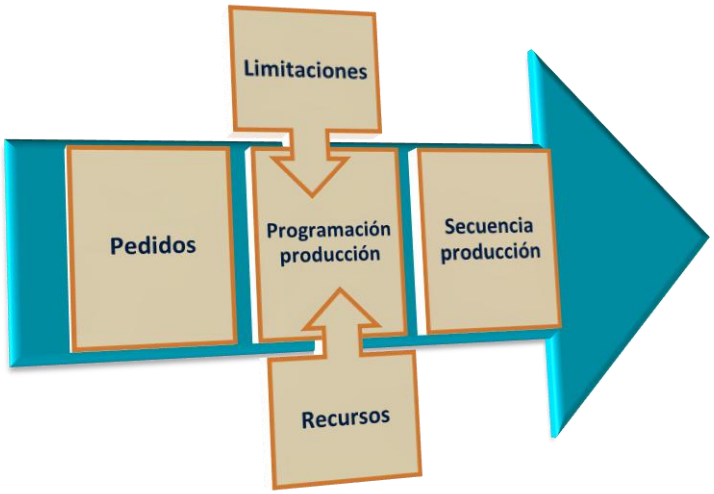


Ilustración 1 Proceso de planificación

3.1.1 Aspectos generales

La planificación es una actividad tan común en nuestra época que si se preguntase: ¿quiénes son los planificadores?, la respuesta conduciría, indudablemente, a una nueva pregunta: ¿quién no hace planes? Hoy más que nunca una buena parte de la humanidad mide, proyecta, experimenta, diseña, coordina, en suma, está planificando.

J. FRIEDMANN

Planificar es establecer un conjunto de acciones que sean medibles en su ejecución, ordenadas en un tiempo definido y con un orden estructurado que permiten alcanzar un fin con un mínimo de riesgos. Con este proceso se obtiene información clave que posibilita la toma de decisiones, así lo describe Ander Egg[1] en su libro Introducción a la planificación (1978).

Carlos Matus[2] publicó en 1971 que se debe entender la planificación como una dinámica de cálculo que precede y preside la acción, que no cesa nunca, que es un proceso continuo que acompaña la realidad cambiante. Gagne y Briggs[3] (1986) plantearon que planificar es una búsqueda constante y organizada de análisis de información clave para la toma de decisiones.

Sea cuál sea la interpretación que se realice acerca de este proceso, se desprende de estas definiciones que se trata de un proceso crítico, de gran valor, y que debe realizarse con las mayores certezas posibles, utilizando toda la información necesaria para tomar las mejores decisiones.

Existen varias etapas que componen el proceso de planificación, en primer lugar se identifica el problema, para así desarrollar alternativas, se elije la más conveniente y se ejecuta el plan.

En el contexto de la planificación productiva se destacan la planificación agregada y programación de la producción, la planificación de necesidades de materiales (MRP), la filosofía de producción Justo a tiempo, la programación temporal de proyectos (PERT/CMP) y el control de la función de producción.

[1] Ezequiel Ander Egg: pedagogo, filósofo, sociólogo y ensayista argentino, nacido en 1930.

[2] Matus: economista chileno, en 1977 escribió “Planificar situaciones”.

[3] Gagne y Briggs: autores de “Planificación de la enseñanza. Sus principios”

La planificación de tareas, desde hace ya 50 años, se ha convertido en un punto de estudio importante en distintos rubros del ámbito industrial. Se analiza la planificación de las tareas para las que existe cierta incertidumbre y los datos con los que se cuenta para así modelar la situación. Uno de los problemas más abordados es el Job Shop Scheduling que trata la asignación de tareas limitadas a recursos a lo largo del tiempo, con la finalidad de optimizar objetivos. Esta situación es descripta en el libro: "Applying constraint satisfaction techniques to job shop scheduling" de Cheng y Smith (1997)

Los recursos con los que se elabora la planificación pueden ser: máquinas, líneas productivas, materiales, unidades de procesamiento. Las tareas a planificar serán: operaciones del proceso productivo, etapas de un proyecto de ingeniería o ejecuciones.

Los planes se elaboran conociendo el plan maestro (que debe notificar cantidades y fechas en las que se necesitan los productos terminados), el estado de inventario y el árbol de fabricación.

La programación con restricciones es una tecnología ampliamente utilizada en la resolución de multitud de problemas de diversas áreas (Barber - 2003), incluyendo problemas de planificación y scheduling. En esta programación se destaca la satisfacción de restricciones y los problemas de optimización. Para resolverlos hay que modelar el problema, definiendo las variables, el dominio y las restricciones. Luego se aplica algún mecanismo de resolución, existiendo varias estrategias para abordarlas, la mayoría se trata de algoritmos de búsqueda, técnicas de consistencias y técnicas híbridas.

3.1.2 Definición problema

Los procesos de manufactura se han vuelto muy complejos debido a que las máquinas que intervienen en estas tareas pueden ejecutar un mayor número de operaciones (Administración de producción y operaciones – Gaither y Frazier - 2000). Lo que se persigue es reducir los costos de operación, es relevante entonces para los sistemas de manufactura solucionar el scheduling, determinando los tiempos y las máquinas que conformarán la secuencia operativa.

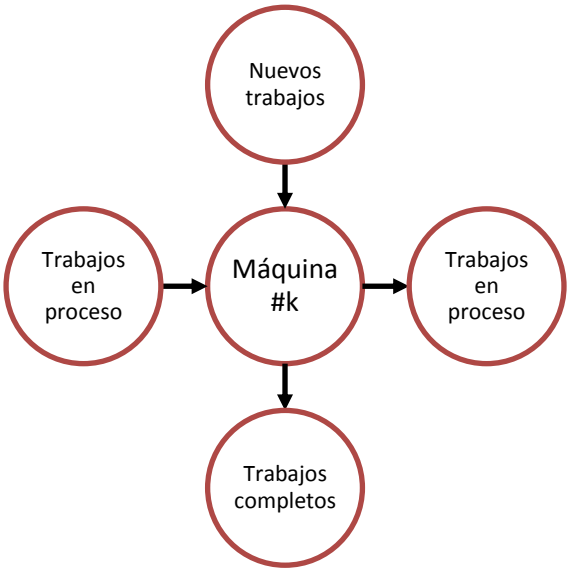


Ilustración 2 Flujo de trabajo en JSSP

Fogarty, Blackstone y Hoffmann (1991) definen que, la planeación de producción es la fijación de objetivos de producción cuantificables en el tiempo (planeación estratégica) y la toma de decisiones sobre la determinación de cómo lograrlos; por lo que esto implica determinar las cantidades a producir para un horizonte temporal de planificación, conformando lo que se denomina un plan de producción. La programación de operaciones (scheduling), en un sentido amplio, puede pensarse como; la asignación de recursos (máquinas) en un lapso de tiempo para realizar un conjunto de trabajos, o bien como resolver el problema de encontrar la asignación óptima de ciertos recursos a determinados tareas. Por lo que, en un problema de planificación siempre existirá tres componentes diferenciados: las tareas u operaciones (trabajos) que se tienen que realizar, los recursos disponibles para su realización (máquinas), y las finalidades u objetivos (función objetivo) que se desean lograr y que permiten identificar, entre varias planificaciones posibles, aquellas que sean óptimas. De acuerdo a los componentes anteriores, los problemas de planificación se dividen en dos: determinísticos y estocásticos.

En 1963 Fischer y Thompson propusieron el clásico problema de programación minimizando el makespan: J/Cmax, es decir un pequeño ejemplo de un problema de Scheduling tipo Job-Shop, con diez trabajos y diez máquinas, este problema intentó ser resuelto por más de 25 años, hasta que finalmente en 1989 Carlier y Pinson lo lograron. Pero durante ese tiempo otros investigadores han intentado resolverlo y descubrieron algunas nociones

3. Marco teórico

interesantes, por ejemplo Kan (1978) pudo determinar que el problema del Job Shop Scheduling es del tipo NP-duro.

Dentro del scheduling el problema de Job shop[4] es el que más aplicación práctica ofrece, permitiendo incrementar la eficiencia de los procesos de manufactura, esta área engloba varios problemas en los que es necesario determinar un plan de ejecución para un conjunto de tareas que pueden estar relacionas entre sí por restricciones de precedencia. La resolución del problema consistirá en obtener un plan de ejecución que, satisfaciendo tanto las restricciones de precedencia como las de recursos, optimice alguna función objetivo, que suele estar relacionada con el tiempo. En la mayoría de los casos esta función objetivo es el makespan o tiempo de finalización de la última operación realizada.

Existen multitud de propuestas para resolver el JSSP[5]. Entre las más conocidas y referenciadas se encuentran: programación disyuntiva, heurísticas basadas en cuello de botella, programación con restricciones y métodos heurísticos.

El problema (como lo enuncia Michael Pinedo en su libro Scheduling de 1994) consiste en programar de forma óptima, un conjunto de N tareas, que deben ser procesadas en un conjunto de M máquinas considerando que las tareas tienen la misma secuencia de producción. El objetivo es minimizar el tiempo total requerido para terminar las tareas (makespan). Los parámetros más relevantes son: el índice i que corresponde al número de tarea, j que corresponde al número de máquina y P_{ij} que es el tiempo de procesamiento requerido por la tarea i en la máquina j. Por esta razón el trabajo consiste en M operaciones y la j-ésima operación de cada trabajo debe ser procesada en la máquina j.

Lo que persiguen los algoritmos de resolución es encontrar la secuencia de los trabajos a programar, la cual debe minimizar el makespan[6]. En este contexto se considera el trabajo *i* como un conjunto de operaciones, pasando por cada máquina *j* una sola vez.

Cada tarea θ_{ij} tiene asociado un tiempo de procesamiento sin interrupción de d_{uij} unidades de tiempo durante el cual requiere del uso exclusivo de un único recurso. Cada trabajo tiene un tiempo de inicio más temprano y en ocasiones se considera también un tiempo de término más tardío, lo que obliga a que los tiempos de inicio de las tareas tomen valores en

[4] Job shop: problema de optimización combinatoria, del tipo NP difícil, busca determinar la programación dada por los tiempos de inicio y finalización de cada operación, de tal forma que se minimice el tiempo de realización de todas las operaciones.

[5] Job Shop Scheduling Problem

[6] Makespan: tiempo total en el que todos los trabajos completan su ejecución

dominios finitos. Cabe considerar que las operaciones de un mismo trabajo deben ejecutarse en un orden determinado, de forma que θ(i+1)j no puede comenzar hasta que θij haya terminado completamente. Además, las operaciones que comparten una misma máquina son no interrumpibles y mutuamente exclusivas. Las restricciones del problema se pueden resumir de la siguiente manera:

- Una tarea no puede visitar una misma máquina dos veces.
- No hay restricciones de precedencia entre operaciones de distintas tareas.
- Las operaciones no se pueden interrumpir.
- Cada máquina puede procesar sólo una tarea a la vez.
- No se especifican ni *release times* (fecha de tarea lista a ser procesada) ni *due dates* (fecha de entrega).

El objetivo consiste en encontrar una planificación factible, es decir, una asignación de tiempos de inicio s_{tij} para cada una de las tareas, que minimice el makespan (Cmax) o tiempo de finalización de la última tarea. El job shop se representa a través de un grafo conocido como grafo disyuntivo.

Las variantes para el problema del scheduling se relacionan con la secuencia de ordenación de las operaciones, denominada patrón de flujo (Industrial Scheduling – Muth y Thompson). Si todos los trabajos tienen la misma secuencia de ordenación de las operaciones, el problema se conoce como flow-shop, mientras que si existe una ordenación determinada para cada uno de los trabajos y también hay un orden de uso de cada uno de los recursos por parte de las operaciones, el problema se denomina permutation flow-shop. Si no existe ninguna restricción de ordenamiento, el problema pasa a llamarse open-shop. El job-shop corresponde al problema en el cual cada trabajo tiene una ordenación determinada.

Por cada trabajo se establece una ordenación determinada entre sus tareas. En la ilustración 1 se muestra un ejemplo de este tipo de problema, donde T_i - 1 ¸ i ¸ 4 es un trabajo, y R_j - 1 ¸ j ¸ 5 es el recurso que emplea la tarea j que lo compone.

T_1	R_2	R_3	R_1	R_4	R_5
T_2	R_3	R_1	R_4	R_5	R_2
T_3	R_1	R_3	R_5	R_4	R_2
T_4	R_4	R_5	R_1	R_3	R_2

Ilustración 3 Ordenación de tareas y recursos establecidos para cada trabajo de un JSSP

3. Marco teórico

En el JSSP puesto que cada secuencia de operaciones puede ser permutada independientemente de la secuencia de operaciones de otra máquina, el número total de posibles soluciones para el JSSP es $(n!)^m$, donde n denota el número de trabajos y m el número de máquinas. Este número constituye el espacio de búsqueda del problema.

El siguiente ejemplo muestra el tamaño del problema para resolverse por métodos clásicos de búsqueda:

Suponiendo un problema de 24 operaciones y 1 máquina, el espacio de búsqueda para este problema es (24!) = 6:20 x 10^{23} posibles soluciones. Ahora bien, suponiendo que la edad del universo es 20 mil millones de años tenemos que 20 x 10^9 x 365.25 x 24 x 60 x 60 = 6.31 x 10^{17} segundos, que es aproximadamente 6.31x10^{23} μ seg. Si existiera una computadora capaz de procesar una solución cada μseg, aún considerando toda la edad del universo, apenas se hubiera podido resolver este problema por enumeración, esto da una idea de la complejidad del JSSP.

3.1.3 Representación gráfica

En el libro de Michael Pinedo "Scheduling" (1994) se deja especificada la representación del Job Shop Problem con un grafo disyuntivo G= (V, A ∪ D), en la que cada nodo del conjunto V representa una tarea del problema, además se incluyen dos nodos 0 y ∞ que corresponden a tareas ficticias con duración 0 y tales que la tarea 0 ha de preceder a todas las demás tareas y la tarea ∞ ha de ser posterior a las otras.

Los arcos A son llamados arcos conjuntivos y representan las restricciones de precedencia entre tareas. Los arcos D corresponden a los arcos disyuntivos y representan las restricciones de capacidad de las máquinas.

Cada uno de los arcos posee un peso equivalente al tiempo que tarda en procesarse la tarea de la que parte el arco en nuestro problema.

Un grafo solución determina unívocamente un orden procesamiento de tareas y viceversa, un orden de procesamiento de tareas puede representarse como un grafo.

El makespan es el tiempo de finalización de la última tarea, o en el grafo la longitud del camino más largo entre el nodo inicial y el final.

3. Marco teórico

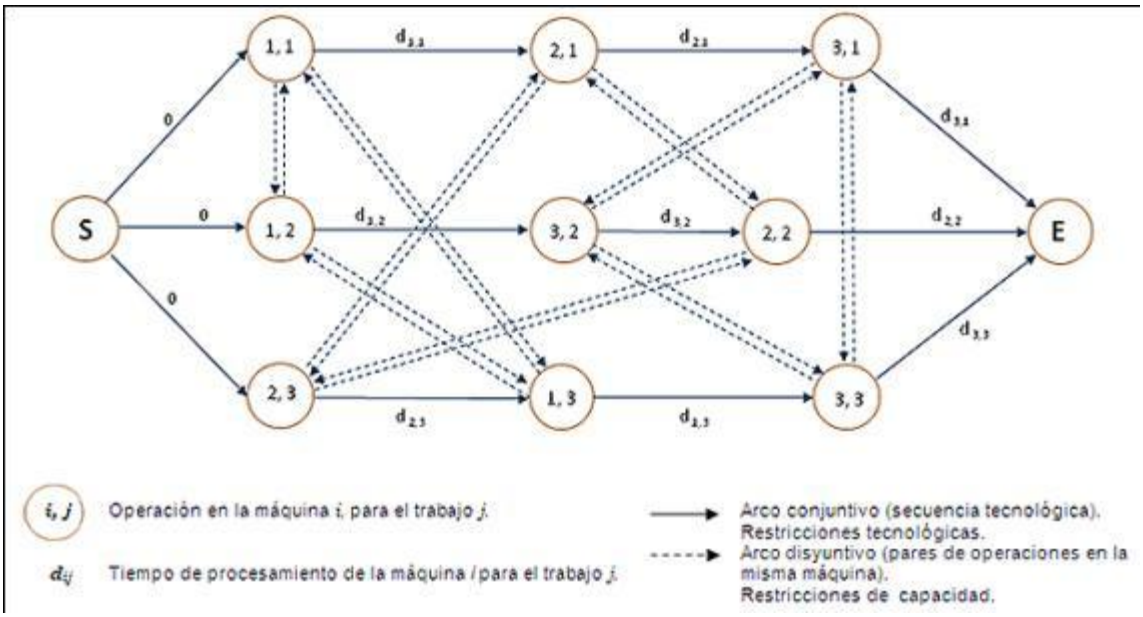


Ilustración 4 Grafo disyuntivo para un problema de 3 trabajos en 3 máquinas

3.1.4 Modelo matemático

Planificar es tomar decisiones óptimas en cuanto a la decisión de secuenciación de operaciones.

A continuación se presenta el modelo que Michael Pinedo propuso en su libro Scheduling, Theory, Algorithm and Systems (1994):

$J_n \,|prec\,|C_{\max}$

Donde, **J** indica que se trata de un problema de Job shop con **n** máquinas, **prec** señala que se tienen restricciones de procesamiento definidas por los tiempos en que cada operación esté lista (ready time) y sus tiempos de terminación obligatorios (deadline) de cada operación y $\mathbf{C_{max}}$ es la función objetivo a minimizar, llamada Makespan.

La función objetivo que se busca es la de obtener un schedule que minimice el tiempo de terminación máximo de la última tarea en el sistema ó Makespan. El Makespan se representa por max *Ci*. De esta forma la función objetivo *Z* es el minimizar *Ci* (el máximo tiempo de término encontrado de una operación i):

$Z=\min(\max C_i)$

Sujeto a (con i y j como par de operaciones):

$C_i \leq s_j$

$cap_{ij} = (C_i \leq s_j) \vee (C_j \leq s_i)$

$s_i \geq r_i$

$C_i \leq d_i$

Donde:

$$C_i = s_i + p_i$$

Este problema se puede ver como uno de satisfacción de restricciones (CSP)[7] que consiste en determinar si un conjunto de restricciones puede ser satisfecho. La solución de CSP es una asignación para las variables que intervienen, que simultáneamente las satisfaga a todas ellas. La mayor dificultad de solucionar este problema proviene de las interacciones entre restricciones y, en consecuencia CSP se torna difícil.

Parámetros

M	Número de máquinas, índice i=1,…,m
N	Números de trabajo índice j=1,…,n
h_j	Número de operaciones del trabajo J_j, índice h=1,…,h_j
$O_{j,h}$	Operación h-ésima del trabajo J_j
$M_{j,h}$	Subconjunto de máquinas en que la operación $O_{j,h}$ puede ser asignada
$P_{i,j,h}$	Tiempo de proceso de operación $O_{j,h}$ en la máquina M_i, $M_i \in M_{j,h}$, $p_{i,j,h} > 0$
L	Un número grande

$$a_{i,j,h} = \begin{cases} 1 \; si \; la \; máquina \; M_i \; puede \; procesar \; la \; operación \; O_{j,h} \\ 0 \; en \; otro \; caso \end{cases}$$

[7] CSP: problemas de satisfacción de restricciones, son problemas matemáticos que se define como un conjunto de objetos cuyo estado debe cumplir una serie de restricciones o limitaciones.

Variables de decisión:

C_{max}	Makespan o tiempo de completitud de los trabajos
k_i	Número de operaciones asignadas a la máquina M_i
$Tm_{i,k}$	Tiempo de inicio de la k-ésima operación de la máquina M_i
$t_{j,h}$	Tiempo de inicio de la operación $O_{j,h}$

$$a_{i,j,h} = \begin{cases} 1\ si\ la\ operación\ O_{j,h}\ es\ asignada\ a\ M_i\ \in\ M_{j,h}\ en\ k-ésimo\ lugar, k = 1, \dots, k_i \\ 0\ en\ otro\ caso \end{cases}$$

$$y_{i,j,h} = \begin{cases} 1\ si\ la\ operación\ O_{j,h}\ es\ asignada\ a\ M_i \in M_{j,h} \\ 0\ en\ otro\ caso \end{cases}$$

Modelo matemático:

Min C_{max} (1)

s.a

$$C_{max} \geq t_{j,h_j} + P_{s_j,h_j} \forall\ j = 1, \dots, n \quad (2)$$

$$\sum_i y_{i,j,h}.p_{i,j,h} = P_{s_j,h}\ \forall\ j = 1, \dots, n; h = 1, \dots, h_j; \quad (3)$$

$$t_{j,h} + P_{s_j,h} \leq t_{j,h+1} \forall\ j = 1, \dots, n; h = 1, \dots, h_j - 1; \quad (4)$$

$$Tm_{i,k} + p_{i,j,h}.x_{i,j,h,k} \leq Tm_{i,k+1} \forall\ i = 1, \dots, m; j = 1, \dots, n; h = 1, \dots, h_j; k = 1, \dots, k_i - 1; \quad (5)$$

$$Tm_{i,k} \leq t_{j,h} + \left(1 - x_{i,j,h,k}\right).L\ \forall\ i = 1, \dots, m; j = 1, \dots, n; h = 1, \dots, h_j; k = 1, \dots, k_i \quad (6)$$

$$Tm_{i,k} + \left(1 - x_{i,j,h,k}\right).L \leq t_{j,h} \forall i = 1, \dots, m; j = 1, \dots, n; h = 1, \dots, h_j; k = 1, \dots, k_i; \quad (7)$$

$$y_{i,j,h} \leq a_{i,j,h} \forall\ i = 1, \dots, m; j = 1, \dots, n; h = 1, \dots, h_j \quad (8)$$

3. Marco teórico

$$\sum_{j}\sum_{h} x_{i,j,h,k} \leq 1 \forall\ i = 1, \dots, m; k = 1, \dots, k_i; \quad (9)$$

$$\sum_{i} y_{i,j,h} = 1\ \forall\ j = 1, \dots, n; h = 1, \dots, h_j; \quad (10)$$

$$\sum_{k} x_{i,j,h,k} = y_{i,j,h} \forall i = 1, \dots, m; j = 1, \dots, n; h = 1, \dots, h_j; \quad (11)$$

$$t_{j,h} \geq 0 \forall j = 1, \dots, n; h = 1, \dots, h_j; \quad (12)$$

$$Ps_{j,h} \geq 0 \forall j = 1, \dots, n; h = 1, \dots, h_j; \quad (13)$$

$$Tm_{i,k} \geq 0 \forall i = 1, \dots, m; k = 1, \dots, k_i; \quad (14)$$

$$x_{i,j,h,k} \in \{0,1\} \forall i = 1, \dots, m; j = 1, \dots, n; h = 1, \dots, h_j; k = 1, \dots, k_i; \quad (15)$$

$$y_{i,j,h} \in \{0,1\} \forall i = 1, \dots, m; j = 1, \dots, n; h = 1, \dots, h_j; \quad (16)$$

La función objetivo (1) corresponde a la minimización del makespan, determinado por la restricción (2) como el mayor tiempo de finalización de las operaciones. La restricción (3) indica el tiempo de proceso por operación luego de la selección de la máquina. La restricción 4 determina la secuencia de operaciones, según los tiempos de inicio de las operaciones en cada máquina j, la restricción 5 obliga que cada máquina procese más de una operación a la vez. Las restricciones 6 y 7 no permiten que una máquina procese más de una operación o que una operación sea procesada por más de una máquina. Con las restricciones 8, 9, 10 y 11 se expresa que cada operación debe ser asignada en una sola de las máquinas pertenecientes al conjunto de máquinas factibles para la operación.

Las restricciones 12, 13 y 14 establecen la no negatividad para los tiempos de inicio, proceso e inicio de trabajo de las máquinas respectivamente. Por último, las ecuaciones 15 y 16 establecen la binariedad para las variables de secuenciamiento y asignación.

Dado su naturaleza combinatoria el número de variables y los tiempos de cómputo crecen, permitiendo resolver sólo instancias pequeñas. En las medianas o grandes el modelo permite calcular cotas superiores o inferiores, que no corresponden necesariamente a soluciones factibles para el problema. Por este motivo es necesario buscar nuevos métodos que permitan encontrar soluciones.

3. Marco teórico

3.1.5 Ejemplo de modelo

Partiendo de lo anteriormente expuesto, es decir:

- n trabajos, m máquinas
- Cada trabajo se constituye en una ruta predeterminada (secuencia de máquinas).

El problema tiene los siguientes elementos:

- Operación (i,j): procesamiento del trabajo j en la máquina i.
- Tiempo de procesamiento pij
- Los trabajos no se reprocesan
- Minimizar el C_{max}

Para 3 trabajos en 4 máquinas:

Trabajos	Secuencia de máquinas	Tiempos de proceso
1	1,2,3	p_{11}=10, p_{21}=8, p_{31}=4
2	2,1,4,3	p_{22}=8, p_{12}=3, p_{42}=5, p_{32}=6
3	1,2,4	p_{13}=4, p_{23}=7, p_{43}=3

Tabla 1 3 Trabajos, 4 Máquinas

Formulación:

Variables:

t_{ij} = tiempo de inicio del trabajo j en la máquina i, para i = 1,2,3 y j =1,2,3

x_{ijk} = 1, si el trabajo j precede al trabajo k en la máquina i.

0, de lo contrario. Para i = 1,2,3,4

j =1,2,3 y j < k

Min C_{max}

s.t

$C_{max} \geq t_{31} + p_{31}$

$C_{max} \geq t_{32} + p_{32}$

3. Marco teórico

$C_{max} \geq t_{33} + p_{43}$

$t_{21} \geq t_{11} + p_{11}$

$t_{31} \geq t_{21} + p_{21}$

$t_{12} \geq t_{22} + p_{22}$

$t_{42} \geq t_{12} + p_{12}$

$t_{32} \geq t_{42} + p_{42}$

$t_{23} \geq t_{13} + p_{13}$

$t_{43} \geq t_{23} + p_{23}$

$t_{i1} + p_{i1} \leq t_{i2} + M(1-x_{i12})$ para i=1,2,3,4

$t_{i2} + p_{i2} \leq t_{i1} + M\, x_{i12}$

$t_{i1} + p_{i1} \leq t_{i3} + M(1-x_{i13})$

$t_{i3} + p_{i3} \leq t_{i1} + M\, x_{i13}$

$t_{i2} + p_{i2} \leq t_{i3} + M(1-x_{i23})$

$t_{i3} + p_{i3} \leq t_{i2} + M\, x_{i23}$

$t_{ij} \geq 0,$ para i=1,2,3,4 y j=1,2,3

$x_{ijk} \in \{0,1\}$ para i = 1,2,3,4 j =1,2,3 y j < k

Gráfico disyuntivo:

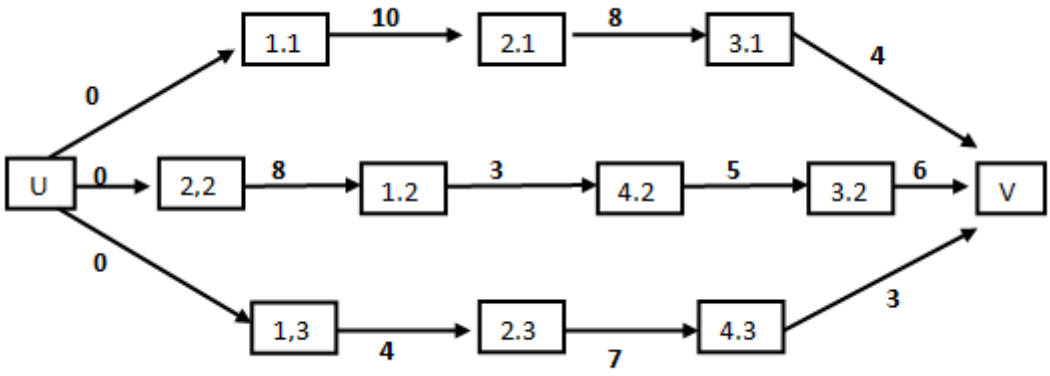


Ilustración 5 Gráfico disyuntivo

3. Marco teórico

3.1.6 Técnicas de optimización

El scheduling es tratado en muchos libros y estudios, existen distintas técnicas que se propusieron para abordar su resolución, pero aún existen grandes enigmas a disipar cuando se presentan determinadas condiciones iniciales y los tiempos de solución se acotan.

A continuación se presenta una breve descripción de algunos de los algoritmos propuestos, las técnicas van desde reglas de despacho (dispatching) no refinadas, hasta los algoritmos paralelos de ramificación y poda que son altamente sofisticados, y también los algoritmos genéticos paralelos. Los mismos fueron publicados en 2009 en el libro "Computational Intelligence in Flow Shop and Job Shop Scheduling" escrito por Chakraborty.

Con la aparición de nuevas metodologías, como redes neuronales y computación evolutiva, investigadores de campos como la biología, genética y neurofisiología han tenido también contribuciones regulares a la teoría de scheduling, poniendo de manifiesto la naturaleza multidisciplinar de este campo.

Existen:

• Técnicas de optimización, estas producen una solución globalmente óptima, pero requieren un tiempo de cómputo muy alto.

• Técnicas de aproximación, proporcionan una buena solución en un tiempo aceptable. A su vez pueden distinguirse varios subgrupos:

- Reglas de prioridades, fáciles de implementar cuando la cantidad de restricciones y variables es acotada. Las reglas de prioridad son probablemente las reglas heurísticas aplicadas con mayor frecuencia para resolver problemas de scheduling. Los algoritmos de Giffler y Thompson se pueden considerar como la base de todas las heurísticas basadas en reglas de prioridad. Propusieron dos algoritmos para generar schedules: procedimientos de generación de planificaciones: activos y no delay. Un schedule no delay tiene la propiedad de que ninguna máquina permanece ociosa si hay un job disponible para su procesamiento. Un schedule activo tiene la propiedad de que ninguna operación se puede iniciar tempranamente sin retrasar otro job. Exploran la estructura del problema mediante una estructura de árbol.

- Heurísticas de cuellos de botella, esta técnica requiere un mayor nivel de sofisticación. Dornforf y Pesch propusieron un algoritmo evolutivo basado en máquinas donde un cromosoma se codifica como una secuencia de máquinas y un schedule se construye con una heurística basada en corrimientos del cuello de botella sobre la secuencia. La heurística de corrimiento del cuello de botella, propuesto por Adams, Balas y Zawack, es probablemente el procedimiento más poderoso entre todas las heurísticas para el problema de job shop scheduling. Las máquinas se secuencian una a una, sucesivamente, tomando aquella máquina identificada como el cuello de botella entre las máquinas aún no secuenciadas. Cuando se secuencia una nueva máquina, todas las secuencias establecidas previamente se re-optimizan localmente. Tanto los procedimientos de identificación del cuello de botella y de re-optimización local se basan en la resolución repetida de un cierto problema de scheduling de una única máquina que es una simplificación del problema original. La principal contribución de esta opción es la forma en que se usa esta simplificación para decidir respecto del orden en el cual se secuencian las máquinas.

- Inteligencia Artificial, permiten desarrollar algoritmos de carácter general, centradas principalmente en sistemas basados en el conocimiento (KBS: Knowledge Based Systems), métodos de satisfacción de restricciones (CSP) y clausura. Un problema de planificación, desde el punto de vista de la inteligencia artificial, se especifica normalmente como sigue: dada una descripción del estado actual de un sistema, un conjunto de acciones que pueden realizarse sobre el sistema y una descripción de un conjunto de estados meta para el sistema, encontrar una secuencia de acciones para transformar el sistema en uno de los estados meta. En estos casos de alta complejidad en la programación, las técnicas de inteligencia artificial permiten encontrar mejores soluciones y en menor tiempo. Dichas técnicas y sus derivaciones han sido aplicadas en una gran variedad de campos de la gestión, de la producción, la simulación, decisiones de reemplazo de equipos, selección de proveedores, comercio colaborativo, optimización de redes y procesos multiobjetivo, entre otras.

3. Marco teórico

- Métodos de búsqueda local, considerados complementarios de los anteriores. La búsqueda local comienza desde una secuencia inicial (la mejor secuencia encontrada en la iteración) e intenta repetidamente mejorar la secuencia actual reemplazándola con soluciones vecinas. Si en el vecindario de la secuencia actual π se encuentra una mejor solución, ésta reemplaza a π y la búsqueda local continua desde la nueva solución. El algoritmo de búsqueda local, aplica estos pasos repetidamente hasta que se dé una de las siguientes condiciones: a) no se encuentra una mejor secuencia vecina, b) después de cuatro iteraciones que generan soluciones con la misma secuencia de tareas, ó c) después de diez iteraciones con soluciones diferentes, y mejores o iguales a la anterior.

- KBS: sistemas basados en el conocimiento (razonamiento temporal). Dentro de la Inteligencia Artificial y concretamente dentro de los sistemas basados en el conocimiento (de donde surgió la arquitectura REAKT, uno de los campos de mayor difusión ha sido la teoría de agentes inteligentes. Para poder incorporar conocimiento en un sistema que trabaja en un entorno con restricciones de tiempo real, se escogió la representación mediante el modelo de blackboard utilizando el paradigma de agente/sistemas multi-agente. Así, múltiples agentes pueden ejecutarse en paralelo, compartiendo datos comunes. Como consecuencia directa de esta línea de investigación se sigue trabajando en la teoría de agentes: arquitectura, comunicación y lenguajes. Adicionalmente, se sitúa el desarrollo de herramientas para la representación y tratamiento de hechos temporales (TempoClips).

- CSP: modelos de satisfacción de restricciones, son problemas matemáticos que se define como un conjunto de objetos cuyo estado debe cumplir una serie de restricciones o limitaciones. CSP representan las entidades de un problema como una colección homogénea de las limitaciones finitas de las variables, que se resuelve mediante métodos de satisfacción de restricciones. CSP son objeto de una intensa investigación, tanto en la inteligencia artificial y la investigación de operaciones, ya que la regularidad en su formulación proporciona una base común para analizar y resolver los problemas de muchas familias no relacionadas. CSPs a menudo presentan una alta complejidad, que requiere una combinación de

métodos y heurística de búsqueda combinatoria que hay que resolver en un plazo razonable. El problema satisfactibilidad booleana (SAT), las Modulo Teorías satisfactibilidad (SMT) y la programación de conjunto de respuestas (ASP) se puede considerar más o menos como ciertas formas del problema de satisfacción de restricciones. Esta metodología resulta útil para solucionar problemas combinatorios, temporales, de configuración, cuya resolución esté sujeta a la satisfacción de un conjunto de restricciones entre las variables del problema. Han demostrado su utilidad en aplicaciones de investigación operativa (generación de horarios, scheduling, etc.) La resolución de problemas mediante un CSP puede considerarse como un método fundamentalmente declarativo. De esta forma, basta especificar el problema a resolver como un CSP. Para resolver un CSP y obtener soluciones pueden utilizarse alguna de las múltiples herramientas disponibles, tanto comerciales, software libre, o librerías con procedimientos especializados en el manejo de restricciones.

En esta investigación se plantea el problema de la optimización de la planificación o scheduling como uno de satisfacción de restricciones booleanas (SAT), y se presenta un algoritmo que posibilite la resolución cuando los tiempos que se prevén con los otros métodos sean demasiado extensos o no encuentren un sistema de solución adecuado.

En los puntos sucesivos se realiza una descripción de los problemas SAT y se plantea el Job Shop Scheduling como un problema de optimización con restricciones.

3.1.7 Programación con restricciones

Tiene sus orígenes en la inteligencia artificial y comunidades de Informática. Sus inicios se dieron en 1970, según Edward Tsang, en Foundations of Constraint Satisfaction, los problemas de satisfacción de restricciones requieren de la búsqueda de una solución viable que satisfaga todas las restricciones dadas. Para facilitar la búsqueda de una solución a tal problema se desarrollaron varios lenguajes especiales. Sin embargo, durante la última década del siglo XX, la programación con restricciones no sólo se ha utilizado para la solución de problemas de factibilidad, sino también para los de optimización. Varios enfoques han sido desarrollados para facilitar la aplicación de la programación con restricciones a los problemas de optimización. Uno de tales enfoques es a través del lenguaje de

optimización de programación (OPL), que fue diseñado para modelar y resolver problemas de optimización a través tanto de las técnicas de programación y de los procedimientos de programación matemática.

Las características de un Problema de Satisfacción de Restricciones son las siguientes:

- El problema se puede representar mediante un conjunto de variables.

- Cada variable tiene un dominio de valores (un conjunto finito de valores discretos o intervalos de valores continuos)

- Existen restricciones de consistencia entre las variables (binarias, n-arias)

- Una solución es una asignación de valores a esas variables

Un problema de satisfacción de restricciones puede ser representado mediante una terna (X, D, C) donde:

- X es un conjunto de n variables $\{x_1, \ldots, x_n\}$.

- $D =< D_1, ..., D_n>$ es un tupla de dominios finitos donde se interpretan las variables X, tal que la i-esima componente Di es el dominio que contiene los posibles valores que pueden asignarse a la variable x_i. La cardinalidad de cada dominio es $d_i = |D_i|$.

- $C = \{c_1, c_2, \ldots, c_p\}$ es un conjunto finito de restricciones.

Cada restricción k-aria c_i está definida sobre un conjunto de k variables $var(c_i) \subseteq X$, denominado su ámbito, y restringe los valores que dichas variables pueden simultáneamente tomar.

Particularmente, una restricción es binaria cuando relaciona únicamente a dos variables x_i y x_j, y se suele denotar como c_{ij}. Todas las restricciones definidas en un CSP son conjuntivas, de manera que una solución debe de satisfacer a todas ellas.

La instanciación de una variable es un par variable-valor (x, a) que representa la asignación del valor a a la variable x (x = a). La instanciación de un conjunto de variables es una tupla de pares ordenados, donde cada par ordenado (x_i, a_i) asigna el valor $\{a_i \in D_i\}$ a la variable x_i.

Una tupla $((x_1, a1), \ldots, (x_i, a_i))$ es localmente consistente si satisface todas las restricciones formadas por variables $\{x_1, .., x_i\}$ de la tupla. Para simplificar la notación, se sustituye la tupla ((x1, a1), . . . , (xi, ai)) por (a1, . . . , ai).

3. Marco teórico

Un valor ai ϵ Di es un valor consistente para xi si existe al menos una solución del CSP en la cual xi= ai. El dominio mínimo de una variable xi es el conjunto de todos los valores consistentes para la variable, es decir, quedan excluidos aquellos valores que no forman parte de ninguna solución ($\forall x_i \in X, \forall a \in D_i, x_i =$ a forma parte de una solución del CSP).

Una solución a un CSP es una asignación (a1, a2, . . . , an) de valores a todas sus variables, de tal manera que se satisfagan todas las restricciones del CSP. Es decir, una solución es una tupla consistente que contiene todas las variables del problema. Una solución parcial es una tupla consistente que contiene algunas de las variables del problema.

Un CSP es consistente, si tiene al menos una solución, es decir una tupla consistente. Dos CSP son equivalentes si ambos representan el mismo conjunto de soluciones.

Una vez modelado el problema como un CSP, los objetivos del CSP consisten en la satisfactibilidad del mismo, es decir, obtener una o varias soluciones, sin preferencia alguna, o bien obtener una solución optima, o al menos una buena solución, en base a una función objetivo previamente definida en términos de algunas o todas las variables.

3.2 Lógica computacional

La planificación determinista es usada frecuentemente para plantear y resolver problemas de planificación y secuenciación de tareas (Scheduling), su resolución exige el uso de medios de computación adecuados, por esto es que en este apartado se realiza una breve introducción a los aspectos computacionales implicados en la resolución del Job Shop Scheduling mediante el método propuesto.

En el libro “Dynamic Logic” de Harel, Kozen y Tiuryn (2009), quedan expuestos varios conceptos concernientes a la planificación y la lógica computacional, describen que un algoritmo es una sucesión finita de operaciones elementales, que se ejecuta sobre una serie de datos para obtener un resultado, para medir su eficiencia se analiza el tiempo de ejecución, la memoria principal y secundaria que utiliza, y es importante aislar ese análisis de la máquina en la que se utilice.

La teoría de la computación es una rama de la matemática y la computación que focaliza su estudio en las capacidades y limitaciones de las computadoras, específicamente busca la creación de modelos matemáticos y la clasificación de problemas.

Existen tres modelos principales son: los autómatas finitos[8], los autómatas con pilas[9] y las máquinas de Turing[10], cada uno tiene variantes deterministas y no deterministas.

Lo que se estudia es además la posibilidad de solucionar problemas mediante algoritmos, es decir que si se descubre problemas imposibles no se intentará resolverlos algorítmicamente, ahorrando así esfuerzo. Existe por ello una clasificación de los problemas que distingue los computables, los semicomputables y los no computables.

Pero aún siendo del tipo computable puede que en la práctica no se halle solución, debido a que la memoria requerida, o el tiempo de ejecución para realizar el procesamiento sean excesivos. De este análisis surgen las clases de complejidad computacional.

Estas clases son conjuntos de problemas que poseen la misma complejidad computacional. En el libro "Theory of autómata" de Kumar (2010) se explica que existen dos grandes clases de complejidad las P y NP. La primera es el conjunto de todos los problemas de decisión que pueden ser resueltos por una máquina determinista en tiempo polinómico en proporción a los datos de entrada. La segunda es el conjunto de todos los problemas de decisión que pueden resolverse por una máquina no determinista en tiempo polinómico.

La importancia de la clase NP es que contiene muchos problemas de búsqueda y de optimización para los que se desea saber si existe una cierta solución o si existe una mejor solución que las conocidas. En esta clase se encuentran: el problema del viajante dónde se quiere saber si existe una ruta óptima que pasa por todos los nodos en un cierto grafo y el problema de satisfactibilidad booleana (SAT) en dónde se desea saber si una cierta fórmula de lógica proposicional puede ser cierta para algún conjunto de valores booleanos para las variables.

Dada su importancia, se han hecho muchos esfuerzos para encontrar algoritmos que resuelvan algún problema de NP en tiempo polinómico. Sin embargo, pareciera que para algunos problemas de NP (los del conjunto NP-completo) no es posible encontrar un algoritmo mejor que simplemente realizar una búsqueda exhaustiva.

Algo vital en la resolución de problemas es determinar si pertenecen a la clase NP-Completo, esta clase contiene los más difíciles de la clase NP, son los que más lejanos se

[8] Modelos con cantidad limitada de memoria.

[9] Modelos con gran capacidad de memoria, pero que solo puede manipularse como pila (el último almacenado es el siguiente leído).

[10] Mucha memoria almacenada como cinta, con la capacidad de simular autómatas.

encuentran de los del tipo P. Garey y Johnson escribieron en 1979 el primer libro sobre la teoría de la NP completitud, de allí se obtiene que la propiedad principal es que no puede ser resuelto en tiempo polimonial. Desde el punto de vista práctico, el fenómeno de la NP-completitud puede prevenir la pérdida de tiempo cuando se busca un algoritmo de tiempo polinomial no existente para resolver un problema determinado.

"Es dudoso que el género humano logre crear un enigma que el mismo ingenio humano no resuelva."

Edgar Allan Poe

Existen todavía desafíos que los hombres debemos resolver, uno de ellos es la relación entre las clases de complejidad P y NP a la que la teoría de la complejidad computacional no ha podido dar respuesta. Esto significaría verificar rápidamente soluciones positivas a un problema del tipo Si/No.

El problema de P versus NP se considera el problema abierto más importante en la ciencia de la computación. De hecho el Instituto Clay lo incluye entre los 7 "problemas del milenio", que corresponden a los problemas matemáticos abiertos más importantes según ese instituto.

La pregunta que propone el problema P vs NP es: suponiendo que la solución a un problema puede ser verificada rápidamente con una computadora. Entonces ¿es posible que la solución en sí misma pueda ser computada rápidamente?

3.2.1 Problemas SAT (Satisfactibilidad Booleana)

SAT es un problema fundamental en la lógica matemática, la teoría de la computación, la inferencia, el razonamiento automatizado y la ingeniería entre otras áreas. Dicho problema se refiere a la tarea de encontrar asignaciones que hagan verdadera una expresión booleana (Hoos 2005). Su importancia reside en el hecho de que es uno de los problemas más simples para los que se demostró que es NP- Completo (se expone en la siguiente sección). Siendo útil en la práctica para demostraciones automáticas, teoremas, verificaciones y detecciones de fallas de hardware, detección de isomorfismos de grafos, y hasta para plantear el problema del viajante.

3. Marco teórico

El problema de satisfactibilidad consiste en la asignación de valores de verdad a fórmulas en lógica proposicional. En el documento escrito en 1978 por Schaefer, "The complexity of satisfiability problems", da su significado y características. El objetivo del problema de satisfactibilidad es determinar si existe una asignación de valores de verdad para las variables booleanas que hagan verdadera a una fórmula. Una variable booleana es una variable que toma alguno de los valores 0 (falso) o 1 (verdadero). Las variables booleanas se consideran proposiciones, que son los objetos elementales de la lógica proposicional. El valor de la variable especifica la verdad o falsedad de la proposición. La proposición x es verdadera cuando a la variable booleana se le asigna el valor 1, cuando el valor asignado es 0 la proposición es falsa. Una asignación de verdad 1 es una función que asigna 0 o 1 a toda variable booleana.

Las conectivas lógicas ∧ (conjunción), ∨ (disyunción) y ¬ (negación) se usan para construir proposiciones llamadas fórmulas bien formadas a partir de un conjunto de variables booleanas.

Siendo V un conjunto de variables booleanas:

1. Si x ∈ *V*, entonces x es una formula bien formada.

2. Si x, y ∈ V, entonces (¬x), (x ^ y) y (x ∨ y) son fórmulas bien formadas.

3. Una expresión es una fórmula bien formada sobre V solamente si puede ser obtenida a partir de las variables booleanas en el conjunto V por un número finito de aplicaciones de las operaciones dadas en 2.

Se dice que una fórmula satisface una asignación de verdad si los valores de las variables hacen que *u* tenga el valor 1. Una cláusula es una formula bien formada que consiste de una disyunción de variables o la negación de variables. Una variable no negada se llama literal positivo y una variable negada literal negativo. Usando esta terminología, una cláusula es una disyunción de literales, las fórmulas:

- x ∨ ¬y
- ¬x ∨ z ∨ ¬y
- x ∨ z ∨ ¬x

Son cláusulas sobre el conjunto de variables V = {x,y,z}.

3. Marco teórico

Una fórmula está en forma normal conjuntiva si tiene la forma $u=u_1 \wedge u_2 \wedge \ldots \wedge u_m$, dónde cada u_i es una cláusula. Un teorema clásico de lógica proposicional asegura que toda fórmula bien formada se puede transformar en una fórmula equivalente en forma normal conjuntiva. Entonces, de manera rigurosa el problema de satisfactibilidad es el problema de decidir si una fórmula expresada en forma normal conjuntiva se satisface por alguna asignación de verdad.

Algunos de las variantes y problemas equivalentes a SAT son las siguientes:

- SAT(k): dada una fórmula en FNC con a lo sumo k literales por cláusula, determinar si esta es satisfacible.
- SAT*(k): dada una fórmula en FNC con exactamente k literales por cláusula, determinar si esa es satisfacible.
- SAT(3): dada una fórmula en FNC con a lo sumo 3 literales por cláusula, determinar si esta es satisfacible.

3.2.2 SAT es NP-Completo

El concepto de NP-completitud se desarrolló a finales de los años 60 hasta principios de los años 70 en la investigación paralela en los Estados Unidos y la Unión Soviética. En los EE.UU., en 1971, Stephen Cook, publicó un documento al respecto ("The complexity of theorem proving procedures"[11]). Posteriormente Richard Karp escribió "Reducibility among combinatorial problems"[12] (1972), generando un nuevo interés en el documento anterior, que listaba 21 problemas NP-Completos.

Otra contribución importante de Karp fue el anuncio de que muchos de los problemas NP-completos que parecían insolubles en el peor de los casos, son generalmente fáciles para la mayoría de los casos generados aleatoriamente. En 1984, Levin extendió este resultado al demostrar que algunos de los problemas son NP completos en el caso promedio.

El interés teórico de NP-completitud se ha visto reforzado por el trabajo de Theodore P. Baker, John Gill, y Robert Solovay, ellos mostraron en 1975 que la solución de los problemas de NP en el modelo de máquina de Turing requiere tiempo exponencial.

[11] La complejidad de los procedimientos de demostración de teoremas
[12] Reducibilidad de los problemas combinatorios

En la Unión Soviética, un resultado equivalente al de Baker, Gill, y Solovay fue publicado en 1969 por M. Dekhtiar.

Levin publicó en 1973 "los problemas de búsqueda universal". Su enfoque es un poco diferente del de Cook y Karp, ya que considera que los problemas de búsqueda requieren soluciones en lugar de determinar si existen. Suministra en su documento seis de estos problemas de búsqueda NP-completo, o problemas universales, y, además, encontró que cada uno tiene un algoritmo que lo resuelve en el tiempo perfecto.

A continuación se presenta el enunciado y la demostración del teorema de Cook - Levin:

"El Problema de satisfactibilidad booleana (SAT) es NP-completo." Cook

El problema de SAT pertenece a NP dado que puede ser resuelto con una máquina de Turing no-determinista que genera todas las posibles combinaciones de valores para las variables de la expresión y, en forma no-determinista intenta verificar si alguna de ellas hace que la expresión se evalúe en verdadero, en cuyo caso acepta la entrada.

Suponiendo que un problema perteneciente a NP es resuelto por una máquina de Turing $M = (Q, \Sigma, i, F, \delta)$ (donde Q es el conjunto de estados, Σ es el alfabeto de símbolos de la cinta, $i \in Q$ es el estado inicial, $F \subseteq Q$ es el conjunto de estados de aceptación y $\delta \subseteq Q \times \Sigma \times Q \times \Sigma \times \{-1,+1\}$ el conjunto de transiciones) y que M acepta o rechaza una instancia del problema en tiempo $p(n)$ donde n es el tamaño de la instancia y p es una función polinómica.

Para cada instancia "I" del problema se describe una expresión booleana que es satisfacible si y sólo si la máquina M acepta a I.

La expresión booleana utiliza las variables descritas en la siguiente tabla en la cual $q \in Q$, $-p(n) \leq i \leq p(n)$, $j \in \Sigma$, y $0 \leq k \leq p(n)$

Variables	Interpretación	¿Cuántas?
T_{ijk}	Verdadero si la casilla i de la cinta contiene el símbolo j en el paso k del cálculo.	$O(p(n)^2)$
H_{ik}	Verdadero si el cabezal de la máquina está posicionado en la casilla i en el paso k del cálculo.	$O(p(n)^2)$
Q_{qk}	Verdadero si M está en el estado q en el paso k del cálculo.	$O(p(n))$

Tabla 2 SAT para JSSP

3. Marco teórico

Se definen las expresiones booleanas "B" como la conjunción de las cláusulas de la siguiente tabla, para todo $-p(n) \leq i \leq p(n)$ y $0 \leq k \leq p(n)$:

Cláusulas	Condiciones	Interpretación	¿Cuántas?
T_{ij0}	La casilla *i* de la entrada contiene el símbolo *j* en la configuración inicial.	Contenido inicial de la cinta.	O($p(n)$)
Q_{s0}		Estado inicial de *M*	O(1)
H_{00}		Posición inicial del cabezal de la máquina.	O(1)
$T_{ijk} \rightarrow \neg T_{ij'k}$	$j \neq j'$	Un símbolo por cada celda de la cinta.	O($p(n)^2$)
$T_{ijk} = T_{ij(k+1)} \vee H_{ik}$		La cinta no cambia a menos que la transición escriba en la cinta.	O($p(n)^2$)
$Q_{qk} \rightarrow \neg Q_{q'k}$	$q \neq q'$	Sólo un estado en cada momento.	O($p(n)$)
$H_{ik} \rightarrow \neg H_{i'k}$	$i \neq i'$	Sólo una posición del cabezal en cada momento.	O($p(n)$)
La disyunción de las cláusulas $(H_{ik} \wedge Q_{qk} \wedge T_{i\sigma k}) \rightarrow (H_{(i+d)(k+1)} \wedge Q_{q'(k+1)} \wedge T_{i\sigma'(k+1)})$	$(q, \sigma, q', \sigma', d) \in \delta$	Posible transición en el paso *k* del cálculo cuando el cabezal está en la posición *i*.	O($p(n)^2$)
La disyunción de las cláusulas $Q_{f(p(n))}$	$f \in F$.	Debe terminar en un estado de aceptación.	O(1)

Tabla 3 Expresiones booleanas

Si la máquina acepta la entrada "*I*", es decir, si existe una sucesión de estados válidos para *M* con entrada *I*, entonces *B* es satisfacible, al asociar a las cláusulas T_{ijk}, H_{ik} y Q_{ik} sus correspondientes interpretaciones. Por otra parte, si *B* es satisfacible, entonces existe un

cálculo de la máquina *M* que partiendo de la entrada "*I*" conduce a un estado de aceptación que sigue los pasos indicados por la asignación de variables.

3.2.3 K-SAT

El problema de K-SAT ocupa uno de los lugares principales en los procedimientos de prueba de NP-Completitud, generalizada es la prueba para la evaluación del rendimiento de los algoritmos de búsqueda combinatoria, debido a la insolubilidad típica de muestras aleatorias generadas cerca de la criticidad. (Enrico Giunchiglia 2005)

El problema se define: dado un conjunto de n variables booleanas, denotados por B_k el conjunto de todas las k-cláusulas, es decir, las disyunciones de k literales diferentes y no complementarios de las n variables. La fórmula k-SAT aleatoria $F_k(n,m)$ se genera seleccionando aleatoriamente m cláusulas del conjunto B_k, separándolas por el operador lógico ∧. El problema k-SAT consiste en hallar una asignación de valores a las variables tal que todas las cláusulas de F_k (n, m) sean satisfechas.

3.2.4 Job Shop Scheduling como k-SAT

En 1994 Crawford y Baker propusieron una manera de plantear el Job Shop Scheduling como un problema SAT. Especificaron ciertas cláusulas que se detallan en la tabla que se presenta a continuación. Ellas son evaluadas como ciertas cuando todas sus respectivas restricciones están satisfechas. La primera restricción de precedencia indica una orden de ejecución para el par de tareas (i, j), expresa que *i* debe haber terminado antes de que *j* se puede iniciar. La segunda limitación es la restricción de la capacidad de los recursos, establece que no hay suficientes recursos (máquinas) para las operaciones (i, j) involucradas; también esta restricción establece que debe determinarse cuál de las operaciones (i,j) se ejecuta primero. La tercera limitación significa que para la operación i hay un tiempo de espera (r_i) que indica cuando la operación puede comenzar. Finalmente la cuarta restricción define la duración de cada operación en función de su fecha límite.

Restricciones Job Shop Scheduling		Codificación SAT	Significado SAT	Nº
Precedencia	$s_i + p_i \leq s_j$	$pr_{i,j} = T$	i precede a j	(1)
Capacidad de recursos	$(s_i + p_i \leq s_j) \vee (s_j + p_j \leq s_i)$	$pr_{i,j} \vee pr_{j,i} = T$	i precede a j o j precede a i	(2)
Tiempo de comienzo operación	$s_i \geq r_i$	$sa_{i,r_i} = T$	i inicia con r_i o después	(3)
Máximo tiempo de duración de la optimización	$s_i + p_i \leq d_i$	$eb_{i,d_i} = T$	Tiempo máximo de finalización d_i de la operación i	(4)

Tabla 4 JSSP restricciones para SAT

Crawford y Baker definen las condiciones de coherencia en la tabla que sigue. Si t es la hora de inicio reciente (LST) de cada operación ($t=r_i$), entonces los literales $sa_{i,t-1}$, $\sim eb_{i,t}$ y $\sim eb_{i,t+pi-1}$ son todas trivialmente ciertas. Entonces la cláusula con estos literales puede ser eliminada y entonces, la última cláusula ($\neg sa_{i,t} \vee \neg pr_{i,j} \vee sa_{j,t+p_i}$) es usada por la codificación RSF es la única cláusula que se necesita para definir la viabilidad de la solución. De esta manera la solución factible se obtiene mediante la representación 3-SAT en RSF.

Condiciones coherentes para JSSP en todos los tiempo relevantes del intervalo [r_i a d_i]	Significado
$\neg \boldsymbol{sa_{i,t}} \vee \boldsymbol{sa_{i,t-1}}$	No inicia la operación i hasta el tiempo t o después del tiempo t-1.
$\neg \boldsymbol{eb_{i,t}} \vee \boldsymbol{eb_{i,t+1}}$	La operación i no termina hasta el tiempo t o hasta que termine t+1
$\neg \boldsymbol{sa_{i,t}} \vee \neg \boldsymbol{eb_{i,t+p_{i-1}}}$	La operación i no inicia hasta el tiempo t o después de él, o no termina antes del tiempo $t+p_i$.
$\neg \boldsymbol{sa_{i,t}} \vee \neg \boldsymbol{pr_{i,j}} \vee \boldsymbol{sa_{j,t+p_i}}$	La operación i no inicia hasta t o después, o si no precede a j o si j inicia al tiempo $t+p_i$ o después.

Tabla 5 Coherencia en la transformación a SAT

3.2.5 Ejemplo codificación SAT de un Job Shop Scheduling:

Este ejemplo consta de dos máquinas (M_1, M_2) y 2 puestos de trabajo cada uno con 2 operaciones. Los recursos y la capacidad de restricciones de precedencia se muestran en la siguiente figura:

3. Marco teórico

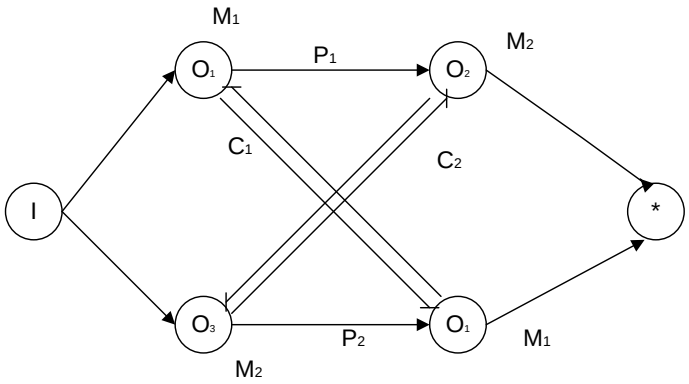


Ilustración 6 Planificación 2 trabajos, 2 máquinas

Las restricciones de precedencia son P_1 y P_2 y la capacidad de los recursos C_1 y C_2.

Las condiciones de operación:

Operación	Tiempo de procesamiento
1	2
2	6
3	5
4	4

Tabla 6 Condiciones de operación

Secuencia propuesta que conduce a solución verdadera:

Precedencia de actividades:

Actividad	Precedencia
O_1	I
O_2	O_1, O_3
O_3	I
O_4	O_1, O_3
*****	O_2, O_4

Tabla 7 Precedencia de actividades

Tiempos de inicio y procesamiento:

Operación	Tiempo de procesamiento p_i	Tiempo de inicio
1	2	0
2	6	5
3	5	0
4	5	5

Tabla 8 Tiempos de inicio y procesamiento

Evaluación SAT:

$$\left(\neg sa_{3,t} \vee \neg pr_{3,2} \vee sa_{2,t+p_3} \right) = \left(\neg sa_{3,0} \vee \neg pr_{3,2} \vee sa_{2,0+5} \right) = F \vee F \vee T = T$$

$$\left(\neg sa_{1,t} \vee \neg pr_{1,4} \vee sa_{4,t+p_1} \right) = \left(\neg sa_{1,0} \vee \neg pr_{1,4} \vee sa_{4,0+2} \right) = F \vee F \vee T = T$$

3. Marco teórico

El LST de la operación 3 es cero (ya que esta operación no tiene una operación precedente), entonces esta operación puede empezar en tiempo cero como es indicado por el tiempo de la tabla de tiempos de inicio y procesamiento, entonces $\neg sa_{3,t}$ es falsa. La segunda preposición $\neg pr_{3,2}$ es falsa porque la secuencia propuesta define que la operación 3 precede la operación 2. La última proposición $sa_{2,t+p3}$ es verdadera porque el tiempo de inicio de la operación 2 (igual a 5) es mayor o igual entonces el LST de la operación 3 más su tiempo de procesamiento.

3.3 Codificación del Job Shop Scheduling

El Job Shop Scheduling (JSSP) es el problema que permite al solucionarlo planificar un conjunto de N trabajos y M máquinas. Cada trabajo consiste de una serie de operaciones que se encuentran secuenciadas. Cada operación utiliza una sola máquina y tiene un tiempo de procesamiento correspondiente. Una vez determinado el tiempo de inicio de cada operación se programan todas las operaciones y se sabe la duración de todo el proceso (makespan). (“Genetic algorithms in engineering systems”, Takeshi Yamada y Ryohei Nakano, 1997).

El problema del Job Shop Scheduling no sólo pertenece a la clase NP-completa, sino que además es uno de los más difíciles de resolver de este tipo. Un indicador de ello son los problemas 10 x 10 formulados por Muth y Thompson que permanecieron sin solución por más de 20 años.

Pero luego surgieron distintos métodos para abordarlos e intentar darles solución, algunos algoritmos se basan en reglas de actividad y otros más sofisticados como los de cuello de botella que realizan aproximaciones estocásticas y simulaciones no lineales.

Otra posibilidad es plantearlos o codificarlos como problemas de satisfacción de restricciones considerando que en la formulación de estos problemas hay dos tipos de limitaciones: de capacidad y de precedencia como ya se describió en este trabajo.

En el punto 3.2.4 se hace una descripción de la representación del Job Shop Scheduling como k-SAT.

En el presente proyecto se hará uso de la codificación por orden que se describe en el siguiente punto.

3. Marco teórico

3.3.1 Codificación por orden (Order encoding)

El método de codificación (order encoding) es básicamente el mismo diseñado por Crawford y Baker para el Job Shop Scheduling en 1994 “Experiment al Results on the Application of Satisfiability Algorithms to Scheduling Problems”. Comparativamente $x \leq a$ es codificado por una variable Booleana diferente para cada variable entera x y para cada valor entero a.

El método es propuesto por Tamura y Taga y publicado en 2008 en el paper “Compiling finite linear CSP into SAT” en el marco de la competencia internacional de solucionadores SAT. El método propuesto transforma restricciones lineales enteras en problemas de satisfacción booleana (SAT) en CNF (forma normal conjuntiva).

El beneficio de esta codificación es la representación natural del orden de relación en los enteros. Los axiomas con dos literales como $\{\neg(x \leq a), x \leq a+1\}$ para cada entero a, representa el orden de relación para cada variable integral x.

Cuando se codifica de CSP a SAT, la fórmula de codificación no está más en forma general, cada variable se codifica a una fórmula CNF de comparación primitiva. Sin embargo, cuando se expanden las conjunciones para obtener una forma de cláusulas, el número de las cláusulas obtenido es la multiplicación de dos números de comparaciones primitivas.

Al introducir una nueva variable Booleana es posible reducir el tamaño, esta conversión no afecta la satisfactibilidad.

Antes de realizar la codificación de CSP a SAT es necesario determinar los límites inferiores y superiores de las variables enteras. Los autores proponen la siguiente fórmula para hacerlo, aunque existen diversas formas para obtener estos valores:

$$\ell = \max\left(\max_{0 \leq i < n} \sum_{0 \leq j < n} p_{ij},\ \max_{0 \leq j < n} \sum_{0 \leq i < n} p_{ij}\right)$$

$$u = \sum_{0 \leq k < n} \max_{(i-j) \bmod n = k} p_{ij}$$

Dónde:

- U es el límite superior de s_{ij} y m (makespan)
- L es el límite inferior de m (makespan), siendo 0 el límite inferior para s_{ij}.

3. Marco teórico

- N es el número de trabajos o de máquinas.

Luego de obtener estos valores se realiza una llamada al algoritmo propuesto por Tamura y Taga, y que es realizado por un programa desarrollado en Java denominado Sugar que realiza la conversión que se describió en el punto 3.2.4. Así se obtiene un número reducido de variables enteras manteniendo la satisfactibilidad del problema planteado.

3.4 Contexto matemático

3.4.1 Sistemas dinámicos de tiempo continuo

Los sistemas dinámicos han sido un importante punto de estudio para los científicos, incluyendo por ejemplo el análisis de los movimientos de los cuerpos dentro del sistema solar (que realizó Newton) y las leyes de Kepler (dieron origen al desarrollo de su cálculo). Así surgió el planteamiento de modelos de problemas dinámicos como ecuaciones diferenciales ordinarias.

A fines del siglo XIX Poincaré mostró que los métodos de perturbaciones podrían dar resultados correctos en todos los casos, porque las series usadas en los cálculos divergían. Así se fusionó el análisis con la geometría al desarrollar un punto de vista cualitativo para el estudio de ecuaciones diferenciales ordinarias, surgieron de este modo los Sistemas Dinámicos.

En el libro "Ingeniería de Control Moderna" de Ogata (2003), se define en términos matemáticos un sistema dinámico: partiendo de un campo vectorial en un espacio euclidiano $\mathbb{R}^n$, o bien de una función que va de $\mathbb{R}^n$ en sí mismo. Aquí se considerará en detalle el primer caso, siendo estos los sistemas dinámicos de tiempo continuo o diferenciables.

Considerando el campo vectorial X definido en $\mathbb{R}^n$ o en un abierto de $\mathbb{R}^n$ suficientemente diferenciable, digamos C^{∞}. A este campo vectorial $X: \mathbb{R}^n \longrightarrow \mathbb{R}^n$ se le puede asociar un sistema de ecuaciones diferenciales ordinarias que se pueden escribir vectorialmente en la forma: $\dot{x} = X(x)$, donde $x \in \mathbb{R}^n$ y el punto significa derivada con respecto al tiempo. La condición inicial $x_0 \in \mathbb{R}^n$ determina una solución, o sea una curva diferenciable $x = \varphi(t)$ en $\mathbb{R}^n$ tal que φ (0)=0 y su derivada satisface: $\frac{d\varphi}{dt}(t) = X(\varphi(t))$. Esta curva está definida en intervalo maximal $I \subset \mathbb{R}$ que contiene al 0. Se puede decir que este campo vectorial

3. Marco teórico

define un sistema dinámico de tiempo continuo, cuando se está interesado en el estudio global cualitativo de sus soluciones.

Las soluciones más simples son las de equilibrio $\varphi(t) = x_0$ constante, las cuales pueden encontrarse fácilmente mediante la condición de que el campo vectorial allí se anule, es decir $X(x_0)=0$.

A pesar de la posibilidad teórica del cálculo eficiente a través de los sistemas dinámicos que se demostró por distintos autores como: Siegelmann[13], la teoría de los sistemas dinámicos no lineales no ha sido explotada para problemas NP-Completos, a pesar del hecho de que tal como demuestran Gu y Du, en "On optimizing the satisfiability problem" (1999), también Nagamatu y Yanura en "On the stability of Lagrange programming networks for satisfiability problems of propositional calculus. Neurocomputing" (1996) y Wah en "Trace-based methods for solving nonlinear global optimization and satisfiability problems" (1997), k-SAT se puede formular como un problema de optimización global continuo e incluso lanzar como un sistema dinámico analógico.

3.4.2 Modelo matemático con restricciones

"Las limitaciones surgen de forma natural en la mayoría de las áreas del quehacer humano. Son el medio natural de expresión para la formalización de las regularidades que subyacen en el mundo computacional y físico y sus abstracciones matemáticas".

P. Van Hentenryck and V. Saraswat

Constraint Programming: Strategic Directions

La fórmula del problema de satisfactibilidad proposicional (SAT) es representada en forma normal conjuntiva con **m** cláusulas sobre **n** variables proposicionales, donde una cláusula es un subconjunto de literales positivos y negativos.

En el libro "Lógica matemática para Ingeniería de Sistemas y Computación" Cardona, Alonso y Jaramillo (2010), describen el modelado matemático:

[13] Computation beyond the Turing limit

Sean **P^2**, **N^2** dos símbolos relacionales que describen las ocurrencias positivas y negativas de los literales en las cláusulas: **P(x, y)** expresa que la variable x aparece positiva en la cláusula y. Respectivamente, **N(x, y)** expresa su ocurrencia negativa.

Por ejemplo (p V ¬ q V r) ^ (¬p V ¬r) ^ (¬pVq) es codificado como:

$\mathcal{A} = \langle |\mathcal{A}|, N^{\mathcal{A}}, P^{\mathcal{A}} \rangle$,

Donde: $|\mathcal{A}| = \{0,1,2\}, N^{\mathcal{A}} = \{(1,0),(0,1),(2,1),(0,2)\}$ y $P^{\mathcal{A}} = \{(0,0),(2,0),(1,2)\}$

La existencia de una asignación de valores de verdad a las variables que satisfaga una fórmula en forma normal conjuntiva (CNF) puede ser expresada con la sentencia $SO\exists\phi_{SAT}$:

$$(\exists T^1)(\forall y)(\exists x)[(P(x,y) \wedge T(x)) \vee (N(x,y) \wedge \neg T(x))]$$

"Existe una asignación de valores de verdad T tal que en toda cláusula hay al menos una variable que cumple una de las siguientes condiciones:

– la variable aparece positiva en la cláusula y está asignada a verdadero, i.e. T(x)

– la variable aparece negativa en la cláusula y está asignada a falso, i.e. ¬T(x)."

Una diferencia básica entre las restricciones de una programación matemática y las restricciones de un problema de satisfacción, es el hecho de que las restricciones en una formulación de programación matemática son típicamente lineales o no lineales, mientras que las limitaciones en una formulación de programación puede ser de una forma más general.

En el Job Shop Scheduling el problema es verificar que existe un calendario factible de trabajo con un makespan C_{max} que es menor que el valor dado z*.

La reducción del dominio significa que dado el horario parcial ya construido, cada operación aún no se ha programado y tiene una hora de inicio más temprana posible y un último tiempo posible de finalización. Cuando la hora de inicio y la de finalización de una operación son fijos, la comprobación se tiene que hacer sobre cómo afecta la operación recién programada (fija), los tiempos de inicio más tempranos y los plazos de ejecución de todas las operaciones que aún quedan por regular.

La satisfacción de restricciones requiere la especificación de un real makespan, lo que permite el procedimiento para la construcción de una agenda, así como ir hacia atrás en el tiempo.

3. Marco teórico

3.4.3 El caos determinista

"(...) Dentro de la compleja variedad de un sistema, realmente encontramos regularidades ocultas. Ese es el motivo de que, ahora, la del caos se haya convertido en una teoría muy amplia que se usa para estudiarlo todo, desde la bolsa hasta multitudes que producen tumultos, pasando por las ondas cerebrales durante la epilepsia. Cualquier sistema complejo en el que haya confusión y que sea imposible de predecir. Podemos hallar un orden subyacente (...). La teoría del caos dice dos cosas: primero, que los sistemas complejos, como el clima, tienen un orden subyacente. Segundo, la inversa de eso, que sistemas simples pueden producir un comportamiento complejo".

(Parque Jurásico, Crichton 1993)

El caos es una nueva y muy promisoria manera de aplicar las leyes conocidas de la física, con la ayuda fundamental de la computadora, a fenómenos muy variados que abarcan, además de los tradicionales en física, a los que se presentan en las ciencias biológicas y en las ciencias sociales, siempre y cuando se les pueda encarar como si se tratará de sistemas dinámicos complejos. ("Entre el orden y el caos. La complejidad", Sametband. 1999)

Los fenómenos de "caos determinista" o de "complejidad" se refieren a muchos sistemas que existen en la naturaleza cuyo comportamiento va cambiando con el transcurrir del tiempo (sistemas dinámicos). ("The Chaos Book" – Cvitanovic 2013) Dichos fenómenos aparecen cuando los sistemas se hacen extremadamente sensibles a sus condiciones esenciales de posición, velocidad, etc., de modo que alteraciones muy pequeñas en sus causas son capaces de producir grandes diferencias en los efectos. Como consecuencia de ello no es posible predecir con exactitud cómo se comportarán dichos sistemas más allá de cierto tiempo, por lo que parecen no seguir ninguna ley, cual si estuviesen regidos por el azar.

Pero los investigadores han encontrado que los sistemas dinámicos en estas condiciones presentan pautas de regularidad colectiva aunque no sea posible distinguir el comportamiento individual de cada uno de sus componentes.

Existe caos cuando en un sistema dos sucesos que empiezan en condiciones iniciales muy próximas evolucionan de manera diferente, de forma que se separan exponencialmente en el espacio de las fases. (Peter Smith 2001)

3. Marco teórico

El caos determinista comprende una serie de fenómenos encontrados en la teoría de sistemas dinámicos, la teoría de ecuaciones diferenciales y la mecánica clásica. En términos generales el caos determinista da lugar a trayectorias asociadas a la evolución temporal de forma muy irregular y aparentemente azarosa que sin embargo son totalmente deterministas, a diferencia del azar genuino. La irregularidad de las trayectorias está asociada a la imposibilidad práctica de predecir la evolución futura del sistema, aunque esta evolución sea totalmente determinista. (Von der Becke 2001)

Comprender el caos no es sencillo, no existe una única definición del caos pero hay tres puntos en los que todos los científicos están de acuerdo:

- Las trayectorias no se ajustan a un punto fijo, órbita periódica u órbita cuasiperiódica cuando $t \to \infty$.
- El sistema no contiene parámetros al azar. El comportamiento irregular surge de la no linealidad. Por eso se define como determinista.
- Las trayectorias que comienzan cerca, con el tiempo se separan exponencialmente.

Definir un atractor tampoco es fácil ya que todavía hay desacuerdos sobre la definición exacta. Un atractor es un conjunto en las que todas las trayectorias cercanas convergen (Lorenz, 1963). Los puntos fijos y círculos límite son un ejemplo de ello. Al igual que en la definición del caos, hay 3 acuerdos universales:

- Cualquier trayectoria que esté en un atractor, estará en él para $t \to \infty$.
- Atraen un conjunto de condiciones iniciales. El conjunto lo componen las condiciones iniciales que su trayectoria que acabe en el atractor cuando $t \to \infty$.
- No existen condiciones iniciales que satisfagan las dos reglas anteriores.

Dentro de los atractores se define como atractor extraño o caótico cuando el atractor exhibe dependencia sensible con las condiciones iniciales.

Existen atractores: de punto fijo (se estabiliza en un único punto), atractor de ciclo límite (se mantiene en un periodo igual para siempre) y el caótico (aparece en sistemas no lineales que tienen una gran sensibilidad a las condiciones). (Orden y caos en sistemas complejos, Ricard Solé y Susana Manrubia, 2001)

3. Marco teórico

3.4.4 El caos determinista para medir la dificultad de SAT

En un trabajo desarrollado en 2011 por Ercsey-Ravasz y Toroczkai ("Optimization hardness as transient chaos in an analog approach to constraint satisfaction"), publicado en la tapa de la revista Nature Physics, demostraron que más allá del umbral de densidad de restricciones, las trayectorias analógicas se vuelven transitoriamente caóticas, y los límites entre las cuencas de atracción de los grupos de solución serán fractales, señalando la aparición de la dureza de optimización. Argumentos y simulaciones analíticas indican que el sistema siempre encuentra soluciones para fórmulas satisfacibles incluso en los regímenes congelados de azar 3-SAT y de los problemas de ocupación bloqueados (considerado como uno de los puntos de referencia más difíciles algorítmicos), la propiedad se debe en parte a los sistemas de carácter hiperbólico. El sistema encuentra soluciones en tiempo polinómico continuo, pero a expensas de las fluctuaciones exponenciales en su función de energía.

Lo que publicaron estos investigadores es un sistema dinámico de tiempo continuo para resolver k-SAT, con una dinámica diferente a la que proponen enfoques anteriores.

A continuación se presentan las variables continuas que lo conforman: $s_i \in [-1,1]$, de manera que $s_i = -1$ si la i-ésima variable (x_i) es falsa y s_i=1 es verdadera. Definición de restricción $K_m(s)=2^{-k}\ \Pi^N{}_{i=1}(1 - c_{mi}\, s_i)$ correspondiente a la clásula m, se tiene que $K_m \in [0,1]$ y K_m =0 si y sólo si m es cláusula satisfecha. El objetivo sería encontrar una solución s* con $s_i^* \in \{-1,1\}$ a E(s*)=0, dónde E es la función energía $E(s)=\sum_{m=1}^{M} K_m(s)^2$. Si s* existe, será un mínimo global para E y una solución para el problema k-SAT. Sin embargo, encontrar s* por una minimización directa de E(s) será típicamente incorrecta debido a que los atractores[14] no solucionan la captura dinámica de búsqueda. Para evitar esto definen una función de energía modificada $V(s,a) = \sum_{m=1}^{M} a_m K_m(s)^2$, usando variables auxiliares $a_m \in (0,\infty)$ similar a los multiplicadores de Lagrange. Denotando $\mathcal{H}_N$ el dominio continuo $[-1,1]^N$. Su límite es el N-hipercubo $Q_N=\delta\mathcal{H}_N$ con el conjunto de vértices: $\mathcal{V}_N = \{-1,1\}^N \subset Q_N$. El conjunto de soluciones para una determinada fórmula k-SAT, llamado espacio de soluciones, ocupa un subconjunto de V_N. Los grupos de solución están formados por soluciones que se pueden conectar a través de un solo cambio de variable, permaneciendo siempre dentro de las asignaciones satisfactorias. Claramente $V \geq 0$ en $\Omega \equiv \mathcal{H}_N \times (0,\infty)^M$ y V(s,a)=0 con $\mathcal{V}_N$ si y sólo sí s=s* $\in \mathcal{V}_N$ es una solución k-SAT, para cada $a \in (0,\infty)^M$. A continuación se introduce el sistema dinámico de tiempo continuo en Ω:

[14] Atractor: conjunto al que el sistema evoluciona después de un tiempo suficientemente largo

$$\frac{ds_i}{dt} = (-\nabla_s V(s,a))_i = \sum_{m=1}^{M} 2a_m c_{mi} K_{mi}(s), \quad i = 1,\ldots,N \qquad (1)$$

$$\frac{da_m}{dt} = a_m K_m(s), \quad m = 1,\ldots,M \qquad (2)$$

Donde: ∇_s es el operador gradiente con respecto a s, y $K_{mi}=K_m/(1-c_{mi}s_i)$. Las condiciones iniciales para s son arbitrarias $s(0) \in \mathcal{H}_N$, sin embargo para *a* tiene que ser estrictamente positiva, $a_m(0) \geq 0$. La solución k-SAT $s^* \in \mathcal{V}_N$ son puntos fijos, para culaquier $a \in (0,\infty)^M$.

Los conjuntos de solución k-SAT son conjuntos que abarcan trozos compactos, conjuntos conectados en Q_N y todos los puntos en el son un punto ajustado. El sistema tiene una serie de propiedades claves. La dinámica en s permanece confinada a $\mathcal{H}_N$. Las soluciones k-SAT $s^* \in \mathcal{V}_N$ son puntos ajustados atractivos. En particular, cada punto *s* desde la ortante de una solución k-SAT s* con la propiedad $|s|^2 \geq N - 1 + (k-1)^2/(k+1)^2$ está garantizada para fluir en el atractor correspondiente a s*. No existen ciclos limitados. Para satisfacer fórmulas los atractores de punto único fijos de la dinámica son los mínimos globales de V, con V=0. Teniendo en cuenta que la proyección de la dinámica en $\mathcal{H}_N$ podría ser atrapado en algún punto $\bar{s}$, mientras $da/dt \neq 0$ indefinidamente. Aquí no sucede, por otra parte, los argumentos analíticos compatibles con simulaciones indican que la trayectoria dejará cualquier dominio que no contiene soluciones. Teniendo en cuenta, que las funciones de restricción (de ahí su satisfactibilidad) dependen directamente sólo en la ubicación de la trayectoria en $\mathcal{H}_N, K_m = K_m(s)$, y no en las variables auxiliares. La dinámica en el a-espacio es una expansión simple, y es por esta razón las característica de toda la fase en el espacio Ω se encuentran dentro de su proyección sobre $\mathcal{H}_N$. Se pueden eliminar por completo las variables auxiliares de las ecuaciones primero resolviendo para dar $a_m(t) = a_m(0) \exp\left(\int_0^t K_m\big(s(\tau)\big)d\tau\right)$, y a continuación insertarla.

Otra característica fundamental es que es determinista: para una fórmula dada *f*, cualquier condición inicial genera una trayectoria única, y cualquier conjunto de $\mathcal{H}_N$ tiene una imagen inversa única arbitraria en el tiempo. Por lo tanto, las características del espacio de la solución se reflejan en las propiedades de los conjuntos invariantes de la dinámica dentro del hipercubo[15] $\mathcal{H}_N$. La naturaleza determinista permite definir cuencas de interés de las agrupaciones solución coloreando cada punto en $\mathcal{H}_N$, según la cual el grupo de trayectoria

[15] Hipercubo: cubo desfasado en el tiempo, es decir, cada instante de tiempo por el cual se movió pero todos ellos juntos.

3. Marco teórico

fluye, si se inicia a partir de ahí. Estas cuencas llenan $\mathcal{H}_N$ hasta un conjunto de medida cero (de Lebesgue), que forma el límite de la cuenca, desde donde la dinámica (por definición) no puede fluir a cualquiera de los atractores. Una fórmula k-SAT f se puede representar como un hipergrafo[16] $\mathcal{G}(f)$ (o de manera equivalente, un factor de gráfico) en la que los nodos son variables y los "hiperedges[17]" son cláusulas que conectan los nodos/variables en la cláusula con literales puras (las que participan en una o más cláusulas, pero siempre en la misma forma), por lo que siempre se puede elegir, como para satisfacer esas cláusulas. El núcleo de $\mathcal{G}(f)$ es el subgrafo que queda después de retirar secuencialmente todos las hyperedges que tienen literales puras. Para las fórmulas simples (tales como los que no tienen un núcleo), la dinámica es de flujo laminar y los límites de las cuencas forman conjuntos lisos y no fractales. La adición de más restricciones $\mathcal{G}(f)$ se desarrolla un núcleo, las ecuaciones de giro se acoplan mutuamente, las trayectorias pueden llegar a ser caóticas y los límites de las cuencas fractales. Por lo tanto, como la densidad de restricción $\alpha = M/N$ se incrementa dentro de conjuntos predefinidos de fórmulas (k-SAT aleatorios, los problemas de ocupación, k-XORSAT, etc) se espera un cambio brusco de comportamiento caótico, que en un punto de transición caótica α_x, donde un núcleo caótico aparece con un pesos estadístico distinto a cero en el conjunto $N \longrightarrow \infty$. Como ejemplo, se considera 3-XORSAT, en este caso debido a su naturaleza intrínsecamente lineal, es mejor trabajar directamente con las ecuaciones de verificación de paridad como limitaciones, en lugar de su forma CNF. El núcleo caótico aquí es un pequeño finito hipergráfico y entonces α_x coincide con el llamado punto de transición dinámica calculado por Mézard. Un núcleo puede ser no-caótico, y por lo tanto la existencia de un núcleo es sólo una condición necesaria para la aparición de caos, y en general las dos transiciones pueden no coincidir. Al aumentar aún más el número de restricciones (dentro de cualquier conjunto de fórmulas) aparece la no satisfactibilidad en el umbral $\alpha_s > \alpha_x$ más allá del cual casi todas las fórmulas son insatisfactibles (régimen UNSAT).

Cuánto más cerca α es a α_s, más difícil es encontrar soluciones, y más allá del llamado punto de transición de congelación $\alpha_f < \alpha_s$ (llamado el régimen congelado) todos los algoritmos conocidos toman tiempos exponencialmente largos o simplemente no encuentran so-

[16] Hipergrafo: generalización de un gráfico en el que un extremo se puede conectar cualquier número de vértices. Es un par H=(X,E) dónde X es un conjunto de elementos llamados *nodos* o *vértices,* y E es un conjunto de subconjuntos no vacíos de los X llamado hyperedges o bordes.

[17] Hyperedges: son conjuntos arbitrarios de nodos, y por lo tanto pueden contener un número arbitrario de nodos.

lución. Una variable se congela si toma el mismo valor para todas las soluciones dentro de un grupo, y un grupo se congela si se congelan un gran número de sus variables. En el régimen congelado todos los grupos se congelan y también están muy separados ($\mathcal{O}(N)$ distancia de Hamming). Para un 3-SAT aleatorio (las cláusulas son elegidas uniformemente al azar fijando α) $\alpha_s \cong 4{,}26$, $\alpha_f \cong 4{,}25$ y todos los algoritmos de búsqueda conocidos locales se convierten a exponencial no más allá de α=4,21, mientras que los algoritmos basados en la propagación de la encuesta no más allá de α=4,25. Dado que el régimen congelado es muy delgado en 3-SAT aleatorio, se han introducido los llamados problemas de ocupación bloqueados (LOPs). En LOPs todos los grupos están formados por exactamente una solución, por lo tanto, están completamente congelados y el régimen congelado se extiende desde la agrupación (dinámica) del punto de transición l_d al umbral de satisfactibilidad l_s, y por lo tanto es muy amplia.

Como el caos está presente para satisfacer fórmulas, es decir, cuando el sistema ha fijado los puntos atraidos, es necesariamente de tipo transitorio. El caos transitorio se ubica en los sistemas con muchos grados de libertad, tales como una turbulencia del fluido. Aparece como el resultado de las intersecciones homoclínicas[18] / heteroclínicas[19] de los colectores invariantes de hiperbólicas de puntos fijos (inestable) de la falsedad dentro de los límites de la cuenca, que conduce a foliaciones complejos (fractal) del espacio de fases. Se ha observado la prevalencia de caos transitorio en toda la región $\alpha_x<\alpha<\alpha_s$ para todas las clases de problemas que fueron estudiados. Las fluctuaciones de la velocidad de las trayectorias en el régimen caótico son cualitativamente similares a los de las parcelas de fluido en los flujos turbulentos. Los resultados obtenidos por Ercsey-Ravasz y Torockzai en su inversitigación sugieren que el comportamiento caótico puede ser una función genérica de algoritmos de búsqueda de soluciones en problemas de optimización difíciles, lo que corrobora las observaciones de Elser utilizando un algoritmo heurístico basado en mapas iterados.

[18] Homoclínica: trayectoria que inicia en punto de equilibrio del tipo punto de silla termina en el mismo punto de equilibrio. . Una órbita V_0 empezando en un punto $x \in \mathbb{R}^n$ es llamada homoclínica al punto de equilibrio x_0 del sistem si $X_t(x) \to x_0$ cuando $t \to \pm\infty$.

[19] Heteroclínica: una órbita V_0 empezando en un punto $x \in \mathbb{R}^n$ es llamada heteroclínica a los puntos de equilibrio $x_{(1)}$ y $x_{(2)}$ del sistema si $X_t(x) \to x_{(1)}$ cuando $t \to -\infty$ y $X_t(x) \to x_{(2)}$ cuando $t \to +\infty$.

3. Marco teórico

3.5 CTDS para resolver Sudokus

En el siglo XVIII Euler creó un sistema de probabilidades para representar una serie de números sin repetir, por eso se lo considera el inventor del juego Sudoku, en 1970 se publica en una revista este pasatiempos, pero es en 1984 cuando se lo llama *Sūji wa dokushin ni kagiru* (数字は独身に限る) "los números deben estar solos", f*ue Kaji Maki, presidente de Nikoli, quien le puso el nombre abreviado (Sudoku).*

Un sudoku consiste en una cuadrícula de 9×9 cuadrados o celdas. El problema está en intentar rellenar la cuadrícula con los números del 1 al 9 de forma que en cada fila, en cada columna y en cada subcuadrícula de 3×3 celdas, no se repita nunca el mismo número. Para cada problema del sudoku correctamente formulado, existe una única solución.

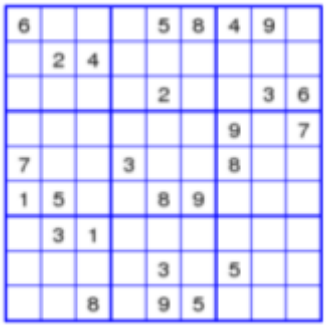


Ilustración 7 Ejemplo sudoku

No todos los sudokus tienen la misma dificultad, generalmente depende de la cantidad de números que aparecen en el sudoku antes de comenzarlo y de la colocación de los mismos. Lo que sí es una norma es que **todo sudoku bien planteado debe tener solución única.** (“Sudoku as a SAT Problem”, Inés Lynce y Joel Ouaknine, 2006)

La estructura matemática de los rompecabezas de Sudoku es similar a los problemas de satisfacción de restricciones duras típicas que se encuentran en el corazón de muchas aplicaciones, incluyendo la planificación de la producción o Job Shop Scheduling.

Dado que hay pocas esperanzas en la prestación de algoritmos de tiempo polinomial para los problemas NP-completos (Sudoku, Job Shop Scheduling), la atención se desplazó hacia la comprensión de la naturaleza de la complejidad que prohíbe soluciones rápidas a estos problemas. Ha habido un considerable trabajo en esta dirección, en especial para el problema satisfactibilidad booleano k-SAT, que es NP-completo para k> = 3. Debido a la exhaustividad, todos los problemas en NP, pueden ser traducidos (en tiempo polinómico) y formularse como un problema k-SAT.

3. Marco teórico

En k-SAT se dan N variables booleanas a los que hay que asignar 0s o 1s (VERDADERO o FALSO) de tal manera que un determinado conjunto de cláusulas en forma normal conjuntiva estén todos satisfechos. Aquí también hay muchas maneras exponenciales 2^N para las configuraciones o asignaciones de búsqueda.

En este trabajo se hace uso del algoritmo desarrollado por Ercsey-Ravasz y Toroczkai en 2011, el mismo trata con un sistema dinámico. Un algoritmo es un conjunto finito de instrucciones que actúan en un espacio de estado, se aplica iterativamente a partir de un estado inicial hasta que se alcanza un estado final.

El algoritmo propuesto es del tipo determinista es decir, una vez que se da un estado inicial, la trayectoria del sistema dinámico se determina de forma exclusiva ("Determinismo, Caos e impredictibilidad", Campos, 2002). Por lo tanto, se espera que la dinámica de los algoritmos que explotan la estructura de los problemas difíciles se refleje la complejidad inherente al problema en sí. El comportamiento complejo de los sistemas dinámicos deterministas se acuñó como "el caos" en la literatura, y por lo tanto se espera que el comportamiento de los algoritmos para problemas difíciles sea muy irregular o caótico.

Aunque la teoría de los sistemas dinámicos no lineales y el caos están bien establecidos, aún no ha sido explotada en el contexto de los algoritmos de optimización. Una de las dificultades recae en el hecho de que la mayoría de los algoritmos de optimización son discretos, pero es por ello que no es complejo aplicar los métodos de la teoría del caos. Los investigadores propusieron en su paper "Optimization hardness as transient chaos in analog approach to constraint satisfaction", un solucionador de tiempo continuo determinista para el problema satisfactibilidad booleana k-SAT junto con ecuaciones diferenciales ordinarias (ODE) con una relación uno a uno entre los grupos de solución de k-SAT y los atractores del correspondiente sistema de ecuaciones diferenciales ordinarias. Este sistema dinámico de tiempo continuo (CTDS) se encuentra en forma natural adecuada para métodos de la teoría del caos, y por lo tanto permite estudiar la relación entre la optimización de la dureza y el comportamiento caótico. La investigación se centra en los casos satisfacibles (tienen solución) y por lo tanto el comportamiento caótico observado será necesariamente transitorio. Hay que destacar, sin embargo, que las propiedades dinámicas caracterizan tanto el problema como el propio algoritmo. Por esta razón, se comparan las propiedades dinámicas a través de los problemas de dureza variable utilizando el mismo algoritmo. Sin embargo, ya que hay instancias de problemas que son difíciles para todos los algoritmos

3. Marco teórico

conocidos, la aparición de caos transitorio debe ser una característica universal de los problemas difíciles.

Es importante observar que el caos transitorio no es un comportamiento asintótico $N \rightarrow \infty$, pero se parece a N finito, y por lo tanto las medidas de caos se pueden usar para caracterizar y clasificar la dureza de los casos individuales de problemas finitos.

Ercsey-Ravasz y Toroczkai además de publicar el solucionador para sudokus planteados como k-SAT, realizaron posteriormente un análisis del comportamiento de las trayectorias correspondientes al CTDS (Sistema Dinámico de Tiempo Continuo) y allí quedó demostrada la aparición de caos transitorio cuando aumenta la dureza de los problemas, y se muestra que nivel de dureza se correlaciona bien con una caótica invariante, siendo el tiempo de vida del caos: k^{-1}, donde k es la llamada frecuencia de escape.

3.5.1 Conversión sudoku a SAT (en forma CNF)

En la lógica booleana CNF es una representación de un conjunto de cláusulas en forma normal conjuntiva, siendo una cláusula una disyunción de literales. Todas las conjunciones de literales y todas las disyunciones de literales están en CNF, ya que se pueden considerar como conjunciones de las cláusulas de un literal y conjunciones de una sola cláusula, respectivamente. (“Sudoku un problema NP completo”, Valverde, 2011)

Un problema SAT se representa mediante n variables proposicionales x_1, x_2,…, x_n, a las que se pueden asignar valores de verdad: 0 (falso) o 1 (verdadero). Un *l* literal es ya sea una variable x_i (es decir, un literal positivo) o su complemento: $\neg x_i$ (es decir, un literal negativo). Una cláusula ω es una disyunción de literales y una fórmula CNF ϕ es una conjunción de cláusulas. Un l_j literal de una cláusula de ω_a al que se le asigna un valor de verdad 1 y la cláusula será satisfecha. Si se asigna el valor de verdad: 0 literal, a continuación se elimina de la cláusula. Una cláusula de un solo literal se dice que es la unidad y su literal tiene que ser el valor 1 para la cláusula a ser satisfecha. La derivación de una cláusula vacía indica que la fórmula es insatisfactoria para la tarea encomendada. La fórmula es satisfecha si todas sus cláusulas son satisfechas.

El problema SAT consiste en decidir si existe una asignación de verdad para las variables de tal manera que la fórmula se convierte en satisfactoria. Un rompecabezas de Sudoku puede ser fácilmente representado como un problema SAT, aunque requiere un número

significativo de variables proposicionales. Si se estuvieran usando las variables con dominios arbitrarios, entonces las variables 9 x 9 con dominio sería la opción más adecuada. Sin embargo, codificar un sudoku en CNF requiere 9 x 9 x 9=729 variables proposicionales. Para cada entrada de la grilla se asocian 9 variables. Las limitaciones se añaden a la codificación SAT para referirse a cada entrada, cada fila, cada columna y cada sub-cuadrícula de 3x3.

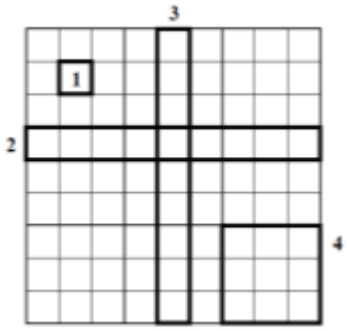


Ilustración 8 Restricciones del Sudoku

Resumiendo, teniendo un tablero de tamaño n x n:

- Cada digito del 1 a n debe ir colocado en cada fila y columna sin repetirse, al igual que en cada sugrilla $\sqrt{n} \times \sqrt{n}$.
- Hay números que ya se encuentran ingresados en el rompecabezas.
- Existe una única solución.

Hay dos etapas en la traducción de un rompecabezas de Sudoku en un problema SAT. Durante la primera etapa, el problema SAT es equivalente a una rejilla de Sudoku vacía. En primer lugar, se deben identificar todas las variables booleanas, y luego formar las cláusulas de acuerdo a cada regla. Durante la segunda etapa, las células parcialmente llenas de sudoku se incorporarán en el problema SAT como cláusulas que contienen una sola variable.

Como se mencionó anteriormente habrá un total de 9x9x9 = 729 variables. Las variables existen como una matriz de enteros 9x9x9, denominada Var, donde Var (i, j, k) indica si el número k ocupa la celda (i, j). Para simplificarlo, Var (i, j, k) será identificado como un entero o por el valor i x 100 + j x 10 + k. Una variable que aparece en una cláusula como un entero positivo representa un literal positivo asociado a esa variable, y viceversa. Por

3. Marco teórico

ejemplo, "129" indica el número 9 aparece en la columna 2 de la fila 1, mientras que "-129" indica que el número 9 no aparece en la fila 1, columna 2.

Las cláusulas también están representadas por matrices de enteros. Para cada columna, fila y sub-grilla, cada número del 1 al 9 puede aparecer exactamente una vez. Por lo tanto, dos tipos de cláusulas son necesarias para cada columna, fila y sub-grilla: estas son las cláusulas At-Least (al menos) y las At-Most (a lo sumo). Una cláusula At-Least se compone de 9 variables como literales positivos, mientras que una At-Most se compone de sólo 2 variables ambas como literales negativas. La combinación de ambos tipos de cláusulas obliga a la propiedad "una sola vez" para ser verdad.

Para codificar un problema 9x9, existirán:

- Restricciones D_{ij} que impondrán la singularidad en una celda dada.
- Restricciones impuestas a las filas, columnas y subgrillas en cada capa.
- Restricciones dadas por las pistas que se encuentran en el tablero.

De este modo las primeras brindarán 81 restricciones: (x^1_{ij}, x^2_{ij}, x^3_{ij}, x^4_{ij}, x^5_{ij}, x^6_{ij}, x^7_{ij}, x^8_{ij}, x^9_{ij}). Las segundas proveen 9 x 27, dadas por:

Filas:

$$(x^a{}_{i1}, x^a{}_{i2}, \dots, x^a{}_{i9}), \;\; i = 1,..,9$$

Columnas:

$$\left(x^a{}_{1j}, x^a{}_{2j}, \dots, x^a{}_{9j}\right), \;\; j = 1,..,9$$

Subgrilla:

$$(x^a{}_{m+1,n+1}, x^a{}_{m+1,n+2}, x^a{}_{m+1,n+3},$$
$$x^a{}_{m+2,n+1}, x^a{}_{m+2,n+2}, x^a{}_{m+2,n+3},$$
$$x^a{}_{m+3,n+1}, x^a{}_{m+3,n+2}, x^a{}_{m+1,n+3})$$
$$m, n = 0,3,6$$

En total entonces, hay ahora 324 restricciones. Las ultimas vienen dadas por las pistas d, está demostrado que para que tenga solución única deben ser al menos 17, entonces d≥ 17. Cada pista dada, elimina 4 restricciones, en la torre vertical, la columna, la fila y la subgrilla que contienen la pista. (“The chaos within Sudoku”, 2012)

3. Marco teórico

Por ejemplo dado el siguiente sudoku

Ilustración 9 - Ejemplo sudoku

Para determinar la capa 4 (L_4):

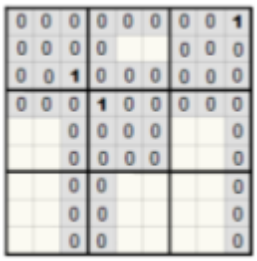


Ilustración 10 - Capa 4

Hay 3 pistas en las celdas (1,9), (3,3), (4,4) y por lo tanto: $x^4_{1,9}=1$, $x^4_{3,3}=1$, $x^4_{4,4}=1$ se fijan como verdaderas, y el resto de las variables son falsas. Las variables desconocidas se encuentran en las celdas claras. Las otras pistas eliminarán restricciones y variables en otras capas y columnas. El número total de variables desconocidas N depende de *d* y del lugar donde se encuentran las pistas. El número de restricciones es siempre 324-4d, sin embargo el número de variables en una clausula puede variar. En este caso la segunda fila en la capa 4, tiene solo 2 variables desconocidas.

Luego de que las variables booleanas desconocidas y las limitaciones se han identificado hay que transformar la fórmula a CNF. Para esto existen varias maneras.

Un 1-k-SAT definido sobre las variables $(y_1,y_2,\ldots,y_k)$puede ser escrita como un k-SAT y k(k-1)/2 de 2-SAT restricciones:

$$(y_1 \vee y_2 \vee \ldots \vee y_k) \wedge \left[\bigwedge_{i<j} (\bar{y}_i \vee \bar{y}_j)\right]$$

3. Marco teórico

La disyunción (∨) de la primera variable k hace cumplir que al menos una variable debe ser cierta, pero el resto de (k(k-1)/2) 2-SAT tipo de restricciones asegura que solo una de ellas debe ser verdadera.

Cada celda debe tener exactamente un valor:

$$E_d = \bigwedge_{y=1}^{XY} \bigwedge_{x=1}^{XY} \left[\bigvee_{v=1}^{XY} (x, y, v) \right]$$

No puede ocurrir un valor dos veces:

$$E_u = \bigwedge_{y=1}^{XY} \bigwedge_{x=1}^{XY} \bigwedge_{v_1=1}^{XY-1} \bigwedge_{v_2=v_1+1}^{XY} [\neg(x, y, v_1) \vee \neg(x, y, v_2)]$$

Para todas las fórmulas predefinidas [x,y]=v, agregar una cláusula: Asignado={[x,y,v]|celda [x,y] tiene el valor v}

Hay diferentes formas de codificar en CNF las cláusulas:

$F_{mínima}=C_d \wedge R_u \wedge C_u \wedge B_u$ Asignado

Resumiendo:

Dada la grilla del Sudoku y las reglas de cada celda se procede a la codificación en CNF, se usan 4 variables booleanas para cada celda (9x9x4=324 variables booleanas). Luego se tienen las siguientes reglas: cada celda debe tener un valor del 1 al 9, todas las celdas de una columna deben tener valores diferentes, todas las celdas de una fila deben tener valores diferentes, todas las celdas del cuadrado (subgrilla) deben tener diferentes valores.

Para codificar la primera regla hay que especificar que el valor de una celda debe ser diferente a las otras, entonces se utiliza una notación que determine que el valor v_1 es diferente a x_1, de esta manera si x_1 es igual a 0 entonces se quiere que v_1 sea 1.

Para la segunda regla se requiere que sean todas diferentes, entonces pueden tomarse de a pares, lo que se necesita es que x sea diferente a y (X!=Y), esto se realiza tomando los 4 bits, entonces se tiene x_1, x_2, x_3, x_4 y y_1, y_2, y_3, y_4

x_1 != y_1 o x_2 != y_2 o x_3 != y_3 o x_4 != y_4

x_1 != y_1 significa(x_1 Y !y_1) O (!x_1 Y y_1) es verdadero

3. Marco teórico

En la forma CNF, se aplica esta regla:

(A y B) o C = (A o C) y (B o C)

Entonces: x_1 != y_1 significa (x_1 O y_1) Y (!x_1 O !y_1)

Se hace lo mismo para todas las variables obteniendo 16 cláusulas.

Luego se puede utilizar este mismo método para las reglas que quedan (3 y 4), eligiendo las celdas correspondientes.

3.5.2 Solucionador Sudoku

Los investigadores Ercsey Ravasz y Toroczkai de las Universidades Babes Bolyai (Rumania) y Notre Dame (EEUU) propusieron un solucionador para Sudokus, que luego de codificarse a forma normal conjuntiva es evaluado por un sistema dinámico de tiempo continuo hasta hallar la solución. Además realizaron un análisis del tiempo de resolución e identificaron una medida de la dificultad de resolución del problema, que es inherente al caos implicado en su solución, los autores lo denominaron "escala Ritcher de Sudoku[20]" debido a que mide la energía necesaria para lograr el estado deseado.

La investigación se basó en el planteo del Sudoku como un problema k-SAT (la planificación del trabajo, también puede serlo), y su análisis mediante dinámicas continuas no lineales.

La escala que desarrollaron para medir la dificultad permite una determinación exacta de la dureza de un rompecabezas Sudoku. Lo que se debe realizar para determinarlo es calcular su η:

$$\eta = - log_{10}(\kappa)$$

Recordando que κ es el ratio de escape del caos, una cantidad fácilmente medible, que se puede expresar teóricamente como un cero espectral del determinante del operador de evolución correspondiente a el sistema dinámico. Este es un coeficiente que mide la dureza del problema, posteriormente se compara con la escala de valores para obtener el nivel al que corresponde:

- Un rompecabezas simple (easy) está en el rango $0 < \eta \leq 1$

[20] Escala Richter: escala sismológica logarítmica arbitraria que asigna un número para cuantificar la energía liberada por un terremoto.

- Los medianos (medium) de $1 < \eta \leq 2$

- Los difíciles (hard) de $2 < \eta \leq 3$

- Los rompecabezas ultra-difíciles (ultra-hard) son los $\eta > 3$

En la ilustración 9 (a) se presenta un sudoku sencillo y a su derecha la evolución de la dinámica de tiempo continua en una de las subgrillas. En (b) se muestra lo mismo pero para el rompecabezas muy difícil denominado "Platinum Blond".

Un rompecabezas con una calificación de 2 lleva, en promedio 10 veces más tiempo para resolver que una con calificación de 1. El Sudoku más complejo encontrado actualmente es el denominado "Platinium Blond" (figura 9 b) su $\eta = 3{,}5789$ es cercano a 3,6.

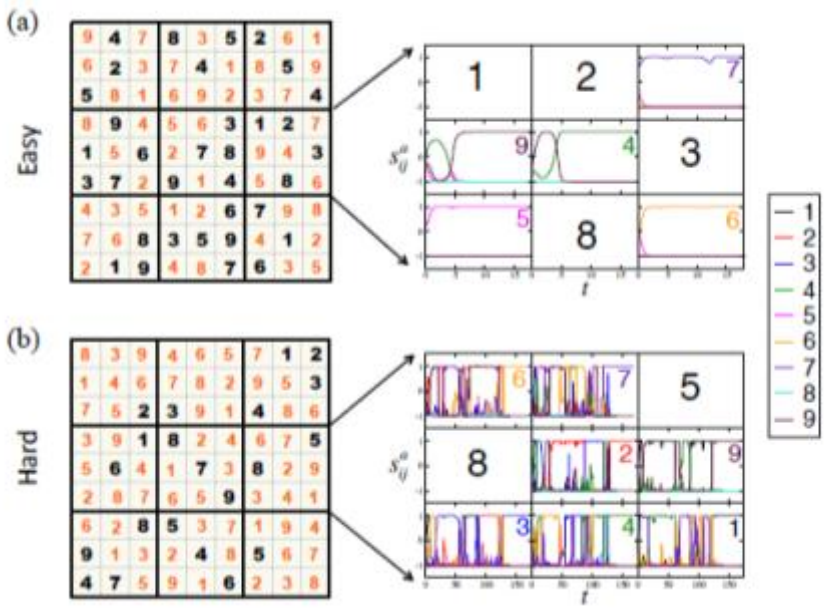


Ilustración 11 - Sudokus resueltos con CTCD

Retomando lo expuesto en puntos anteriores de esta investigación el 1-k-SAT es sencillo pero si k>2, y dado que N se hace más grande el problema empieza a hacerse dificultoso, convirtiéndose al tipo NP, es decir que para encontrar la solución tarda más que el tiempo polinómico.

La conversión a k-SAT del Sudoku como ya se especificó en el punto anterior no es sencilla, dado que un rompecabezas fácil con 34 pistas, tendrá 126 variables y 717 cláusulas en la expresión booleana (CNF). Pero afortunadamente existen algoritmos que realizan esta conversión en forma automática.

3. Marco teórico

Luego de la conversión los autores plantean la aplicación de la función energía que indica que tan lejos se está de la solución con la configuración propuesta. La función energía es cero cuando la configuración es la solución. Esta función de energía es una función continua de las variables de solución que oscilará entre -1 y 1. Esto convierte el problema de una búsqueda discreta a una continua. Lo que sobresale de esta investigación es la conexión entre la búsqueda discreta y la dinámica no lineal continua.

Una vez realizada la transformación a CNF deben quedar N variables y M cláusulas SAT, el número de variables que aparece en las restricciones *m* se denota k_m, m=1,…,M (claramente $1 \leq k_m \leq 9$). Los parámetros son N, M y $\{k_m\}^M_{m=1}$, todo depende de las pistas y de la dificultad de expresarlo analíticamente, pero es sencillo de determinar computacionalmente.

En la publicación de Ercsey-Ravasz y Toroczkai "Optimization hardness as transient chaos in an analog approach to constraint satisfaction" (2011), introdujeron un solucionador para problemas k-SAT presentados en la forma CNF, el conjunto de cláusulas que especifican las restricciones se traduce a una matriz M x N llamada $C=\{c_{mi}\}$ con:

- c_{mi}=1 si la variable x_i es presente en la clausula *m* en directo (forma no negada)
- c_{mi}=- 1 si $\bar{x}_i$ no está presente en la cláusula *m* (forma negada)
- c_{mi}=0 si x_i y $\overline{x_i}$ son ambos ausentes en C_m

Para todas las variables x_i se asocia una variable de rotación continua ("spin") $s_i \in$ [-1,1] de tal manera que cuando $s_i = \pm 1$ entonces $x_1=(1+s_i)/2 \in \{0,1\}$, y para toda clausula C_m se asocia la función:

$$K_m(s) = 2^{-k_m} \prod_{j=1}^{N} (1 - c_{mj}\, s_j), m \in \{1, \dots, M\} \quad (1)$$

Entonces se tiene que $K_m \in [0,1]$ para todo $s \in [-1,1]^N$. Es sencillo verificar que K_m=0 sólo para aquellos $s_i \in \{-1, +1\}$ valores para las correspondientes x_i-s clausulas a satisfacer C_m (de otra manera siempre se tiene K_m>0). Esto es K_m juega el rol de la función de energía para la cláusula C_m y el valor de estado fundamental de K_m=0 se alcanza si C_m es verdadero.

También se necesitan las cantidades $K_{mi}=K_m/(1-c_{mi}s_i)$ esto es, con el i-ésimo término que falta del producto en (1). Claramente, $K_{mi} \in [0,1/2]$. El sistema dinámico de tiempo continuo introducido se define a través del conjunto de (N + M) ecuaciones diferenciales ordinarias (ODE):

3. Marco teórico

$$\frac{ds_i}{dt} = \sum_{m=1}^{M} 2a_m c_{mi} K_{mi}(s) K_m(s), \quad i = 1 \dots, N \quad (2)$$

$$\frac{da_m}{dt} = a_m K_m(s), \quad m = 1, \dots, M \quad (3)$$

Con el único requerimiento que $s_i(0) \in [-1,1]$, $\forall i \; y \; a_m(0) > 0, \forall m$. Esto útlimo implica a partir de (3) que $a_m(t) > 0, \forall m, t$. En el texto publicado[21] demostraron que el sistema (2-3) siempre encuentra soluciones a los problemas k-sat (codificados en la matriz C) cuando existe, en casi todas las condiciones iniciales (a excepción de un conjunto de medidas de Lebesgue cero). Esto se demuestra el punto 2.3.4 de este documento.

Los investigadores para plantear su algoritmo sugirieron que el comportamiento caótico puede ser una característica genérica de algoritmos de búsqueda de soluciones en problemas de optimización complejos, surge esta afirmación de las publicaciones de Elser[22] en "Searching with iterated maps". En este paper se analiza la búsqueda de soluciones para satisfacer restricciones en competencia, donde satisfacer cada uno independientemente no platea un desafío. Propone un mapa iterativo construido a través de las proyecciones de los dos conjuntos de restricciones, este tipo de algoritmo puede utilizarse para gran cantidad de aplicaciones que procesan señales (desde el plegamiento de proteínas hasta el Sudoku).

Para la satisfactibilidad lógica específica se debe decidir entre m cláusulas booleanas simultáneas correspondiendo cada una a las literales de una lista de n variables. Una incorporación natural en el espacio euclidiano de n números reales, asociado a las variables booleanas x, y otras m variables y, o las cláusulas.

La proyección P_{com} de las restricciones C_x=y, dónde C es escasa, puede ser calculada con una precisión suficiente en el tiempo lineal en *n* y *m* usando la minimización del gradiente conjugado. Para la segunda restricción, se imponen valores discretos para las variables x: *s* para Verdadero y –*s* para Falso. Estas asignaciones, cuando la compatibilidad de las restricciones es satisfecha implica $y \in \{-3, -1, 1, 1\}$. Para completar la segunda restricción, se impone $y \geq -1$, dado que una clásula no está satisfecha cuando y=-3. La correspondiente proyección P_{val} es dada por el mapa $x \to s$ sgn(x) y $y \to \max(-1, y)$.

[21]Optimization hardness as transient chaos in an analog approach to constraint satisfaction http://arxiv.org/pdf/1208.0526v1.pdf

[22]Searching with iterated maps http://www.pnas.org/content/104/2/418.abstract

Retomando la resolución del Sudoku propuesta por Ercsey-Ravasz y Toroczkai, existe una relación entre la satisfacción de restricciones para este tipo de problemas y la de solución de problemas de planificación o scheduling que se planteará luego en este informe.

En el documento "Chaos Within Sudoku" (2012) se realiza un análisis acerca de la complejidad computacional del flujo de:

$$\frac{ds_i}{dt} = (-\nabla_s V(s,a))_i = \sum_{m=1}^{M} 2a_m c_{mi} K_{mi}(s), \quad i = 1,\ldots,N \qquad (1)$$

$$\frac{da_m}{dt} = a_m K_m(s), \quad m = 1,\ldots,M \qquad (2)$$

En la siguiente figura (Ilustración 10) se muestra que incluso en la "fase congelada", la fracción del problema no resuelto en el tiempo *t* decae exponencialmente con *t*, debido a una ley asociada al comportamiento caótico $p(t) = re^{-\lambda(N)t}$. La velocidad de desintegración λ(N) obedece a $\lambda(N)=bN^{-\beta}$, con $\beta \cong 1{,}6$.

De estas dos ecuaciones, el tiempo continuo *t(p,N)* necesario para resolver una (1-p)-ésima fracción fija de fórmulas al azar (o perderlo resolviendo la p-ésima fracción de uno de ellos) es: $t(p,N) = b^{-1}N^{\beta} \ln(r/p)$.

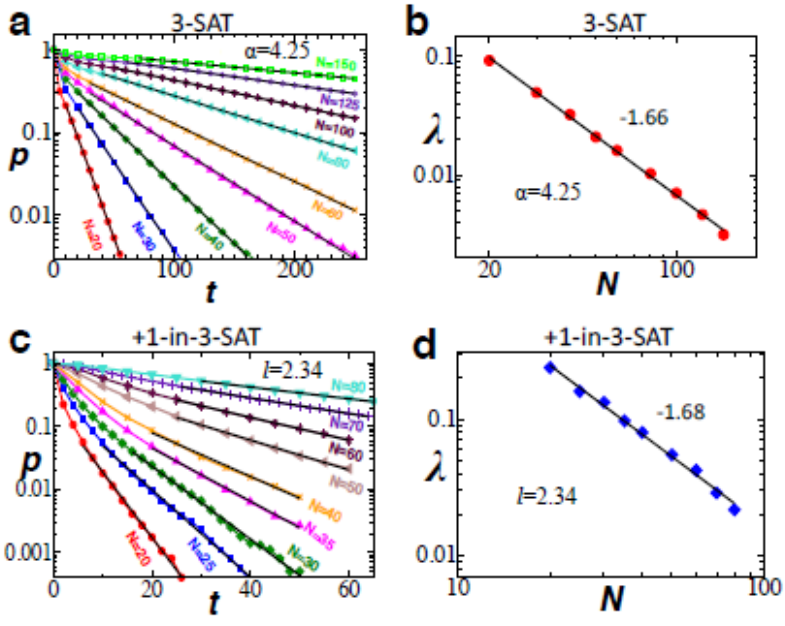


Ilustración 12 Complejidad computacional

3. Marco teórico

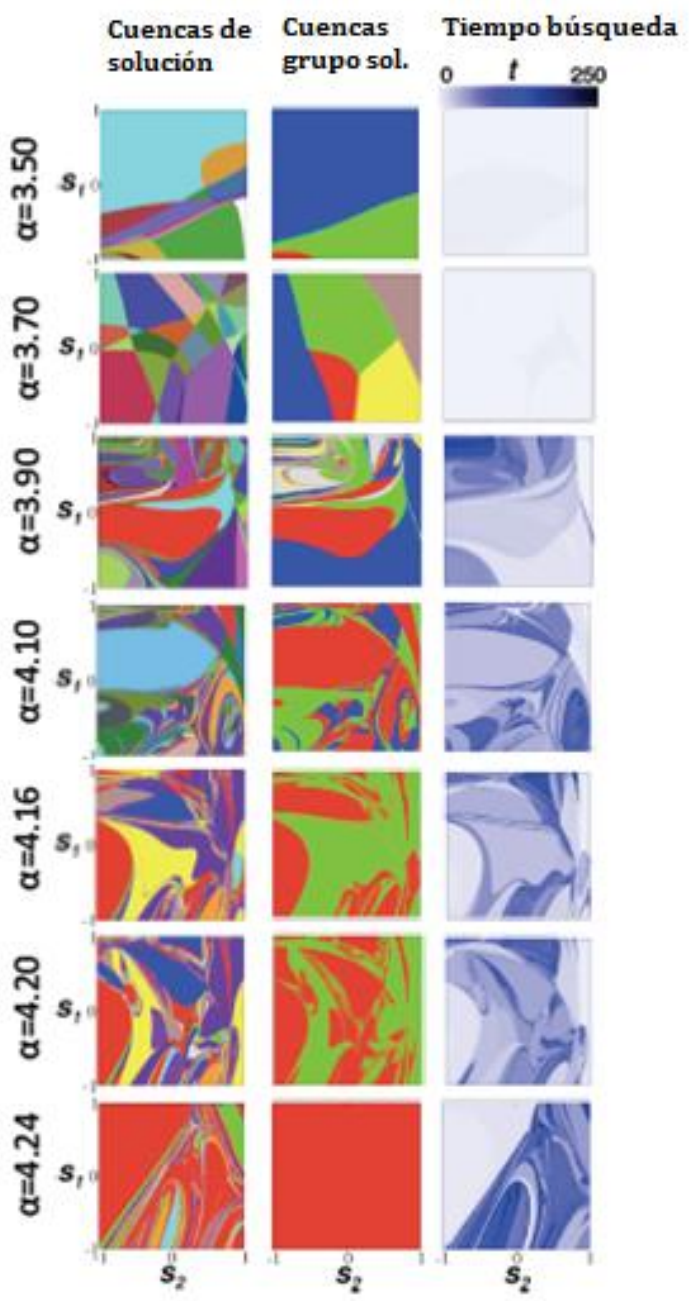


Ilustración 13 Cuencas del atractor

En la figura 11 se observa en a la fracción de problemas *p(t)* que aún no se resolvieron en el tiempo continuo t para 3-SAT en α=4,25, para N=20, 30, 40, 50, 60, 80, 100, 125, 150. En promedio se realizan 10^5 instancias para cada N. En b se observa la caída dada por el ratio ya introducido: λ(N). En c se observan los problemas *p(t)* aún no resueltos cuando l=2,34, para N=20, 25, 30, 35, 40, 50, 60, 70, 80. Para cada instancia la dinámica ha comenzado en paralelo desde 10 líneas mostradas en la caída exponencial, en promedio toma

3. Marco teórico

10^4 para cada N. En d se muestra que el ratio de caída presenta el mismo comportamiento que en b con el exponente $\beta \approx 1{,}68$.

Para hallar la solución al problema del Sudoku es necesario simular el comportamiento de las ecuaciones 1 y 2, para ello se utiliza un método adaptativo de quinto orden Cash-Karp Runge-Kutta con control de error de truncamiento local para asegurar la exactitud. Con el fin de mantener la trayectoria numérica dentro de un pequeño número, preestablecido alrededor de la verdadera trayectoria.

Dadas las reglas como ecuaciones diferenciales es necesario simularlas, el modo más sencillo es con el método de Euler[23], pero debido a que no es tan preciso los autores optaron por el algoritmo Runge Kutta (RK4), este método logra un orden de precisión superior, por esto es uno de los procedimientos más difundidos y exactos para obtener soluciones numéricas a problemas de valor inicial. (Cálculo Científico con Matlab y Octave, Querteroni y Saleri, 2006)

$$k_1 = f\big(y(t)\big), t)$$

$$k_2 = f(y(t) + 0.5\Delta t k_1, t + 0.5\Delta t)$$

$$k_3 = f(y(t) + 0.5\Delta t k_2, t + 0.5\Delta t)$$

$$k_4 = f(y(t) + t k_3, t + \Delta t)$$

$$y(t + \Delta t) = y(t) + \frac{1}{6}\Delta t(k_1 + 2k_2 + 2k_3 + k_4)$$

$$\begin{aligned} k_1 &= f(y(t), t) \\ k_2 &= f(y(t) + 0.5\Delta t k_1, t + 0.5\Delta t) \\ k_3 &= f(y(t) + 0.5\Delta t k_2, t + 0.5\Delta t) \\ k_4 &= f(y(t) + \Delta t k_3, t + \Delta t) \\ y(t + \Delta t) &= y(t) + \frac{1}{6}\Delta t(k_1 + 2k_2 + 2k_3 + k_4) \end{aligned}$$

En este método requiere que las simulaciones sean correctas. El método funciona computando $y(t+\Delta t)$ para Runge Kutta de cuarto y quinto orden. Una gran diferencia entre los dos indica un error grande. Si es este el caso, un método denominado ode45 que provee

[23] Método de Euler: procedimiento de integración numérica para resolver ecuaciones diferenciales ordinarias a partir de un valor inicial dado.

3. Marco teórico

Matlab reduce el tamaño del paso Δt, esto reduce el tamaño del error debajo del predefinido.

Las condiciones iniciales que se tomarán para realizar la simulación serán:

- Valores aleatorios para s_N (entre -1 y 1).
- Valores aleatorios para a_m (mayor a 0).

Las soluciones para el problema SAT serán puntos fijos para el sistema dinámico definido, las propiedades de dicho sistema son: que no hay límites en el ciclo, para satisfacer la fórmula el único punto fijo de los atractores son el mínimo global de V, con V=0. La fórmula utiliza la variable ilimitada a_m.

La aparición del caos transitorio es una característica fundamental de la dinámica de búsqueda y se puede utilizar para clasificar problemas por su dureza. El límite termodinámico ($N \to \infty$, $M \to \infty$, $\alpha = M / N$) de un k-SAT aleatorio aparece como una fase de transición en la llamada "transición caótica del punto α_x" en términos de la densidad de restricción $a=M/N$. Entonces no existe límite termodinámico para Sudokus con N<729, no se puede definir un parámetro simple que corresponda al sistema dinámico. Es así que se hace uso de una variable de escape *k.*

Este ratio de escape denominado k es una cantidad que teóricamente puede ser expresada como el cero del determinante espectral de la evolución del operador correspondiente al sistema dinámico y puede aproximarse utilizando la maquinaria del ciclo de expansión basado en las funciones dinámicas zeta. Representa una medida invariante que caracteriza el caos y que no depende de la distribución de las condiciones iniciales, o los detalles de la región donde se mide el escape.

Los autores del artículo graficaron la distribución de la probabilidad de no resolución en el tiempo t, la distribución es obtenida en diversas condiciones iniciales, el ratio de caída muestra un rango de variación entre los Sudokus, que se modifica de acuerdo a su dificultad.

Elaboraron luego de la resolución una escala (especificada al comienzo de este punto) que permite cuantificar la dificultad de los tableros de Sudoku. La escala se ajusta a la medida de dificultad humana establecida computacionalmente de acuerdo a las técnicas de resolución necesarias para hallar la configuración correcta.

3. Marco teórico

4. Modelo teórico

En el presente trabajo se utiliza la metodología de resolución de problemas mediante computadoras y herramientas de programación planteada por Luis Joyanes Aguilar en su libro: "Fundamentos de la programación" (1999).

En este libro se introducen las fases de resolución de problemas, ellas son:

- Análisis del problema
- Diseño del algoritmo
- Codificación
- Compilación y ejecución
- Verificación
- Depuración
- Mantenimiento
- Documentación

Estas etapas constituyen el ciclo de vida del software y determinan las fases y etapas para solucionar un problema. Las primeras dos conducen al diseño detallado del algoritmo, la tercera es la implementación, la compilación y ejecución de la programación (se traduce y permite corregir el programa). En la verificación y depuración se buscan errores que se presentaron en las etapas anteriores y se eliminan, para finalmente documentar la solución propuesta.

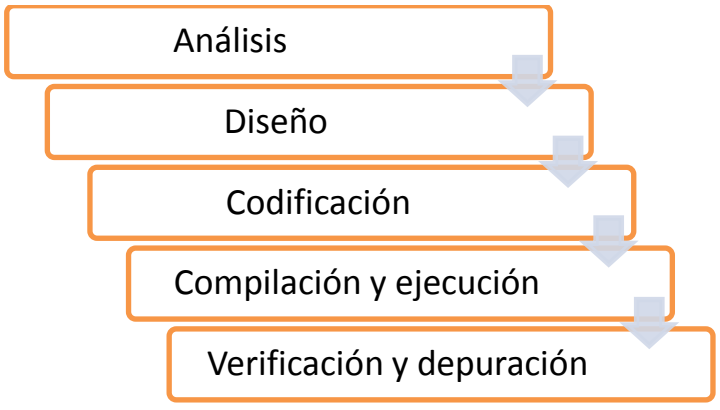


Ilustración 12 Etapas solución problema

De la evaluación de las etapas anteriores surge una estimación de la duración de cada fase, estipulando que se arriba a la solución en aproximadamente 2 meses, en la tabla 9 se refleja esa programación de tareas con las fechas correspondientes. Y la figura 15 es un diagrama

de Gantt que permite el análisis de la sucesión de tareas propuesta y la observación de desvíos para su tratamiento oportuno.

Nombre de la tarea	Duración	Comienzo	Fin	Predecesora
Análisis del problema	7	02/09/2013	10/09/2013	
Diseño del algoritmo	13	11/09/2013	27/09/2013	1
Codificación	20	30/09/2013	25/10/2013	2
Compilación y ejecución	2	28/10/2013	29/10/2013	3
Verificación	3	30/10/2013	01/11/2013	4
Depuración	3	4/11/2013	6/11/2013	5
Documentación	10	07/11/2013	20/11/2013	6

Tabla 9 Tareas y duración

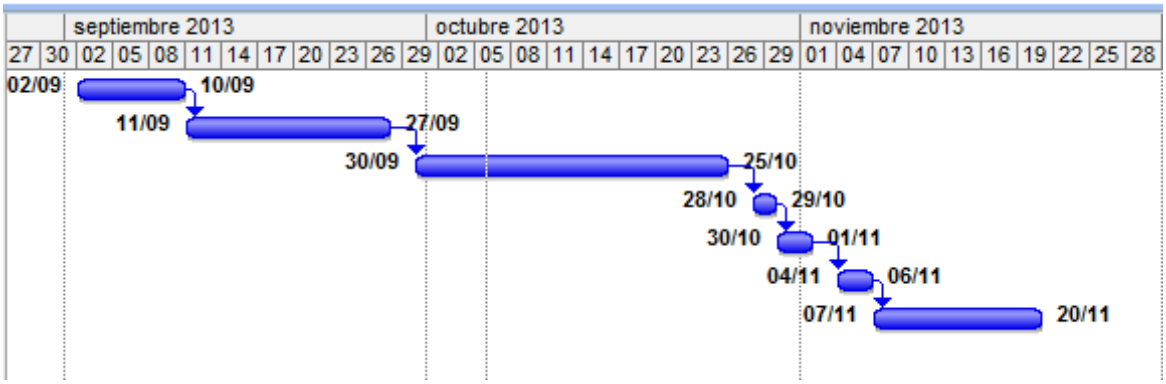


Ilustración 13 Diagrama de Gantt

Recordando que un algoritmo es un método para resolver un problema mediante una serie de pasos precisos, definidos y finitos[24], en los ítems sucesivos de este documento se desarrolla el algoritmo propuesto para resolver el problema de la planificación óptima de la producción cuando se tiene un número de trabajos y de máquinas que deben desarrollarse en un determinado orden.

[24] "Fundamentos de programación". Luis Joyanes Aguilar (1999)

4. Modelo teórico

4.1 Análisis del problema

En esta primera fase se requiere una clara definición de la situación problemática para que en los sucesivos pasos se obtenga la solución necesaria para dar respuesta a la dificultad planteada.

Para realizar un correcto análisis de la situación se utiliza el modelo desarrollado por la Gesellschaft für Technische Zusammenarbeit (GTZ) Organismo Alemán para la Cooperación Técnica, que forma parte del Ministerio Federal de Cooperación Económica del gobierno. Este método se conoce como ZOPP, que es la sigla de la denominación alemana "ZielOrientierte ProjektPlanung". (1981)

Este método constituye una guía para el trabajo, genera un marco de estudio y permite una visualización sencilla de la situación y sus alternativas de resolución. Debido a que los objetivos sólo pueden ser formulados claramente si las causas y los efectos de los problemas a resolver han sido analizados previamente (análisis de problemas). Este enfoque considera que los problemas no son hipótesis abstractas, por el contrario, afectan a entes, organizaciones o grupos sociales, y por lo tanto hay que determinar los grupos afectados y sus intereses (análisis de la participación).

Además es necesario evaluar los objetivos que incluyan posibles soluciones, para esto se confecciona una matriz de planificación que ordena los objetivos de acuerdo a su coherencia, plausibilidad y realismo. Se obtienen así diferentes niveles: el objetivo del proyecto que aporta al objetivo superior, las actividades y resultados alcanzados, que contribuyen al cumplimiento del objetivo del proyecto. Los distintos niveles se entrelazan por la hipótesis basada en las condiciones que rodean al proyecto. Las influencias externas que representan un riesgo para la implementación del proyecto son supuestos importantes, mostrando una dependencia del proyecto con respecto al medio y permitirá apreciar y reducir los riesgos que corre el proyecto.

Se trata de un método de planificación flexible, eficiente, con aplicación realista, que utiliza pasos sucesivos y un enfoque de trabajo. Los pasos principales son: el análisis del problema, el de participación, el de objetivos y las alternativas. Con estos pasos se desarrollará una matriz compuesta de: objetivos, supuestos, indicadores verificables y fuentes de verificación.

4. Modelo teórico

Análisis de problemas:

En este paso se debe identificar el problema central, considerando que un problema no es la ausencia de solución, sino un estado existente negativo. Por esta razón la situación debe plantearse de ese modo. En este caso el problema central será:

LA PRODUCCIÓN NO ES PLANIFICADA

Debido a los cambios repentinos en la demanda y las restricciones productivos se necesita elaborar programaciones de tareas productivas (scheduling) de manera efectiva. Debido a lo dificultoso de esta tarea es necesario construir herramientas computacionales que faciliten la planificación.

Es decir que lo que se necesita es un método para realizar esta planificación de la producción considerando la importancia del control de costos, el tiempo de producción, los conflictos entre recursos, los tiempos ociosos, la insatisfacción del cliente, lo que deriva en fuerte impacto a las ganancias organizacionales.

La programación de la producción tiene un valor fundamental en el proceso productivo, es debido a ello que en las grandes empresas existen equipos sumamente capacitados que con la colaboración de diferentes herramientas estadísticas, financieras y computacionales elaboran planes flexibles acordes a las proyecciones de la demanda.

Pero al evaluar la situación en las PyMEs se hace notar que el problema se acrecienta cuando la determinación de la secuencia óptima de operaciones, dado los tiempos de proceso y el orden de ejecución de cada tarea, queda en manos de los dueños de la empresa o de gerentes poco capacitados en la función de planificación industrial.

Hay que considerar que de acuerdo a datos relevado por la AFIP[25] existen casi un millón de empresas en el país, de las cuales un 99% se encuadran dentro las MiPyMES, un 80% son micro, un 16% pequeñas y un 3% medianas, generando en total el 69% del empleo total y el 42% de las ventas agregadas. Es decir que la importancia para la economía del país es superlativa. Pero de acuerdo a la Asociación Argentina para el Desarrollo de la Pequeña y Mediana Empresa el nivel de mortandad de las mismas es muy alto, un 80% de las mismas fracasa antes de alcanzar los 5 años de vida. Algunos motivos de ello son razones extrínsecas (que se vinculan al contexto económico, político y social) y otras son intrínsecas relacionadas a la poca capacidad de gestión y planificación de los dueños, a las escasas

[25] Fuente: proyecto de Ley 25300. Fomento a la micro, pequeña y mediana empresa argentina.

herramientas e instrumentos que se hayan disponibles para el sector, desaprovechando los beneficios de contar con servicios e información que brinda apoyo al tomar decisiones, para así potenciar y consolidar los emprendimientos.

Es por todo ello que es necesario crear herramientas sencillas, que hagan uso de métodos eficientes para obtener información relevante y confiable para tomar decisiones en el contexto de gestión de pequeñas y medianas empresas.

Cabe destacar que el proceso de planificación es extremadamente complejo dada la cantidad de variables que intervienen en el modelo de optimización combinatoria, y el tiempo de cómputo necesario para resolverlos que crece desproporcionadamente conforme aumenta el tamaño de la dificultad, este tipo de problemas pertenece a la clase NP-Completa.

Los métodos tradicionales de programación de la producción no son capaces de proporcionar una reacción ante las excepciones que se producen en tiempo real en un plazo de tiempo razonable. Por lo tanto, las soluciones que obtienen no son eficientes. Imposibilitando adaptarse a los constantes cambios que se generan en el contexto empresarial.

Además los métodos de optimización propuestos hasta el momento no explotan los algoritmos de optimización, dado que alguno de ellos son del tipo discretos, y no utilizan algunas de las formas que brinda la teoría del caos.

De acuerdo a la situación problemática planteada anteriormente se pueden realizar distintos enunciados que constituyan las causas del problema de la no planificación de la producción.

- Se desconoce el orden de ejecución de las operaciones para minimizar tiempos muertos.
- Las tareas se desarrollan en forma aleatoria desconociendo la necesidad de material y recursos diaria.
- No existe una estimación de la duración del proceso productivo.
- Baja productividad debido al mal uso de los recursos de la empresa.
- Falta de control en desvío de costos y tiempo por la falta de pautas.
- Relación entre beneficios de aplicación de métodos de programación y costos de solución muy bajo.
- Métodos de planificación excesivamente onerosos para pequeñas empresas.
- Baja contratación de mano de obra calificada en Scheduling.
- Escases de modelos previstos para la disposición por proceso (process layout).

4. Modelo teórico

- Falta de definición de rutas de trabajo para las operaciones.

De los puntos analizados anteriormente se desprende el siguiente árbol de problema:

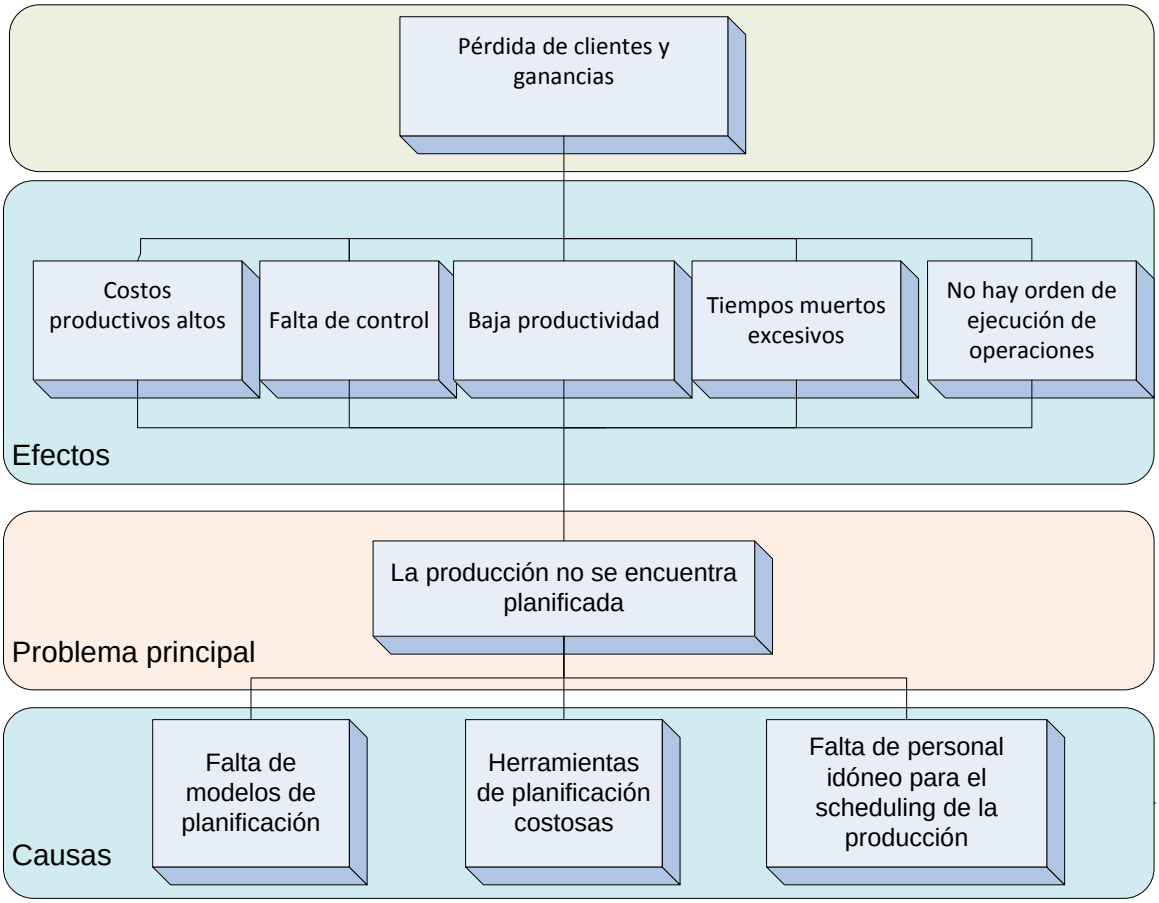


Ilustración 14 Árbol de problemas

Análisis de objetivos:

En este punto se realiza una descripción de la situación que se desea alcanzar mediante la solución del problema, además se identifican distintas alternativas para el proyecto.

La programación de la producción es de suma importancia dado el rol crítico que desempeña en la consecución de los objetivos organizacionales, afectando no solo la necesidad de materiales, y la administración de tiempos ociosos, sino que es un estándar que permite la medición de desvíos mediante simples diagramas de Gantt y el cronometrado de los tiempos operativos.

El objetivo es eliminar los altos costos asociados a la planificación óptima de la producción cuando cada trabajo utiliza una máquina por vez en un determinado orden que puede ser distinto al de otros trabajos. Es decir que se pretende crear un método que optimice el ma-

4. Modelo teórico

kespan (duración de todo el proceso productivo), que pueda ser utilizado por gerentes y responsables que no posean grandes conocimientos sobre investigación de operaciones y con una velocidad de proceso superior a la que demandaría una búsqueda exhaustiva de la mejor propuesta. Hay que considerar el nivel de dificultad que este tipo de problemas conlleva y dado su tratamiento como problema NP-completo alcanzar un algoritmo capaz de resolverlo en un tiempo lógico a su dificultad.

Al aplicar este método se pretende aumentar la efectividad en la toma de decisiones que se relacionan con la gestión productiva una vez definidas la planeación estratégica y la planeación agregada.

Será fundamental que el método propuesto sea novedoso, y que la interfaz que utilice el usuario sea sencilla es decir que al ingresar la cantidad de trabajos, de máquinas, el orden de las operaciones y la duración de las mismas, pueda obtener una planificación óptima de la producción y un diagrama de Gantt para observar los tiempos de inicio y finalización de cada tarea.

De este modo los encargados de planificar la producción podrán llevar a cabo controles, determinar costos y eliminar la mayoría de los tiempos muertos tanto de máquinas como operarios, todo esto acarreará un mejor contacto con el cliente, cumpliendo plazos y presupuestos, lo que derivará en beneficios económicos para la organización.

4. Modelo teórico

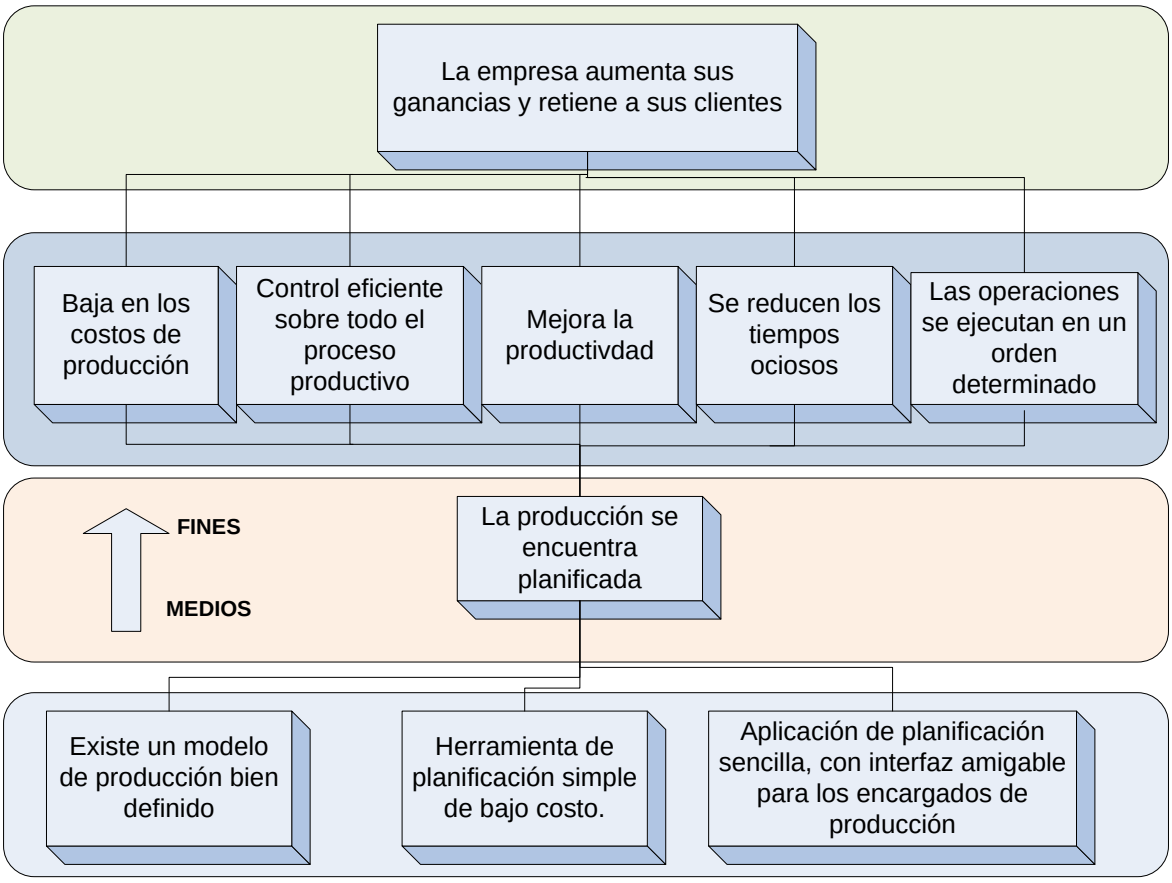


Ilustración 15 Árbol de objetivos

Análisis de alternativas

De acuerdo a los objetivos especificados en el punto anterior se generan tres alternativas que se evalúan de acuerdo a los recursos necesarios, la probabilidad de alcanzar los objetivos, la factibilidad de desarrollo, la relación costo/beneficio, los riesgos y la sostenibilidad.

- Solución mediante método heurístico:

La técnica más común para este tipo de solución es relajar el problema, es decir que se relajan algunas restricciones con el fin de obtener un problema simplificado que pueda ser resuelto óptimamente en un tiempo polinomial. Luego, el coste de la solución para el problema relajado se toma como estimación del coste del problema real. En el caso del problema JSS, las restricciones que pueden ser relajadas son las secuenciales, de capacidad y de no-expulsión. Aunque el heurístico es claramente una subestimación del coste óptimo, no es de esperar que la diferencia sea uniforme para todos los estados a lo largo del proceso de búsqueda. Sin embargo, se espera que esta diferencia sea grande al principio de la

4. Modelo teórico

búsqueda, disminuya a medida que la búsqueda avanza y que tienda a ser nula al final de ésta.

Los métodos de aproximación no garantizan una solución exacta pero son capaces de obtener soluciones cercanas con tiempos de cálculo moderados, por lo tanto son útiles para problemas muy grandes. Algunos métodos utilizan un enlace bilateral entre la investigación de operaciones y la inteligencia artificial, y se encuentran inspirados en fenómenos naturales y resolución inteligente de problemas.

- Aplicación de un software de planificación de la producción enlatado

Existen soluciones desarrolladas por empresas extranjeras que proveen soporte a la hora de realizar la planificación de la programación, una de ellas es Exact, con su software Job-Boss, este puede aplicarse en procesos de fabricación medianos a pequeños, de manera que la ordenación de los trabajos se realiza automáticamente, y el proceso resultante es lo suficientemente flexible como para gestionar un cambio, y se transforma en base estructural para sostener el crecimiento.

Este es uno de los software más utilizados a nivel mundial para la planificación de tareas industriales, una de las razones es que es en base a las necesidades del usuario, sencillo, y hace uso de tecnología superior para dar respuesta a la problemática.

El aplicativo brinda flexibilidad y permite controlar la gestión de la empresa, recopila datos y permite elaborar planes de necesidad de materiales, además brinda soporte para toda la producción incluyendo los centros de congestiones de trabajo o entregas finales proyectados, requerimientos de materiales de planificación si el material se compra a valores o directamente con el trabajo, el trabajo en tiempo real y la información de costos de procesamiento de transacciones y la información de entrega/envío. Además, integra todos los datos financieros, incluyendo la facturación al cliente y cobros, facturación de proveedores y pagos.

Los costos de adquisición de este tipo de software ronden entre los u$s 1000 y los u$s20000.

- Creación de nuevo algoritmo de solución del JSSP que utiliza un solucionador dinámico de tiempo contínuo.

Esta propuesta consiste en una solución que traduce el problema ingresado por el usuario (es decir el número de trabajos, de máquinas, el orden de las operaciones y su tiempo de

4. Modelo teórico

procesamiento) a un problema de satisfacción de restricciones complejas. De este modo se realiza un mapeo del JSSP en un sistema dinámico determinista de tiempo continuo hasta hallar la solución.

Además provee una medida de la dificultad denotada por un ratio de escape del caos transitorio por el que atraviesa hasta encontrar la solución. Este valor se relaciona con la complejidad que requiere, el tiempo y pasos de solución.

Es decir que mediante la utilización de un algoritmo determinista tratado como un sistema dinámico se da respuesta al complejo problema de la planificación industrial, se explota en esta solución el comportamiento complejo del sistema dinámico no lineal y su relación con el caos.

Los actuales algoritmos de optimización, son en su mayoría discretos y no pueden tratar por lo tanto con métodos relacionados a la teoría del caos, en este caso se trata el problema de satisfactibilidad booleana mediante un conjunto de ecuaciones diferenciales ordinarias, este método se encuentra naturalmente adaptado para ser tratado con métodos propios de la teoría del caos y de este modo es posible determinar un relación entre la optimización de la complejidad y el comportamiento caótico.

Mediante el uso del programa que surge de la aplicación del algoritmo los encargados de la planificación obtendrán un plan con el inicio y el fin de cada operación y un diagrama de Gantt que posibilite el control del desarrollo de la producción.

Análisis de alternativas:

Criterio	Alternativa A	Alternativa B	Alternativa C
Define bien el modelo de producción	5	10	10
Herramienta de planificación simple, de interfaz amigable	5	10	10
Bajo costo	10	1	10
Factibilidad de implementación	7	10	8
Factibilidad de desarrollo	7	7	7
Relación costo/beneficio	10	1	10
Tasa de error baja	5	10	10
Sostenibilidad de la solución	5	10	10
TOTAL	**54**	**59**	**75**

Tabla 10 Ponderación de alternativas

Por lo tanto la alternativa seleccionada será la C.

4. Modelo teórico

Objetivos, resultados y actividades del proyecto

A continuación se presenta una matriz de planeación del proyecto en la que se resumen los objetivos y actividades, considerando los supuestos, indicadores y las fuentes de verificación.

Resumen de objetivos/actividades	**Indicadores verificables Objetivamente**	**Fuentes de verificación**	**Supuestos Importantes**
Objetivo superior: Aumentar ganancias y retener clientes	En 2014 las ganancias de la empresa aumentaron un 10%.	Estado de resultados de la organización.	Anteriormente no se utilizaba ningún método de planificación
Objetivo del proyecto: Seguir un plan de producción con tiempos de inicio y fin para cada operación.	Se respetan los tiempos de entrega en un 90%.	Contratos y facturas.	Actualmente se presentan demoras en los tiempos de entrega establecidos.
Resultados/Productos:			
Reducir tiempos inactivos	El tiempo perdido por falta de recursos se reduce un 50%.	Cálculo de desvíos de los tiempos estándar: Tiempo ocioso/tiempo total	Los trabajadores y las máquinas tienen tiempos ociosos.
Aumentar la productividad	El volumen de piezas elaboradas por día aumentó un 25%.	Control de stock de piezas acabadas.	No existen pautas de producción actuales.
Bajar los costos	El pago de horas extras se redujo un 10%.	Liquidación de sueldos y horas extras.	Se pagan horas extras debido a la necesidad de entregar mercadería.
Controlar desvíos	Se redujo un 25% las demoras de entrega por desvíos en el proceso productivo.	Diagrama de Gantt	Actualmente no se controlan las desviaciones, ya que no existe plan.
Actividades: Calcular tiempos de cada operación que compone un trabajo. Establece orden de procesamiento de cada operación de un trabajo. Volcarlos al programa ingresado número de máquinas y trabajo. Convertir el problema a forma normal conjuntiva. Obtener secuencia óptima de producción mediante la simulación de un sistema de ecuaciones apli-	**Insumos** Los gerentes o encargados de producción deberán realizar las mediciones correspondientes y volcarlos a una tabla. El software deberá realizar una conversión a CNF, elaborar una matriz en base a la misma y luego realizar una simulación hasta hallar los valores de inicio y fin correspondientes. Posterior al ingreso de la fecha de inicio de la	**Presupuesto** $2500 (ver cálculo con modelo COCOMO en anexo A)	

4. Modelo teórico

cados a una matriz. Fijar tiempos de inicio y fin de cada operación Obtener diagrama de Gantt. Controlar desvíos	producción se obtiene un diagrama de Gantt que permite llevar a cabo el control del proceso.	

Tabla 11 Matriz de planeación

Resumen de objetivos y actividades:

De lo expuesto anteriormente surgen una lista de actividades a realizar para llevar a cabo el proyecto, también se elabora una agenda y un diagrama de Gantt. Mediante la realización de estas actividades y su implementación en una organización se alcanzarán los objetivos propuestos.

Para llevar a cabo la alternativa seleccionada se deberá:

- Crear un formulario de ingreso del problema con el número de máquinas y de trabajos.

- Leer el problema y obtener el mínimo makespan, sumando la duración de las tareas, sin un orden establecido.

- Convertir el problema a forma normal conjuntiva.

- Crear una matriz denominada Cmi, en donde c toma valor 0 si una variable proposicional del problema planteado no está presente, 1 si se encuentra en el problema, y -1 si se encuentra en forma negada. Luego para cada una de esas variables proposicionales x_i se calculan valores aleatorios denominados s_i que se encuentran entre -1 y 1, determinando "grados de verdad" de acuerdo a su valor. Otros valores aleatorios que se obtienen son las a_m, que actúa como función de ajuste de la fórmula de energía, estos valores se calculan y son superiores a 0.

- Luego la matriz Cmi es procesada por un sistema de ecuaciones donde una es una función de restricciones y la otra una función de energía, esta simulación se ejecuta hasta que todos los puntos son fijados.

- Convertir la matriz de resultados obtenida en los tiempos de inicio de cada operación.

- Solicitar la fecha de inicio del proceso productivo y calcular fecha de inicio y fin de cada actividad.

- Confeccionar un diagrama de Gantt con los valores obtenidos.

4. Modelo teórico

En base a estas actividades se elabora el siguiente cronograma:

Actividad	Dur.	Fecha inicio	Fecha fin	Antecesor
Crear formulario ingreso	1d	lun 30/09/13	lun 30/09/13	
Leer problema	1d	mar 01/10/13	mar 01/10/13	1
Convertir a CNF	5d	mié 02/10/13	mar 08/10/13	2
Crear matriz	4d	mié 09/10/13	lun 14/10/13	3
Crear simulación	6d	mar 15/10/13	mar 22/10/13	4
Convertir a tiempo de inicio	1d	mié 23/10/13	mié 23/10/13	5
Calcular fechas	1d	jue 24/10/13	jue 24/10/13	6
Confeccionar Gantt	1d	vie 25/10/13	vie 25/10/13	7

Tabla 12 Cronograma actividades

Para finalizar la etapa análisis es necesario determinar la información a producir (salidas), los elementos de información dado (entradas) y los procedimientos necesarios para transformas las entradas a salidas (proceso). La figura 4.4 muestra un diagrama denominado H.I.P.O (jerarquía de entrada, procesos y salida) en el que se notan los subsistemas que componen al sistema reduciendo la complejidad percibida del mismo.

4. Modelo teórico

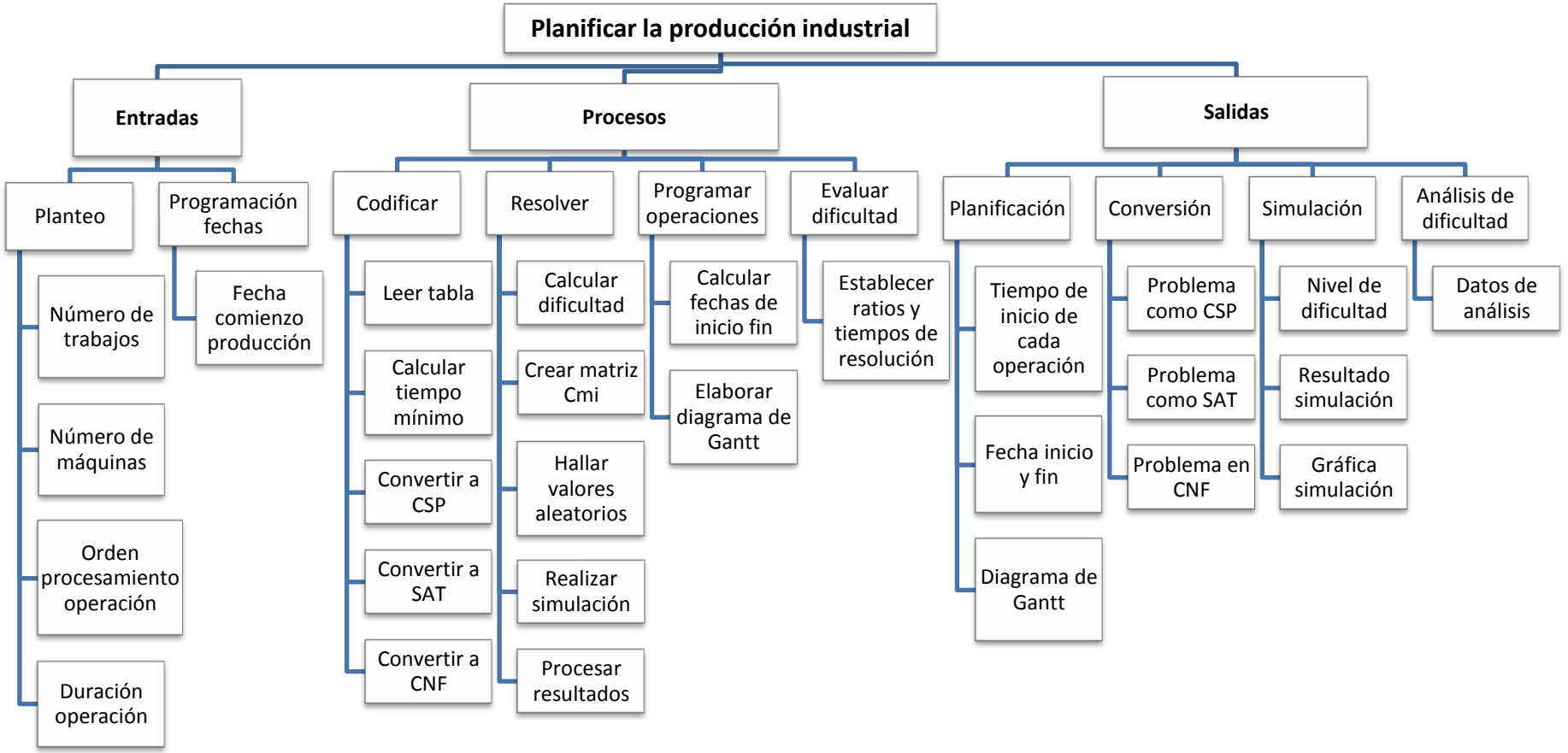


Ilustración 16 H.I.P.O Problema de optimización de la planificación

Con el problema debidamente identificado, y las necesidades a cubrir de entradas, salidas y procesos, organizadas en el diagrama jerárquico 4.4 se continúa con el diseño del algoritmo que dé respuesta a los sub problemas.

4. Modelo teórico

4.2 Diseño

Si en la etapa anterior se determinó qué es lo que el algoritmo debe resolver, en esta se estipula el cómo, es decir que se resuelve el problema graficado en un diagrama de flujo el diseño del algoritmo.

Debido a la dificultad asociada con la resolución de la optimización de la planificación de la producción, será conveniente dividir en subproblemas con niveles más bajos de complejidad. Esta es la técnica de diseño descendente (top-down) o modular. Es decir que se realiza un refinamiento sucesivo donde cada parte del problema se resuelve mediante un módulo.

Para poder resolver cada sub problema independientemente del lenguaje de programación utilizado se diseña el algoritmo, se recurre a dos herramientas: diagrama de flujo y pseudocódigo.

A continuación se presenta la descomposición de subproblemas y sus soluciones:

- **Lectura del problema**
  - Solicitar nº de máquinas y de trabajos
  - Solicitar orden de operaciones de cada trabajo y duración
  - Determinar la duración mínima de cada trabajo y cada máquina

En la figura 19 se presenta el diagrama de flujo del subproceso. Pseudo-código:

```
1  INICIO
2  Ingresar nTrabajos;
3  Ingresar nMáquinas;
4  max=0;
5  Para i=0 hasta nTrabajos
6      Para j=0 hasta nMáquinas
7          Ingresar ordenOperacion[i][j]
8  Ingresar duracionOperación[i][j]
9          sum[i]=sum[i]+duracionOperacion[i][j]
10 Fin_para
11 Fin_para
12 Para i=0 hasta nTrabajos
13     Es max<sum[i]
14         max=sum[i]
15     Fin_Es
16 Fin_para
17 Imprimir max
18 FIN
```

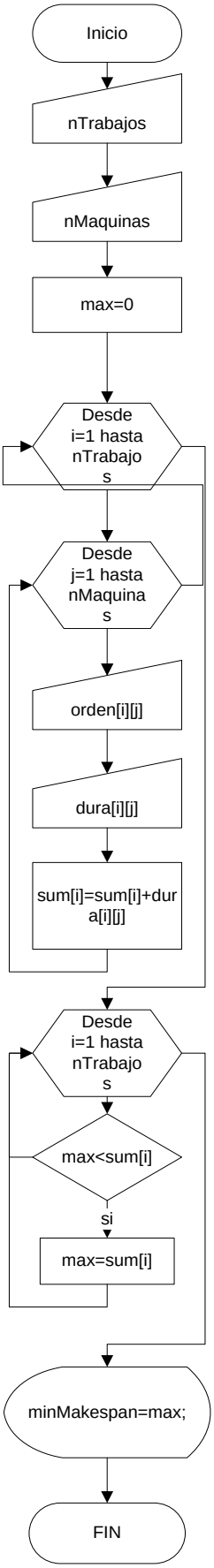


Ilustración 17 Subproceso leer problema

4. Modelo teórico

- **Conversión**
  - Convertir a restricción de CSP
  - Convertir a satisfactibilidad booleana SAT
  - Convertir a forma normal conjuntiva CNF

Para realizar la conversión del problema de Job Shop Scheduling se utiliza una librería creada por Tamura denominada Sugar y que utiliza la codificación por orden descripta en el marco teórico. En el mismo se da como ingreso el problema expresado en CSP (problema de satisfacción de restricciones) y arroja como resultado una conversión a SAT y CNF en archivos de texto. Por lo tanto el algoritmo del subproceso deberá convertir el problema a CSP y luego utilizar la librería para convertirlo a forma normal conjuntiva.

Pseudocódigo

```
INICIO
lb=max;
Leer up;
Escribir archivo jss.csp "(int makespan " lb ub")";
Escribir archivo jss.csp (objective minimize makespan);
Para i=1 hasta nTrabajos
    Para j=1 hasta nMaquinas
Escribir archivo jss.csp "s_"i"_"j "(int s 0"ub")\n";
    Fin_Para
Fin_para
Para i=1 hasta nTrabajos
    Para j=1 hasta nMaquinas
        j1=j+1;
s="s_"i"_"j;
s1="s_"i"_"j1;
Si j1=nMaquinas
    s1="makespan";
Fin_si
p="(<= (+" s" "p") "s1") \n";
Escribir archivo jss.csp "(<= (+ "s" "p") "s1")\n";
Fin_Para
Fin_Para
Para m=1 hasta nMaquinas
    Para i0=1 hasta nTrabajos
        Para i1=i0+1 hasta nTrabajos
j0=maquina[m][i0];
j1=maquina[m][i1];
s0="s_"i0"_"j0;
s1="s_"i1"_"j1;
p0=durac[i0][j0];
p1=durac[i1][j1];
Escribir archivo jss.csp "(or (<= (+ "s0 " " p0") "s1") (<= (+ "s1" "p1") "s0"))\n";
        Fin_Para
    Fin_Para
Fin_Para
Convertir CSP a SAT;
Escribir archivo jss.sat;
Convertir SAT a CNF;
Escribir archivo jss.cnf;
FIN
```

4. Modelo teórico

Como se observa, los últimos dos procesos corresponden a llamadas a la librería Sugar[26], donde se realiza la conversión y escritura de los archivos tomando como entrada el archivo .csp creado por el subproceso, en el que se escribe el problema de la planificación de la producción como uno de Satisfacción de Restricciones.

La codificación por orden es explicada en el punto 3.3.1 de este documento, las restricciones enteras lineales son convertidas a problemas de satisfactibilidad booleana, determinando si es posible su resolución, y posteriormente a CNF, se utiliza esta librería debido a la eficiencia de la transformación a esta forma (Normal Conjuntiva), evitando así realizar una conversión extensa y puesto que la misma se basa en la que propusieron Crawford y Baker para la transformación del problema de Job Shop Scheduling para su resolución mediante algoritmos de optimización.

El formato de escritura del archivo CSP es una convención establecida por el concurso internacional de resolución de este tipo de problemas[27].

En la figura 4.6 se observa el diagrama de flujo de la conversión del problema, el resultado final de la conversión que es un archivo denominado jss.cnf es el que posteriormente se utiliza para resolver el problema mediante la simulación.

[26] http://bach.istc.kobe-u.ac.jp/sugar/

[27] http://www.satcompetition.org/

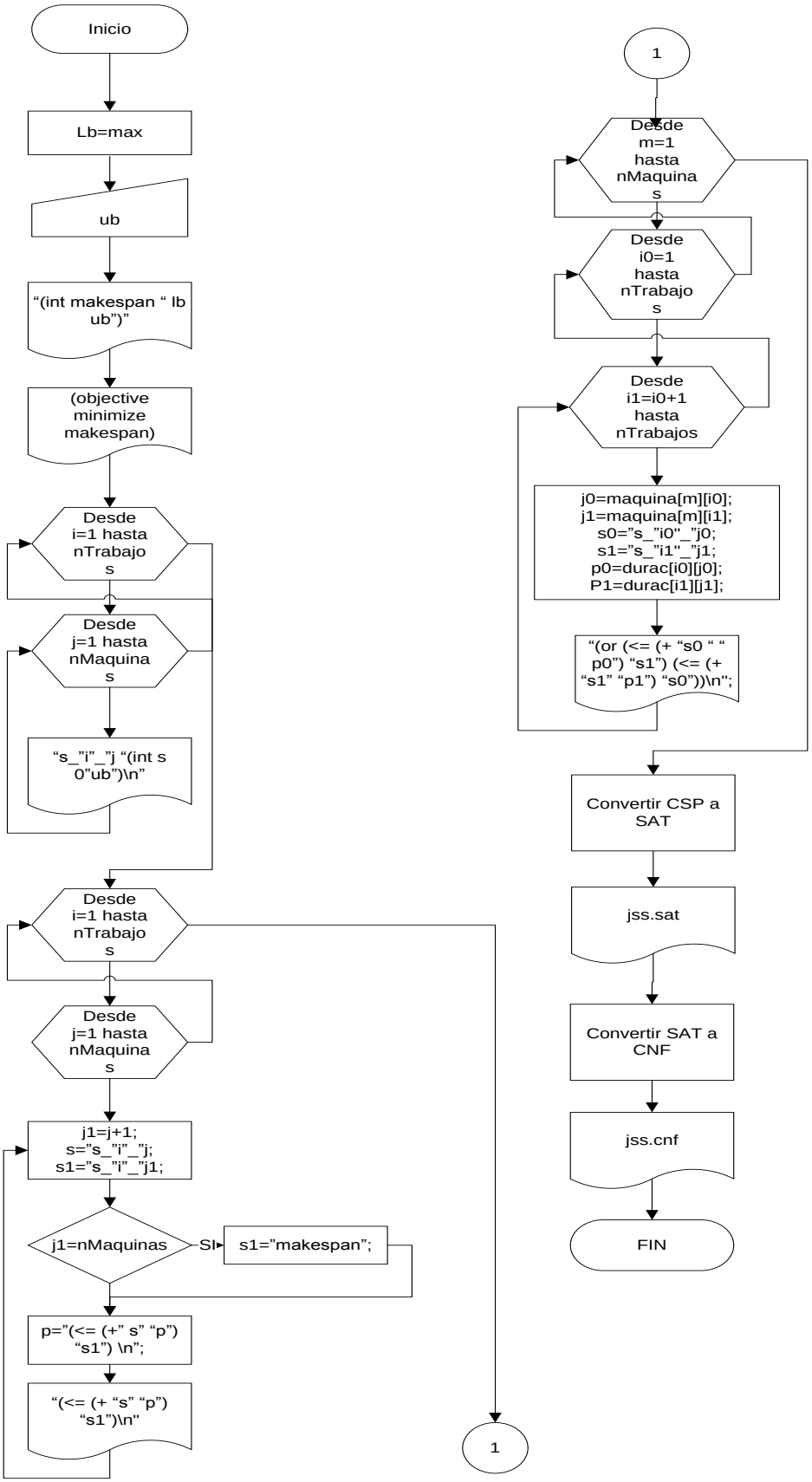


Ilustración 18 Convertir problema

4. Modelo teórico

- **Simulación**
  - Leer el archivo CNF y convertirlo a matriz
  - Obtener valores aleatorios s_{ij} y a_i
  - Realizar simulación con método Runge-Kutta adaptativo de 4-5 orden
  - Obtener valores V o F para cada variable x_{ij}

Este es el subproceso crítico en el que se implementa el solucionador de tiempo continuo para el sistema dinámico. Tomando como entrada el archivo cnf, se lo lee y se crea una matriz compuesta por 0, 1 y -1. Luego se crean dos valores aleatorios uno la variable de "giro" Ss (entre -1 y 1), y la otra: As un valor de ajuste de la función energía entre 0 y 1.

Posteriormente se realiza la simulación mediante el método Runge-Kutta adaptativo de 4-5 orden (figura 4.7), de las fórmulas publicadas por Ercsey-Ravasz y Toroczkai, la simulación termina cuando se haya la energía 0 para cada restricción.

Así se obtiene un archivo que contiene 1 y -1 dependiendo del valor de verdad o no de cada variable.

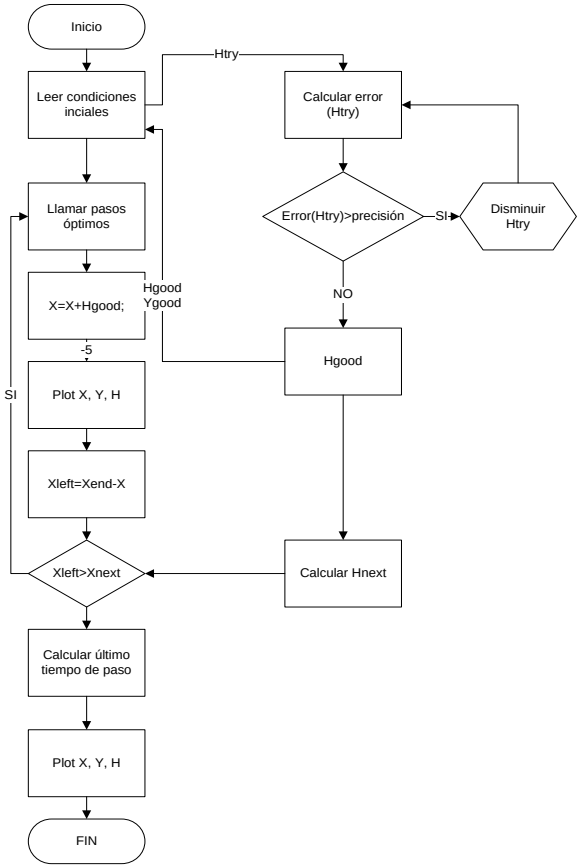


Ilustración 19 Diagrama de flujo Runge-Kutta

4. Modelo teórico

Pseudocódigo del proceso de resolución:

```
INICIO
N=max(jss.cnf);
M=tamaño(jss.cnf);
K=sumar(abs(jss.cnf);
jss[N][M]=leer(jss.cnf);
Para i=1 hasta M
    Para j=1 hasta K[i]
        Var=val. Abs(jss[i][j]);
        Si jss[i][j]==var
            C[i][j]=1;
        Sino
            SI Jss[i][j]==-var
                C[i][j]=-1;
            Sino
                C[i][j]=0;
            FinSI
        FinSI
    FinPara
FinPara
Ss= aleat(1,N)*2-1;
As= aleat(1,M);
Simular ode45 ([0:0.5:175],[Ss, As], tolerancia relativa(1e-3));
Hacer hasta abs(dys <= 0.01)
        sis = y(1:N);
        ams = y(N+1:M+N);
        ss=concatenar (sis',M,1);
        kms = 2 .^ (-K) .* prod(1 - (C.*ss),2);
        dy[N+M][1]=0;
        Para i=1 hasta N
            div = (1- (C(:,i)*sis(i)));
            kmis   = kms ./ (div);
            kmis((isnan(kmis) + isinf(kmis)) > 0) = 0;
            dy(i)  = sum(2 * ams .* C(:,i) .* kmis .* kms);
        FinPara
    dy(N+1:M+N) = ams .* kms; % dam/dt
    dys = dy;
FinHacer
FinSimular
s = sum(abs(Y(:,1:N)),2);
s(s==infinito)=0;
[~,ind] = max(s(:));
sol = sign(Y(ind,1:N));
pasos= ind;
cTime=T(ind);
Graficar simulacion;
Crear resultados.txt;
FIN
```

4. Modelo teórico

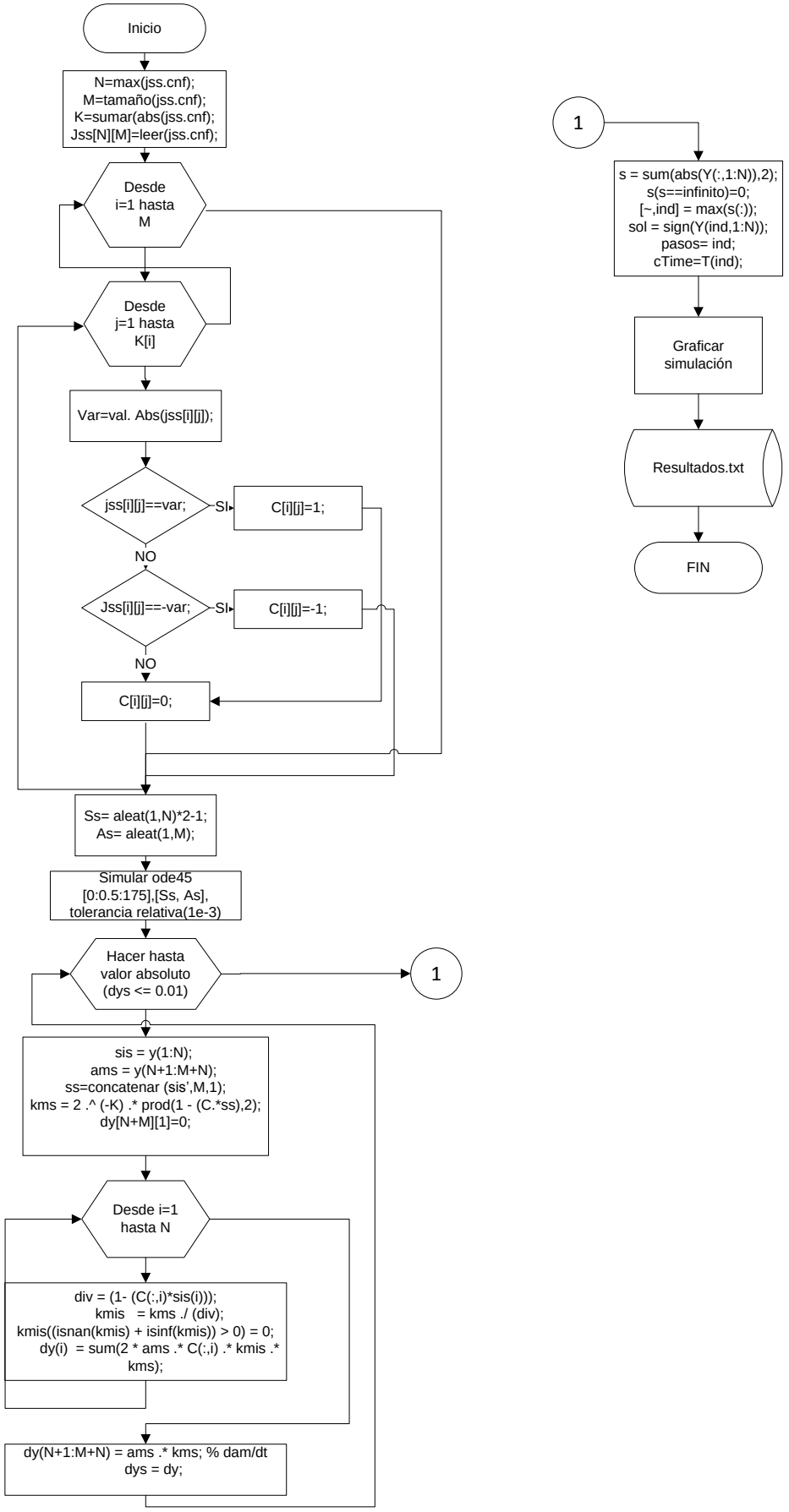


Ilustración 20 Simular

4. Modelo teórico

- **Presentación de la solución**
  - Convertir el archivo de resultados a tiempos de inicio de cada operación
  - Solicitar fecha de inicio de la producción
  - Determinar fecha inicio y finalización de cada operación.
  - Realizar diagrama de Gantt de la planificación.

```
Inicio
Convertir resultado.txt a tInicio[NTrabajos][NMaquinas];
Leer fechaInicio;
Para i=1 a NTrabajos
    Para j=1 a NMaquinas
        tFinal=tInicio[i][j]+duracion[i][j];
        Fecha[i][j]=fechaInicio+tInicio[i][j];
    FinPara
FinPara
Crear gantt;
Fin
```

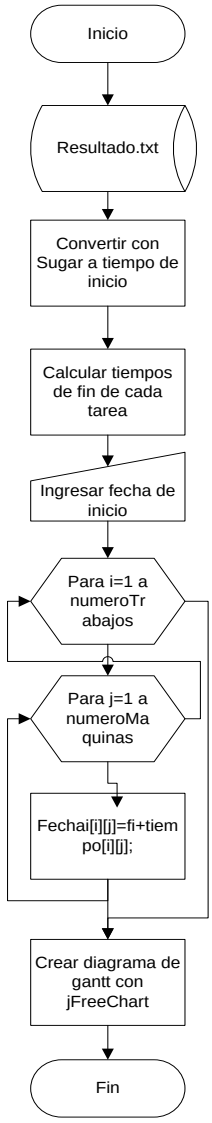


Ilustración 21 Presentar solución

4. Modelo teórico

- **Analizar complejidad**
  - Determinar coeficiente de restricción
  - Especificar: Número de pasos de resolución, tiempo de resolución, gráfica de simulación, cantidad de variables y cantidad de restricciones.

```
INICIO
op=nTrabajos*nMaquinas;
Imprimir nTrabajos;
Imprimir nMaquinas;
Imprimir op;
n=leer(jss.cnf);
m=leer(jss.cnf);
alfa=m/n;
Si op<6
    Si alfa<1.5
        dif="Baja";
        k=  0.0159;
        val= 1,8;
        tprom= 44;
    Sino
        Si alfa < 2
            dif="Media";
            k=  0.0138;
            val= 1,87;
            tprom=60;
        Sino
            dif="Alta";
            k= 0.0201;
            val= 1,7;
            tprom=61;
        FinSi
    FinSi
Sino
    Si op<10
        Si alfa<2
            dif="Baja";
            k= 0.0223;
```

4. Modelo teórico

```
33              val=1.65;
34              tprom=41;
35          Sino
36              Si alfa < 2.5
37                  dif=”Media”;
38                  k=0.0185;
39                  val=1.73;
40                  tprom=49;
41              Sino
42                  dif=”Alta”;
43                  k=0.0179;
44                  val=1.75;
45                  tprom=59;
46              FinSi
47          FinSi
48      Sino
49          Si alfa < 2.5
50              dif=”Baja”;
51              k=0.0197;
52              val=1.7;
53              tprom=69;
54          Sino
55              Si  alfa < 3
56                  dif=”Media”;
57                  k=0.0230;
58                  val=1.64;
59                  tprom=51;
60              Sino
61                  dif=”Alta”;
62                  k=0.0139;
63                  val=1.86;
64                  tprom=66;
65              FinSi
66          FinSi
67      FinSi
68  FinSi
69  Tr=tiempoSol;
70  Np=nPasos;
71  Imprimir n;
72  Imprimir m;
73  Imprimir alfa;
74  Imprimir dif;
75  Imprimir k;
76  Imprimir val;
77  Imprimir tprom;
78  Imprimir tr;
79  Imprimir np;
80  Imprimir gantt;
81  FIN
```

4. Modelo teórico

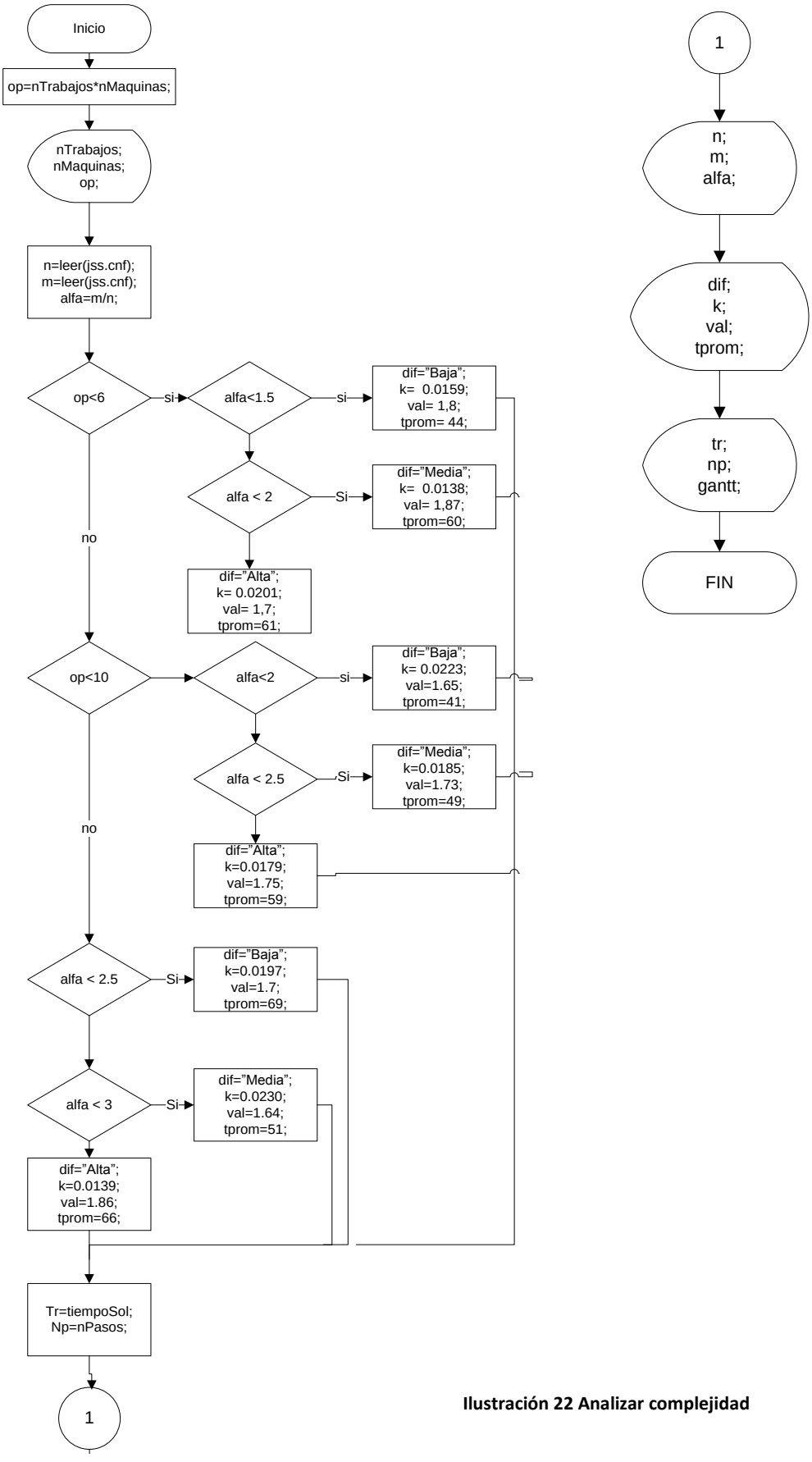


Ilustración 22 Analizar complejidad

4. Modelo teórico

De acuerdo a los diagramas y a los pseudocódigos presentados anteriormente se obtiene una representación de la secuencia de solución del complejo problema de optimización de la programación de tareas. Este método propuesto es independiente de la plataforma y las computadoras involucradas en su resolución.

Los módulos diseñados serán llamados por el solucionador para procesar el problema y obtener así una programación óptima para las tareas. Cumple el algoritmo general de resolución las características que lo hacen eficiente: es preciso (indicando el orden en que se realiza cada paso), definido (permite obtener siempre resultados adecuados) y finito (tiene una cantidad determinada de pasos y termina).

Al haber definido y dividido el problema mediante un diseño descendente es más sencillo comprender la situación y resolverla así de forma óptima, posibilitando modificaciones que afecten a pequeñas partes del modelo y que las pruebas a ejecutar para asegurar la validez del modelo propuesto también sean más simples.

En el siguiente punto se detalla la implementación de la solución propuesta mediante lenguajes de programación, pero el objetivo principal de esta investigación es lo expuesto en este apartado, el desarrollo de un algoritmo para resolver la planificación industrial, haciendo uso del solucionador de tiempo continuo propuesto por Ercsey-Ravasz y Toroczkai.

4. Modelo teórico

4.3 Codificación

En esta etapa se lleva a cabo la codificación del algoritmo propuesto mediante lenguajes de programación. Dada la naturaleza compleja del problema planteado se decide utilizar dos lenguajes de programación, por un lado aprovechar todas las características de un lenguaje orientado a objetos multiplataforma como lo es Java y por otro toda la potencia y capacidad de simulación de un lenguaje de alto nivel para el cálculo numérico, como lo es Matlab, posibilitan la visualización y el desarrollo de funciones, que con las herramientas apropiadas permiten la resolución de ecuaciones diferenciales ordinarias.

Dadas estas característica se opta por la utilización de Matlab para llevar a cabo el subproceso de simulación, utilizando para tal propósito la función propuesta para la solución de ecuaciones ordinarias que utiliza el método Runge Kutta adaptativo de 4-5 orden (ode45). Los demás sub-problemas serán resueltos codificados en el lenguaje Java, aprovechando alguna de sus características como la simpleza, el alto rendimiento, la robustez, portabilidad, y por supuesto las grandes ventajas que aporta la orientación a objetos (como la reutilización de código, la agilización del desarrollo, la capacidad de ampliación, la flexibilidad del sistema generado y la reducción de la complejidad).

Otra de las ventajas que se aprovecha de las disponibles de Java es que provee formas sencillas para diseñar una interfaz amigable que interaccione de forma adecuada con los diferentes procesos necesarios para obtener la solución del problema.

La implementación del algoritmo propuesto en este caso se prueba mediante la creación de una aplicación con interfaz local, que permitirá el acceso del problema mediante una interfaz sencilla elaborada en Java, y que para su solución llamará a una función desarrollada en Matlab que resuelve un sistema de ecuaciones hasta hallar los valores que satisfacen las restricciones del problema, (método propuesto por Ercsey-Ravasz y Toroczkai).

Otro punto importante en la elección de Java, fue la necesidad de utilizar alguna librería para convertir el problema expresado en una tabla a la forma normal conjuntiva que es la que se convierte en el “Input” de la función, por tal motivo en la investigación realizada en el marco teórico puede observarse que en al año 2008 los japoneses Tamura y Taga desarrollaron una librería en lenguaje Java, que realiza la conversión de CSP a CNF para problema de satisfactibilidad booleana y también permite que una vez obtenido el archivo de

solución con los valores de cada variable se realice un mapeo de las mismas y se obtengan los tiempos de inicio de cada operación.

Dados estos motivos se decidió la creación de una interfaz sencilla donde el usuario ingrese el problema de la programación de tareas de un número de máquinas y trabajos dados, y posteriormente se realice una conversión a CSP que se guarda en un archivo, resultando el ingreso para la librería de conversión por orden, que devuelve un nuevo archivo expresado en CNF que será enviado a la función de Matlab, lo procesará creando una matriz y calculando valores aleatorios para las variables enteras, con estos datos realizará la resolución de las ecuaciones diferenciales ordinarias con el soporte de ode45 (un método provisto por Matlab), hasta hallar valores para la función de energía iguales a cero, es decir la satisfacción de las restricciones de orden y prioridad impuestas.

Matlab creará un nuevo archivo que será enviado nuevamente a la librería de codificación por orden, pero esta vez para realizar el proceso inverso, es decir que creará un arreglo que contenga todos los tiempos de inicio de las operaciones.

Desde la interfaz gráfica el usuario podrá ingresar la fecha de comienzo de la producción, y obtener así las fechas para cada operación. En base a estos datos también podrá elaborar un diagrama de Gantt haciendo uso de la librería jFreeChart que posibilita la creación de gráficos complejos, de manera sencilla: dibujando ejes, leyendas y barras.

Por último dado que este sistema es creado en el marco del desarrollo de un proyecto de grado, y mediante este trabajo se requiere verificar su validez, se incorpora un módulo que permite medir su dificultad, presentándole al usuario datos acerca de su codificación, tiempo de resolución, pasos necesarios, y el valor dentro de la “escala de Ritcher” de dificultad, que permitirá identificar si se encuentra dentro de los problemas: “Fáciles”, “Medianos” y “Difíciles”. Además se presenta un gráfico que reflejará el resultado de la simulación realizada mediante ode45 en Matlab.

A continuación se presenta el diagrama de paquetes de la implementación realizada.

4. Modelo teórico

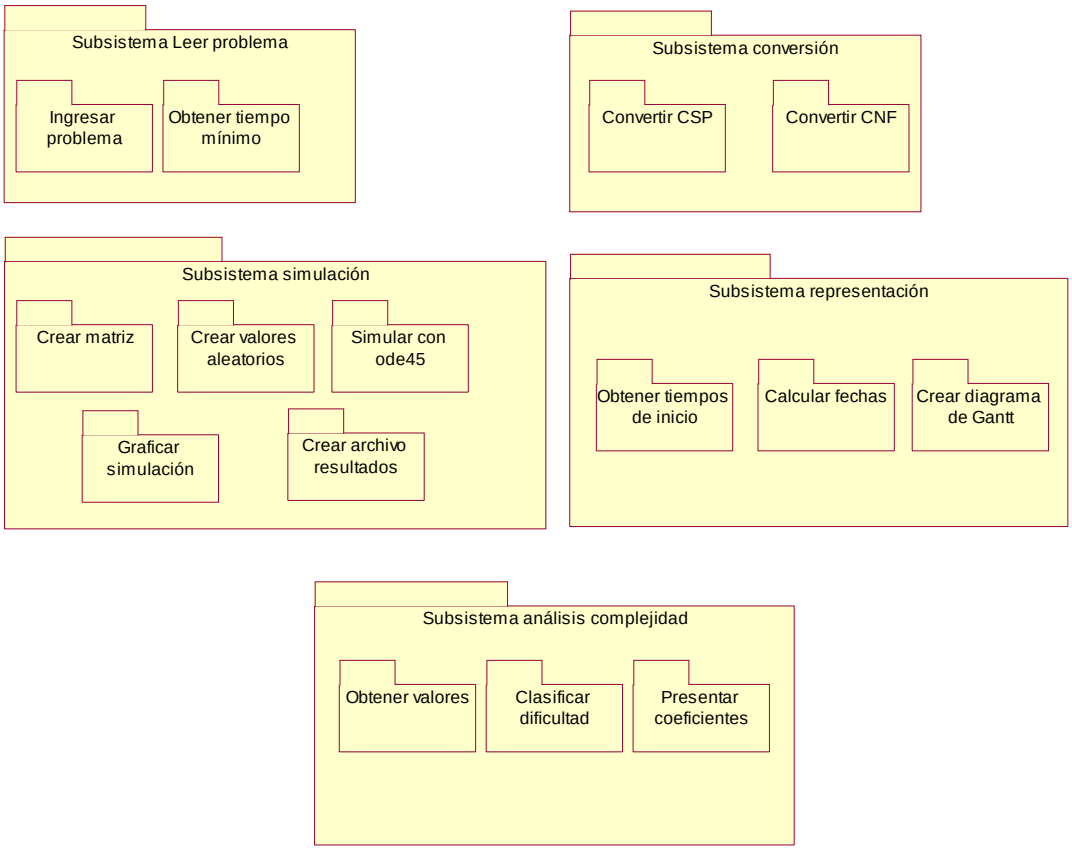


Ilustración 23 Subsistemas implementación

La llamada a la función de Matlab se realiza en Java mediante la librería MatLab Control, la cual gestiona la comunicación entre ambas plataformas de programación, y de este modo se aprovecha el desarrollo del algoritmo de solución del problema de satisfacción de restricciones booleanas, sin necesidad de desarrollar tareas administrativas de bajo nivel tales como la declaración de variables, los tipos de datos o la reserva de memorias. Utilizando Matlab se tiene una iteración rápida en los pasos necesarios, y ejecuta rápidamente operaciones con matrices y vectores densos que surgen en la resolución de este problema. Por otro lado en Java es muy sencillo llevar a cabo la interfaz gráfica adecuada a las necesidades que los usuarios desean satisfacer, destacando además su dinámica a la hora de enlazar clases, su gerencia de memoria, y la posibilidad de crear un código modular reutilizable para posteriormente realizar mantenimiento y mejoras.

Resumiendo la codificación se realizará de la siguiente forma:

4. Modelo teórico

- Lectura del problema: se presenta un frame (Imagen 26) con dos cuadros de texto para ingresar la cantidad de trabajos y máquinas, se genera una tabla en la que el usuario debe ingresar el orden de procesamiento y la duración.

  Posteriormente, se lee la tabla determinando el tiempo mínimo de procesamiento, y el usuario ingresa la cota superior de cálculo. Todo este módulo es desarrollado en lenguaje Java.

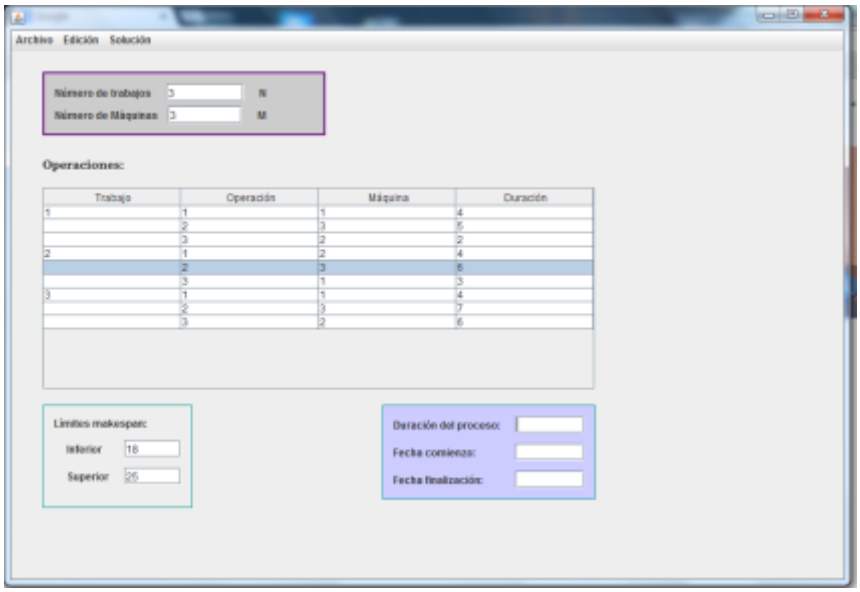


Ilustración 24 Interfaz de lectura del problema

- Conversión: posteriormente se realiza el paso del problema a formato CSP (Imagen 27.b) de acuerdo a las pautas establecidas en la fase de diseño, de este modo se crea un archivo denominado: problema**Ntrabajos-Nmáquinas-fecha-hora**.csp (los valores en negrita se completan según corresponde). Este archivo se transforma en la entrada para la llamada a la librería Sugar que utilizando la conversión por orden crea los archivos problema**Ntrabajos-Nmáquinas-fecha-hora**.map (para la posterior solución), y por último el archivo jss.cnf que será la entrada para la función de Matlab. (Imagen 27 c y d).

  Todo este proceso de conversión, y la librería Sugar utilizada también se realizan en lenguaje JAVA

4. Modelo teórico

Trabajo	Operación	Orden	Duración
1	1	1	4
	2	3	5
	3	2	2
2	1	2	4
	2	3	6
	3	1	3
3	1	1	4
	2	3	7
	3	2	6

a. Problema 3x3

```
; JSS
; 3 3
; 1 4 3 5 2 2
; 2 4 3 6 1 3
; 1 4 3 7 2 6
(int makespan 18 25)
(objective minimize makespan)
(int s_0_0 0 25)
(int s_0_1 0 25)
(int s_0_2 0 25)
(int s_1_0 0 25)
(int s_1_1 0 25)
(int s_1_2 0 25)
(int s_2_0 0 25)
(int s_2_1 0 25)
(int s_2_2 0 25)
(<= (+ s_0_0 4) s_0_1)
(<= (+ s_0_1 5) s_0_2)
(<= (+ s_0_2 2)makespan)
```

b. Conversión a CSP

```
objective minimize makespan
int makespan 1 18..25
int s_0_0 8 0..14
int s_0_1 22 4..18
int s_0_2 36 9..23
int s_1_0 50 0..12
int s_1_1 62 4..16
int s_1_2 74 10..22
int s_2_0 86 0..8
int s_2_1 94 4..12
int s_2_2 102 11..19
```

c. Conversión a MAP

```
p cnf 127 353
-1 2 0
-2 3 0
-3 4 0
-4 5 0
-5 6 0
-6 7 0
-8 9 0
-9 10 0
-10 11 0
-11 12 0
-12 13 0
-13 14 0
-14 15 0
-15 16 0
-16 17 0
-17 18 0
-18 19 0
-19 20 0
-20 21 0
-22 23 0
```

d. Conversión a CNF

Ilustración 25 Conversión

- Conversión: este es el módulo central en la resolución de la planificación, el mismo es desarrollado íntegramente en Matlab, y consta de dos funciones una que realiza la lectura del archivo CNF y lo transforma a matriz, y el otro que crea la matriz cmi, calcula los valores aleatorios y mediante ode45 realiza la simulación hasta llegar a los valores que satisfagan las restricciones. También realiza un gráfico de la simulación efecutada y crea un archivo denominado resultado.txt con los valores positivos o negativos de cada variable en la solución. Devuelve también la cantidad de pasos, el tiempo de resolución, el número de variables y la de restricciones. Esta

4. Modelo teórico

función denominada demo_jssp.m es llamada en Java mediante la librería MatLab Control y al ejecutarse muestra la ventana de comando (Imagen 28).

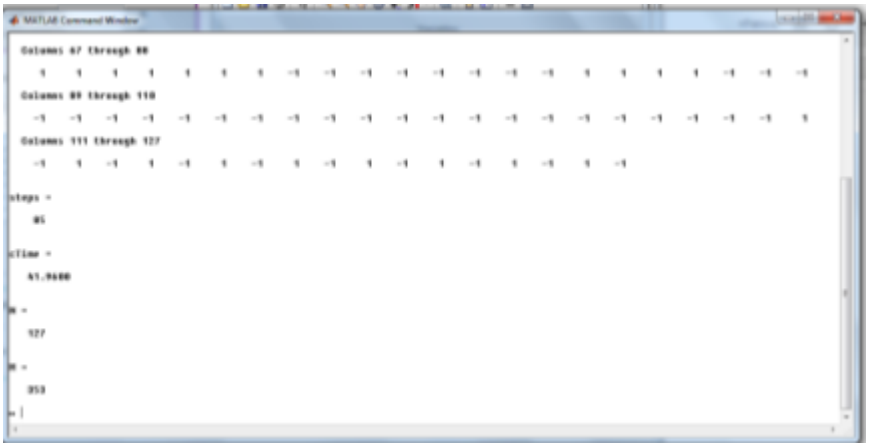


Ilustración 26 Simulación con Matlab

- Presentación de la solución: luego de la obtención de los resultados, se realiza una nueva llamada a la librería Sugar (en Java), en la cual utilizando el archivo .map y el de resultados creado en Matlab se obtienen los tiempos de inicio de cada operación además se calculan adicionándole la duración, el tiempo de finalización, todos estos valores se presentan en la tabla de interfaz (Imagen 29).

  Posteriormente se solicita el ingreso de la fecha de comienzo de la producción y se calcula el resto de las fechas (Imagen 30). En base a los valores obtenidos se realiza un diagrama de Gantt de las distintas operaciones (Imagen 31).

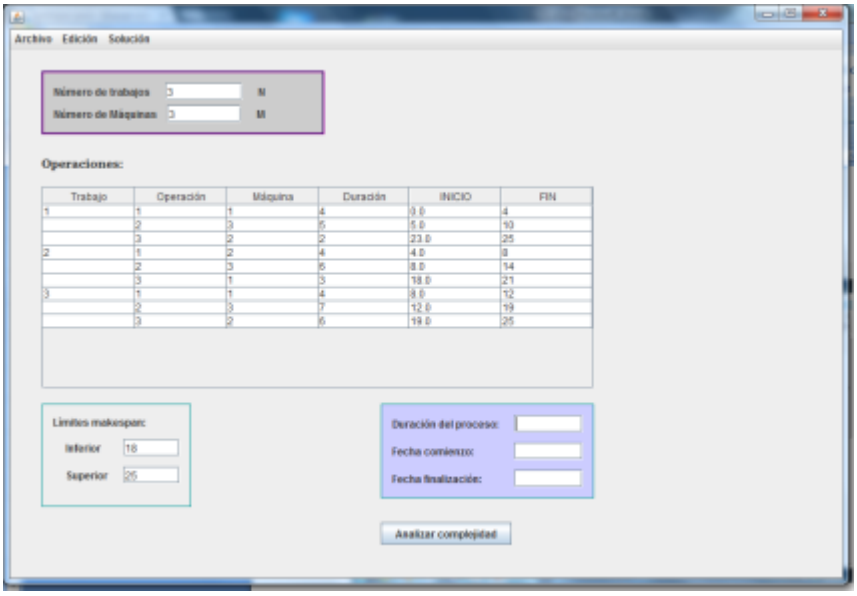


Ilustración 27 Tiempo de inicio y fin

4. Modelo teórico

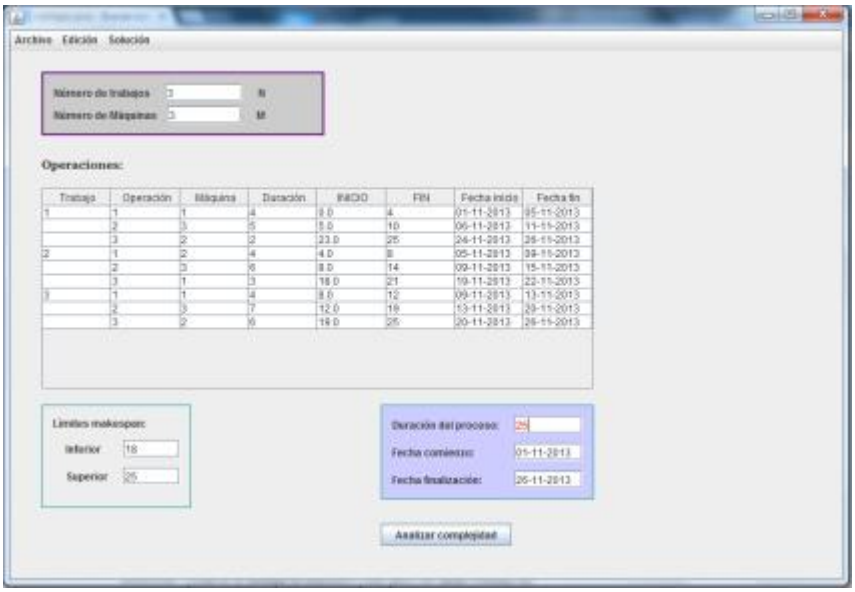


Ilustración 28 Fechas

Ilustración 29 Diagrama de Gantt

- Analizar complejidad: por último el análisis de complejidad, también realizado en Java, se trata de un frame (Imagen 32) que presenta un resumen de distintos valores calculados y obtenidos del procesamiento del problema. Además se incorpora el botón Ver gráfico que permite ver la gráfica de la simulación elaborada por Matlab. (Imagen 33)

4. Modelo teórico

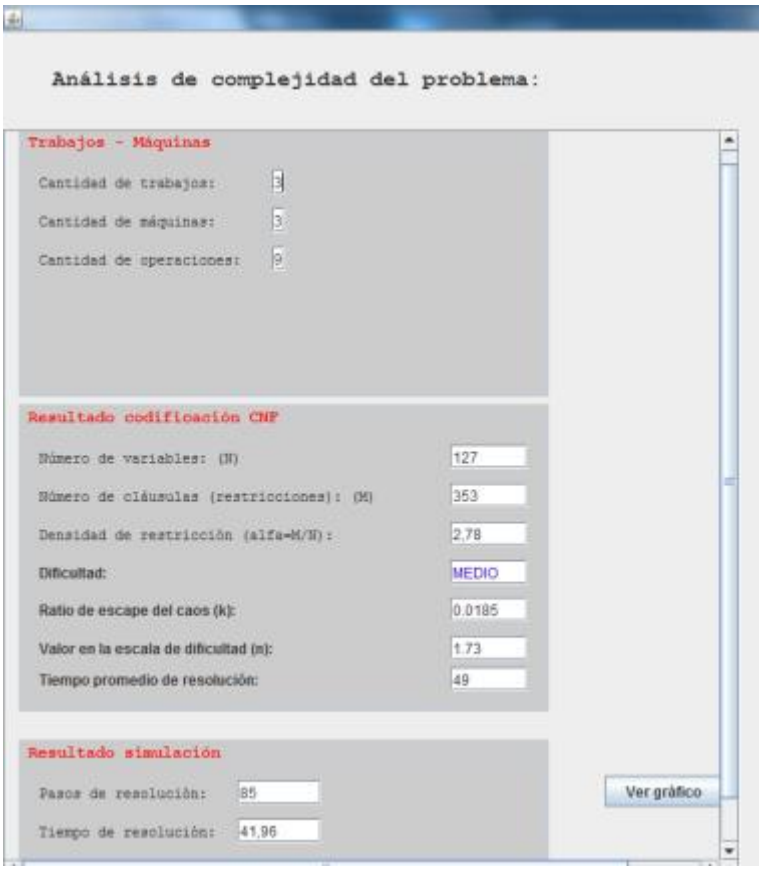


Ilustración 30 Análisis de complejidad

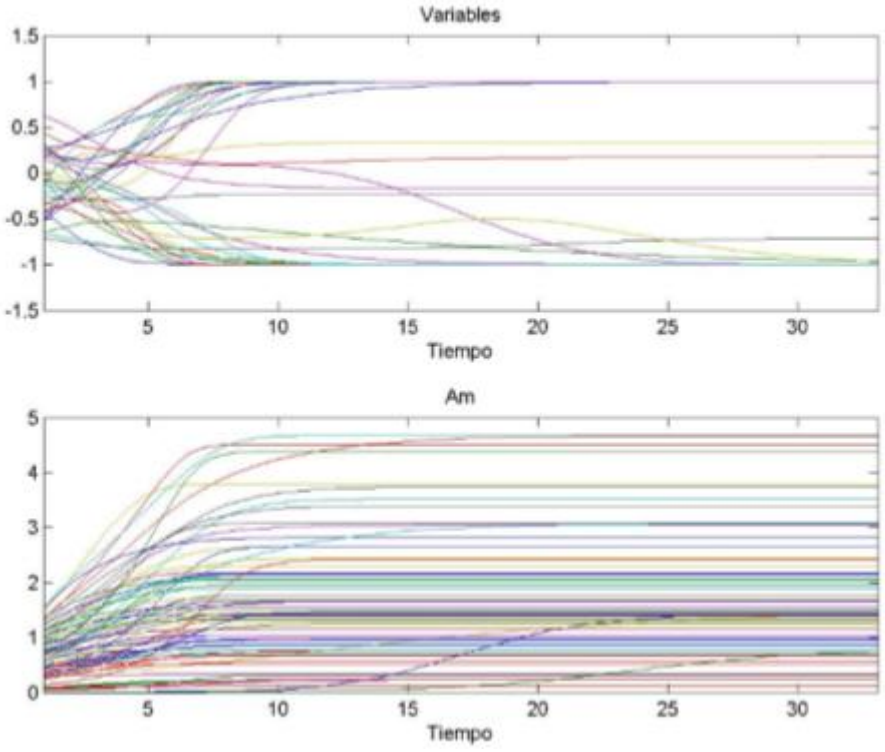


Ilustración 31 Gráfica de simulación

4. Modelo teórico

4.4 Compilación y ejecución

Una vez creado el programa fuente, será necesario cargarlo en memoria y posteriormente almacenarlo en el disco, debido a que se trabaja con el lenguaje java la traducción de código fuente a lenguaje máquina está a cargo de la Java Virtual Machine.

Este proceso se realiza debido a la necesidad de traducir a un lenguaje entendible para la máquina la solución propuesta. El compilador evita que se pueda traducir un programa de código mal escrito y hacer otras verificaciones previas de modo que el código máquina tenga determinadas garantías de que cumple estándares de sintaxis.

Una característica asociada al lenguaje Java es que es independiente del hardware y sistema operativo en que se ejecuta, evitando problemas de compatibilidad, para realizarlo incorpora un paso intermedio que evita que el programa se ejecuta en la máquina directamente, sino que lo hace a través de una máquina virtual que simula Java. De este modo el proceso se amplía en un paso, utilizando un código intermedio en bytecode, que es interpretado por la JVM y da lugar a la ejecución del problema. (Imagen 34)

Para ejecutar el software desarrollado se podrá ejecutar el archivo jssp.jar que es el resultado de la construcción de todas las clases desarrolladas en java. Las llamadas a las librerías y funciones serán gestionadas por el programa.

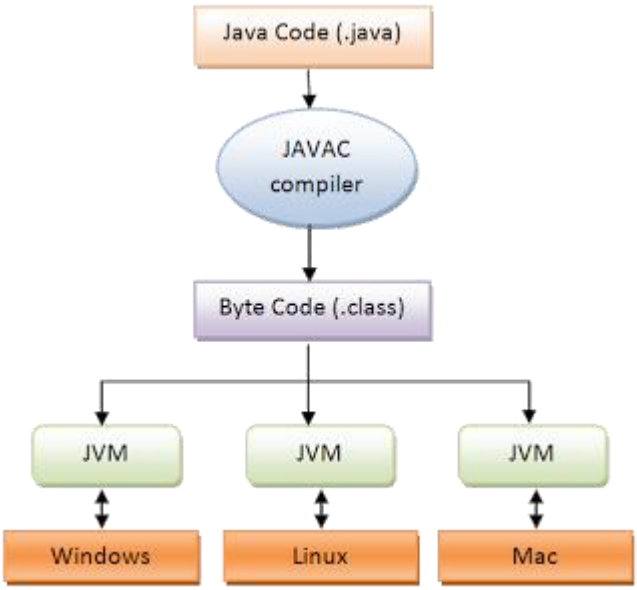


Ilustración 32 Compilación

4. Modelo teórico

4.5 Verificación y depuración

La verificación es el proceso de ejecución con una amplia variedad de datos de entrada (datos de test), que permiten determinar si el programa tiene errores (bugs). Para realizar la verificación es necesario desarrolla una amplia gama de datos, probando los límites del programa y otros que comprueben aspectos especiales.

Este programa fue desarrollado utilizando un diseño modular lo que facilita las pruebas, permitiendo testear por separados los sub-problemas que se presentan al intentar alcanzar la solución.

Por lo tanto se presenta un resumen de las pruebas realizadas a cada uno de los 5 módulos.

4.5.1 Lectura problema:

Datos válidos:

Entrada:

Se ingresan 3 trabajos y 3 máquinas. El orden de operaciones, y su duración:

Trabajo	Operación	Máquina	Duración
1	1	1	4
	2	3	5
	3	2	2
2	1	2	4
	2	3	6
	3	1	3
3	1	1	4
	2	3	7
	3	2	6

Resultado:

- Se obtiene el límite inferior del makespan.

Procedimiento de Prueba

1. Ingresar el número de trabajos y el de máquinas.
2. Presionar Archivo → Nuevo.
3. Completar la tabla con los números de máquina y duración.
4. Presionar Solución → Leer tabla.
5. Debe aparecer En el recuadro límites makespan: Inferior: 18.

4. Modelo teórico

Falta de datos en la tabla principal:

Entrada:

Se ingresan 2 trabajos y 3 máquinas. Se ingresa la siguiente tabla:

Trabajo	Operación	Máquina	Duración
1	1	2	5
	2	1	6
	3	3	8
2	1	3	4
	2	1	
	3	2	

Resultado:

- Se cancela la lectura de la tabla
- Se imprime mensaje de notificación del error.

Procedimiento de Prueba

1. Ingresar el número de trabajos y el de máquinas.
2. Presionar Archivo → Nuevo.
3. Completar la tabla con los números de máquina y duración (de la tabla superior).
4. Presionar Solución → Leer tabla.
5. Aparece el mensaje: "Debe suministrar toda la información solicitada"

Se cancela la lectura de la tabla.

4.5.2 Conversión:

Datos válidos:

Entrada:

Se ingresa el límite de cálculo superior del makespan.

Resultado:

- Se obtienen los archivos problema-*nTrabajos-nMaquinas-fecha-hora*.csp, problema-*nTrabajos-nMaquinas-fecha-hora*.map y jss.cnf.

Condiciones:

Ya se realizó la lectura de un problema con 3 máquinas y 3 trabajos y se obtuvo la cota inferior del makespan.

4. Modelo teórico

Procedimiento de Prueba

1. Ingresar el límite superior de cálculo.
2. Presionar Solución → Convertir a CSP.
3. Abrir la carpeta: C:\JSSP\ y verificar que se creó el archivo problema-3-3-(fecha actual)-(hora actual).csp.
4. Presionar Solución → Convertir a CNF.
5. Se presenta un mensaje con el número de variables y el número de restricciones.
6. En la carpeta C:\JSSP\ verificar que se creó el archivo problema-3-3-(fecha actual)-(hora actual).map.
7. En la carpeta C:\JSSP\matlab\ debe existir el archivo jss.cnf con el número de variables y restricciones especificadas en el mensaje del punto 5.

Problema no satisfacible:

Entrada:

Tabla principal:

Trabajo	Operación	Máquina	Duración
1	1	0	345
	2	2	125
	3	1	232
2	1	2	111
	2	1	231
	3	0	324
3	1	0	211
	2	2	123
	3	1	213

Límite superior makespan: 470

Resultado:

- Se obtiene el mensaje: “El problema no es satisfacible”.

Condiciones:

Ya se realizó la lectura del problema.

Procedimiento de Prueba

1. Ingresar el límite superior de cálculo.
2. Presionar Solución → Convertir a CSP.

4. Modelo teórico

3. Abrir la carpeta: C:\JSSP\ y verificar que se creó el archivo problema-3-3-(fecha actual)-(hora actual).csp.
4. Presionar Solución → Convertir a CNF.
5. Se presenta el mensaje: "El problema no es satisfacible".

4.5.3 Simulación:

Datos válidos:

Entrada:

Se ingresan 3 trabajos y 3 máquinas. El orden de operaciones, y su duración:

Trabajo	Operación	Máquina	Duración
1	1	1	4
	2	3	5
	3	2	2
2	1	2	4
	2	3	6
	3	1	3
3	1	1	4
	2	3	7
	3	2	6

El límite superior de cálculo es: 25.

Resultado:

- Se obtiene el tiempo de inicio de cada operación.

Condiciones:

Ya se realizó la lectura y conversión del problema.

Procedimiento de Prueba

1. Presionar Solución → Resolver.
2. El programa muestra la ventana de comandos de Matlab.
3. Muestra el gráfico de simulación, el tiempo (7.1995) y la cantidad de pasos (154).
4. Verificar la creación del archivo C:\JSSP\matlab\resultados.txt.
5. Se añade la columna tiempo de inicio y de finalización:

Trabajo	Operación	Máquina	Duración	INICIO	FIN
1	1	0	4	0.0	4
	2	2	5	4.0	9
	3	1	2	17.0	19
2	1	1	4	0.0	4
	2	2	6	16.0	22
	3	0	3	22.0	25
3	1	0	4	5.0	9
	2	2	7	9.0	16
	3	1	6	19.0	25

Ilustración 33 Resultados

4. Modelo teórico

Problema demasiado extenso:

Este problema se presenta de acuerdo a la memoria y capacidad de procesamiento del equipo donde se ejecuta el programa.

Considerando que la capacidad de almacenar valores y variables en Matlab depende de la cantidad de memoria disponible, puede suceder que si el resultado de la conversión a CNF arroja una cantidad de variables y restricciones muy grande al realizar el volcado de los datos a una matriz la computadora no logre almacenar todos los valores ocasionando un error que cancela la simulación.

Esta prueba se ejecuta en una Notebook Acer con las siguientes características:

- AMD Dual-Core Processor E-350
- 2 Gb Ram DDR III
- 500 Gb HDD

Entrada:

Se ingresan 10 trabajos y 10 máquinas. El orden de operaciones, y su duración:

1	1	0	2
	2	1	5
	3	2	8
	4	3	9
	5	4	7
	6	5	5
	7	6	6
	8	7	11
	9	8	12
	10	9	1

2	1	1	8
	2	0	9
	3	2	7
	4	3	12
	5	4	1
	6	5	5
	7	7	6
	8	6	9
	9	8	10
	10	9	15

3	1	2	12
	2	1	13
	3	0	14
	4	3	18
	5	4	19
	6	5	20
	7	6	21
	8	7	22
	9	8	11
	10	9	10

4	1	9	9
	2	0	8
	3	1	7
	4	2	6
	5	3	5
	6	4	25
	7	5	65
	8	6	21
	9	7	12
	10	8	27

5	1	9	2
	2	8	3
	3	7	8
	4	6	15
	5	5	21
	6	4	10
	7	3	9
	8	2	7
	9	1	8
	10	0	2

6	1	1	10
	2	2	9
	3	3	18
	4	4	21
	5	5	10
	6	6	12
	7	7	11
	8	8	10
	9	9	9
	10	0	8

7	1	2	7
	2	1	2
	3	3	8
	4	0	5
	5	4	12
	6	5	15
	7	6	7
	8	7	22
	9	8	5
	10	9	9

8	1	9	7
	2	8	6
	3	7	5
	4	6	11
	5	5	12
	6	4	16
	7	3	18
	8	2	14
	9	1	9
	10	0	8

9	1	2	7
	2	1	5
	3	3	15
	4	0	12
	5	4	14
	6	5	7
	7	6	8
	8	7	9
	9	8	25
	10	9	11

10	1	9	12
	2	8	9
	3	7	8
	4	5	7
	5	4	5
	6	6	9
	7	2	10
	8	3	5
	9	1	2
	10	0	1

Resultado:

- Se obtiene un mensaje notificando el problema de memoria y se cancela la solución del problema.

4. Modelo teórico

Condiciones:

Ya se realizó la lectura y conversión del problema.

Procedimiento de Prueba

1. Presionar Solución → Resolver.
2. El programa muestra la ventana de comandos de Matlab.
3. Luego de la lectura del problema, mientras realiza la creación de la matriz emite el error: Out of memory.
4. Se cancela la ejecución.

4.5.4 Presentación resultados:

Datos válidos:

Entrada:

Se ingresa la fecha de comienzo de la producción: 01-10-2013.

Resultado:

- Se obtienen las fechas de inicio y finalización de cada actividad.
- Se muestra la duración del proceso.
- Se presenta la fecha de inicio y finalización de la producción.
- Se exhibe el diagrama de Gantt.

Condiciones:

Ya se realizó la lectura de un problema con 3 máquinas y 3 trabajos y se el tiempo de inicio y fin de cada operación.

Procedimiento de Prueba

1. Presiona Solución → Calcular fechas.
2. Ingresar la fecha: 01-10-2013.
3. Se añaden las columnas: Fecha inicio y Fecha fin. Y se presentan las fechas correspondientes.
4. Se muestra: la duración del proceso, la fecha de comienzo de la producción y la fecha de finalización.
5. Ir a Solución → Graficar
6. Se exhibe el diagrama de Gantt.

4. Modelo teórico

Fecha ingresada de forma incorrecta:

Entrada:

Se ingresa la fecha de comienzo de la producción con el siguiente formato: 01/10/2013.

Resultado:

- Se recibe mensaje notificando que el formato de ingreso de la fecha es incorrecto.

Condiciones:

Ya se realizó la lectura de un problema con 3 máquinas y 3 trabajos y se el tiempo de inicio y fin de cada operación.

Procedimiento de Prueba

1. Presiona Solución → Calcular fechas.
2. Ingresar la fecha: 01/10/2013.
3. Se recibe mensaje notificando que el formato de ingreso es incorrecto.

4.5.5 Análisis complejidad:

Datos válidos:

Entrada:

Para este análisis el usuario ya ingresó los datos correspondientes.

Resultado:

- Se obtienen:
    - Cantidad de máquinas
    - Cantidad de trabajos
    - Cantidad de operaciones
    - Número de variables
    - Número de cláusulas
    - Densidad de restricción
    - Dificultad
    - Ratio de escape del caos (k)
    - Valor en la escala de dificultad n
    - Tiempo promedio de resolución
    - Pasos de resolución
    - Tiempo de resolución
    - Gráfico de simulación

4. Modelo teórico

Condiciones:

Ya se presentaron los resultados y se calcularon las fechas correspondientes.

Procedimiento de Prueba

1. Presionar el botón analizar complejidad
2. Se presentan los valores: cantidad de máquinas, cantidad de trabajos, cantidad de operaciones, número de variables, número de cláusulas, densidad de restricción, dificultad, ratio de escape del caos (k), valor en la escala de dificultad n, tiempo promedio de resolución, pasos de resolución, tiempo de resolución y Gráfico de simulación

No se obtuvo resultado luego de la simulación

Este problema se presenta cuando no arrojó resultados el proceso de simulación, por lo tanto los valores de análisis de complejidad no se obtuvieron

Entrada:

Para este análisis el usuario ya ingresó los datos correspondientes.

Resultado:

No se presenta el análisis de complejidad.

Condiciones:

La simulación no se realizó correctamente.

Procedimiento de Prueba

1. Luego de recibir un error en el proceso de simulación (Por ejemplo: Fuera de memoria). Se desea realizar el análisis.
2. El botón analizar complejidad no se presenta en la pantalla.
3. No se muestra el análisis.

4.6 Documentación y mantenimiento

4.6.1 Descripción general del programa

El software desarrollado soluciona el problema de la planificación de operaciones en el marco de la actividad industrial. Con una interfaz sencilla el usuario ingresa la cantidad de trabajos y de máquinas. Posteriormente ingresa el orden de ejecución de cada actividad dentro del trabajo y su duración.

4. Modelo teórico

El software realiza una lectura del problema y sumando los tiempos de procesamiento de cada trabajo y de cada máquina determina el tiempo mínimo de makespan. El usuario ingresa el tiempo máximo (resulta así el lapso en el que se realizará la programación).

Posteriormente se realizan conversiones del problema a formato CSP, MAP y CNF que se almacenan en la carpeta C:\JSSP.

El software realiza una llamada a la función demo_jssp de Matlab y este realiza una lectura del archivo jss.cnf, en base al número de variable y de restricciones elabora una matriz de acuerdo a si un valor, está o no presente o si está en forma negada.

Luego calcula dos valores aleatorios uno denominado variable de giro que se asocia a cada variable booleana y que toma valores entre -1 y 1 , y el otro es una variable de ajuste para la función energía, cuyo valor es positivo (entre 0 y 1).

Utilizando como entrada la matriz y los valores aleatorios realiza una simulación mediante la función ode45 de matlab resolviendo dos ecuaciones diferenciales ordinarias, y continúa hasta hallar la solución, es decir cuando la función de energía alcanza el 0 en todos los valores.

Una vez hallada esta solución se realiza una gráfica de la solución y se crea el archivo resultado.txt, que indica los valores (-1 o 1) de cada variable, al compararlos con el archivo de mapeo, se obtienen los tiempos de inicio de cada operación, que son expuestos en la tabla.

Luego si el usuario lo desea, ingresa la fecha de comienzo de la producción, y el sistema calculará las fechas de inicio y finalización de cada operación, y basado en estos datos elaborará un diagrama de Gantt de la producción. Permitiendo así la observación de desvíos y el control sencillo de lo planificado.

Por último el usuario puede analizar la dificultad asociada a la resolución de la planificación, observando la cantidad de variables y restricciones relacionadas al problema, el coeficiente de restricción (alfa=M/N), su tiempo de resolución y la cantidad de pasos que tomó. Como así también ver el valor en la escala de dificultad del problema que se calcula de acuerdo a un valor denominado ratio de escape, que se obtuvo luego de la resolución de varios problemas de Job Shop Scheduling.

4. Modelo teórico

4.6.2 Flujograma

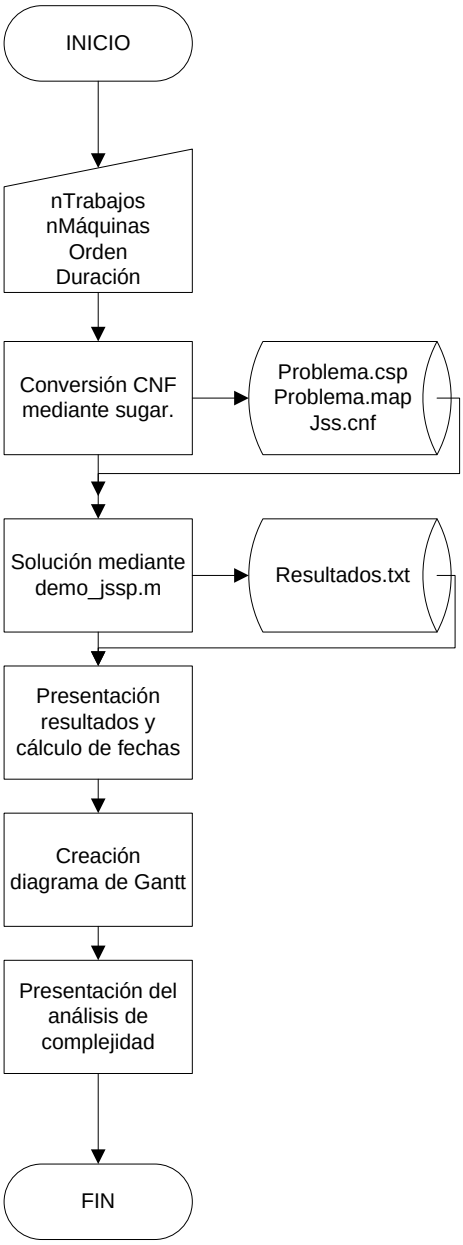


Ilustración 34 Flujograma de la solución

4.6.3 Pantallas

Como ya se describió en el punto de implementación, la interfaz del programa es muy sencilla debido a que permite que el usuario realice un uso intuitivo del mismo, ingresando sólo los datos necesarios para obtener en poco tiempo una producción bien organizada que cumpla con las restricciones pautadas de forma eficiente y de este modo exista un buen aprovechamiento del tiempo y los recursos de la organización.

4. Modelo teórico

A continuación se presentan algunas de las pantallas del programa:

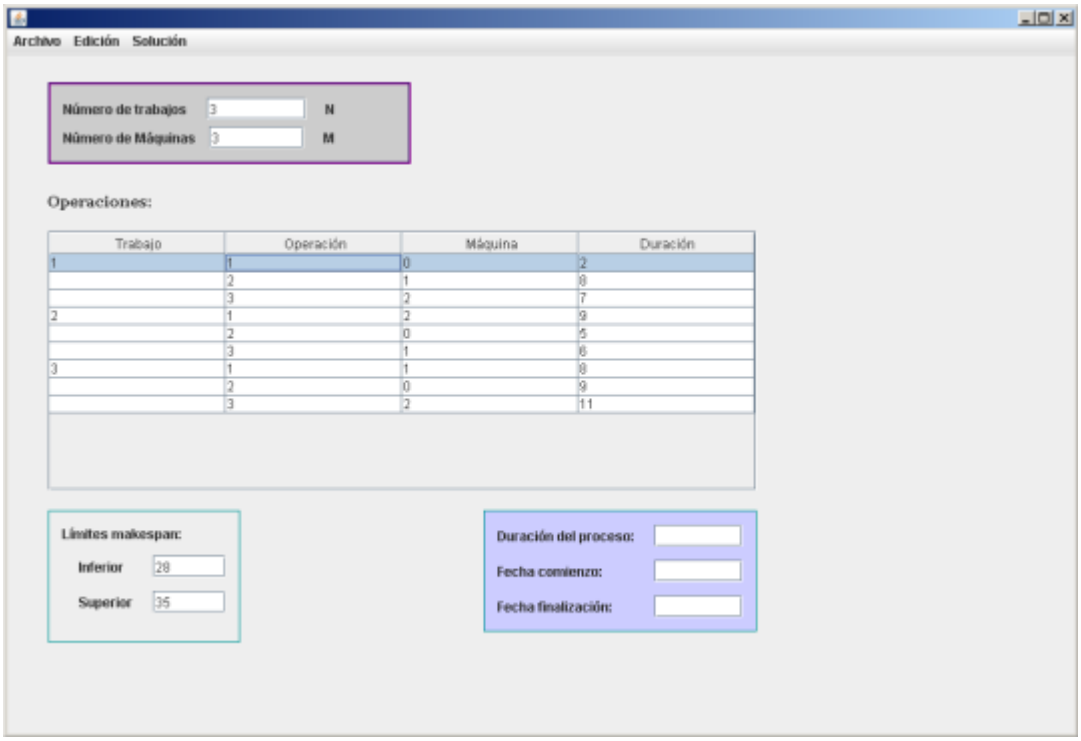


Ilustración 35 Interfaz previa a la solución

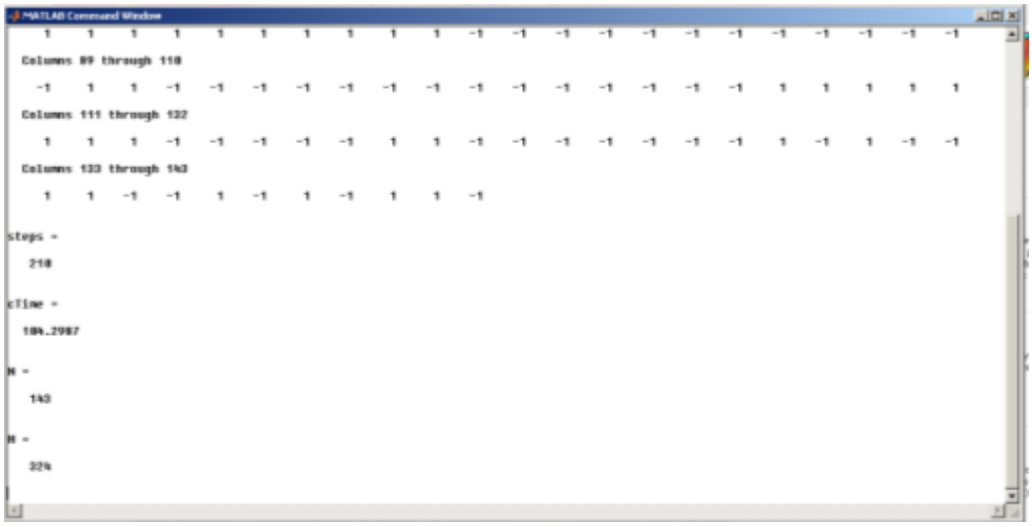
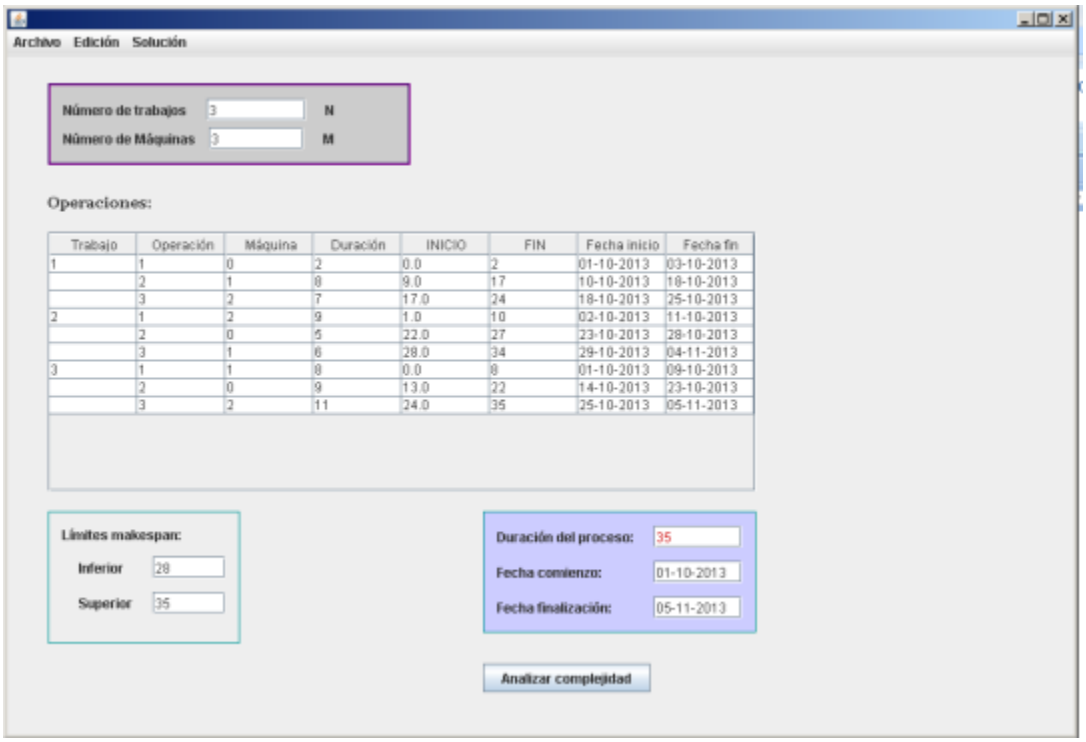


Ilustración 36 Ventana de comandos Matlab

4. Modelo teórico



Trabajo	Operación	Máquina	Duración	INICIO	FIN	Fecha inicio	Fecha fin
1	1	0	2	0.0	2	01-10-2013	03-10-2013
	2	1	8	9.0	17	10-10-2013	18-10-2013
	3	2	7	17.0	24	18-10-2013	25-10-2013
2	1	2	9	1.0	10	02-10-2013	11-10-2013
	2	0	5	22.0	27	23-10-2013	28-10-2013
	3	1	6	28.0	34	29-10-2013	04-11-2013
3	1	1	8	0.0	8	01-10-2013	09-10-2013
	2	0	9	13.0	22	14-10-2013	23-10-2013
	3	2	11	24.0	35	25-10-2013	05-11-2013

Ilustración 37 Resultados

Ilustración 38 Diagrama de Gantt

4. Modelo teórico

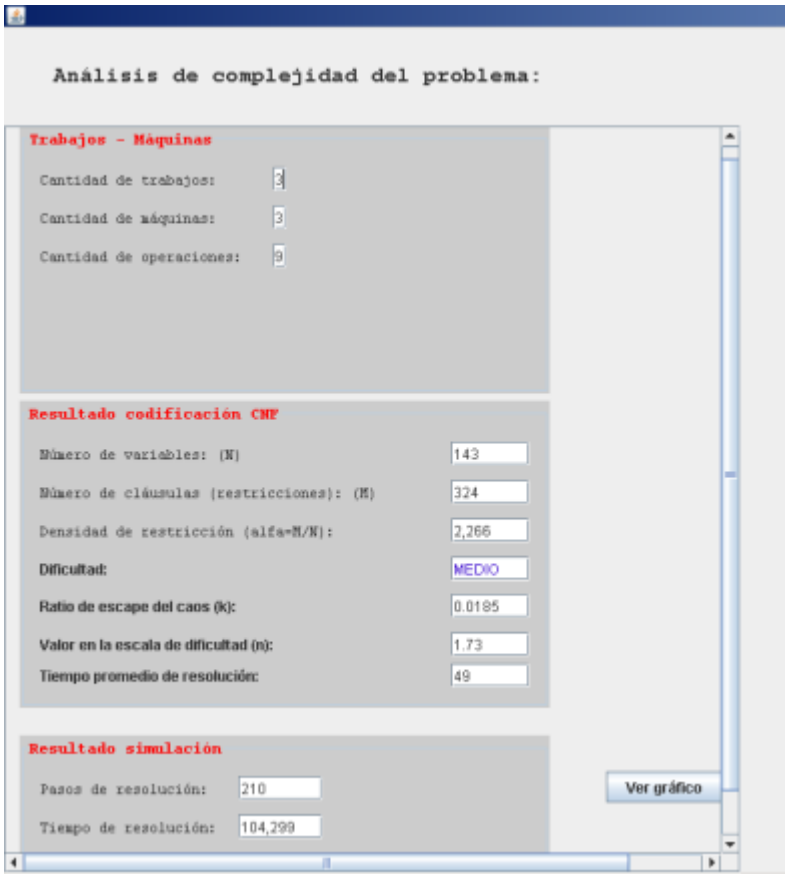


Ilustración 39 Análisis de complejidad

4.6.4 Resguardo

La planificación obtenida puede resguardarse haciendo clic en Edición → Copiar tabla.

De este modo la tabla de resultados es copiada al archivo java.xls.

Y puede ser restaurado haciendo clic en Edición → Pegar tabla.

Además los archivos que surgen de la resolución como:

- problema-*nTrabajos-nMaquinas-Fecha-Hora*.csp

- problema-*nTrabajos-nMaquinas-Fecha-Hora*.map

- jss.cnf

- resultados.txt

Se almacenan al igual que el archivo java.xls en la carpeta C:\JSSP. Los mismos pueden leerse mediante un Editor de texto como bloc de notas o Notepad ++.

4. Modelo teórico

4.6.5 Frecuencia

En el caso de la programación de tareas las actividades de respaldo pueden realizarse cada vez que calcula una programación.

Y en cuanto al mantenimiento del sistema como así también las actualizaciones al mismo pueden ser desarrolladas en módulos. Modificando así sólo las partes que tengan incumbencia en la reforma.

De este modo podrá ingresarse de otro modo el problema, o traducirse mediante otra librería (siempre que se obtenga un archivo CNF), la función de Matlab podrá mejorarse, incorporando por ejemplo un mejor tratamiento de las matrices grandes, o utilizando otro método para resolver las ecuaciones diferenciales. La presentación de los resultados también podrá ser de manera diferente, modificando el módulo correspondiente y por último el análisis de complejidad podrá incorporar otros datos relevantes, modificar valores actuales o eliminarse en el caso de que los usuarios no necesiten disponer de esta información.

5 Concreción del modelo

Luego de la definición de la problemática, y el planteo de la necesidad de elaborar un nuevo modelo para solucionar la planificación industrial se propuso la utilización de un algoritmo que soluciona problemas de satisfactibilidad booleana basado en principios de la teoría del caos.

Se realizó analizó, diseño e implementó un software que realice una conversión del Job Shop Scheduling, lo resuelva mediante el solucionador de tiempo continuo y represente la solución de forma sencilla. Además de presentar un análisis de la complejidad del proceso resolutorio, y pruebas que comprobaron la corrección del modelo propuesto.

A continuación se presenta un análisis de prefactibilidad para determinar la posibilidad real del implementarlo técnica, económica y operativamente.

Luego se analizarán las ventajas que surjan de la puesta en marcha de la solución propuesta en comparación con otros modelos vigentes, y se analizarán los resultados obtenidos.

5.1 Análisis de prefactibilidad

5.1.1 Factibilidad técnica

De acuerdo a lo planteado durante la fase de diseño e implementación de la solución propuesta, se expone claramente que a medida que la cantidad de máquinas y trabajos aumenta, la cantidad de variables y restricciones en forma Normal Conjuntiva crece exponencialmente, requiriendo no sólo mayor tiempo de procesamiento, sino que también genera un agotamiento de los recursos de memoria por parte del software Matlab que es el que realiza la simulación y la solución de las ecuaciones diferenciales ordinarias.

A continuación (Tabla 14) se presentan algunos resultados de la conversión de varios problemas de Job Shop Scheduling con la librería Sugar, en los que se observa cuánto crece la cantidad de variables y restricciones, cuando lo hacen la cantidad máquinas y trabajos.

Problema (Trabajos- Máquinas)	Dificultad	Variables	Restricciones
3-3	FACIL	96	127
3-3	MEDIO	116	265
3-3	DIFICIL	2012	4215
6-6	FACIL	560	2395
6-6	MEDIO	1306	4807
6-6	DIFICIL	11134	58598
10-10	FACIL	3479	26333
10-10	MEDIO	10964	91144
10-10	DIFICIL	44131	353362

Tabla 13 Resultados de conversión

Como se observa, a partir de la conversión difícil del problema 6 x 6 el tamaño de la matriz C necesaria para la resolución del problema será de: [11134; 58598], es decir que en memoria más de 600 millones de variables correspondientes a las x_{ij} que valdrán -1, 0 o 1.

Un problema técnico que podría presentarse es que Matlab agote la memoria disponible para almacenar estos valores, esto se produce según sean las características del equipo donde se ejecuta y su versión (32 – 64 bits). Además será conveniente realizar una configuración en el archivo boot.ini de Windows para reservar más memoria para Matlab.

En la página de Matlab, se enuncian algunas posibilidades para gestionar matrices muy grandes[28]. De acuerdo a este sitio el límite de procesamiento de acuerdo al Sistema Operativo será:

Sistema operativo	Límite del proceso
32-bit Microsoft Windows XP, Windows Vista, Windows 7	2 GB
32-bit Windows XP con 3 GB boot.ini modificado o 32-bit Windows Vista o Windows 7 con increaseuserva	3 GB
32-bit Linux	~3 GB
64-bit Windows o Linux corriendo 32-bit MATLAB	≤ 4 GB
64-bit Windows, Apple Macintosh OS X, o Linux corriendo 64-bit MATLAB	8 TB

Tabla 14 Límites de procesamiento

De acuerdo a lo expuesto, para lograr procesar problemas de 10 máquinas y 10 trabajos, sin limitaciones de memoria por parte del software Matlab será importante ejecutar la aplicación en un equipo con un sistema operativo de 64 bits y una memoria RAM superior a 4 Gb.

[28] http://www.mathworks.es/es/help/matlab/matlab_prog/resolving-out-of-memory-errors.html

Es decir que para la resolución de problemas grandes los requisitos recomendados serán:

Procesador: Intel Core i7/AMD Bulldozer

Memoria: 4 Gb o superior

Disco duro: 2 Gb de espacio libre.

El software con el que debe contar el equipo:

- Java Virtual Machines

- Matlab 64 bits

- Windows/Linux versión 64 bits.

Para problemas más pequeños como los 3x3 un sistema operativo de 32 bits, una versión 32 bits de Matlab y una memoria ram de 2 Gb es suficiente para recibir en tiempo razonable la solución al problema.

5.1.2 Factibilidad económica

Para analizar los beneficios reales de la implementación se presentará a continuación un análisis costo beneficio del modelo propuesto.

La técnica del análisis coste/beneficio tiene como objetivo fundamental proporcionar una medida de los costes en que se incurre en la realización de un proyecto y comparar dicha previsión de costes con los beneficios esperados de la realización de dicho proyecto.

En general los costes suelen ser cuantificables y estimables en unidades económicas, no así los beneficios, los cuales pueden ser tangibles o intangibles. En el siguiente análisis se consideran aquellos aspectos tangibles, es decir, cuantificables en valores como dinero, tiempo, etc., e intangibles, es decir, no ponderables de una forma objetiva.

- **Costos**

Precio del software: de acuerdo a estimaciones que se presentan en el anexo, se estima que el precio de implementar la solución será de unos $2500.

Costos de capacitación: debido a la sencillez de utilización, y la interfaz amigable, será muy fácil capacitar al usuario, en el uso del software. Con lo cual no se consideran relevantes estos costos.

Hardware: como se especificó en el análisis de factibilidad técnica las necesidades de hardware serán muy diferentes para empresas con poca cantidad de operaciones, que podrán ejecutarlo en cualquier equipo de oficina, que para aquellas donde el número de operaciones sea superior a 30. En este caso el equipo a utilizar deberá ser bastante potente, no sólo por la capacidad de cálculo sino para la memoria que utilice la simulación de la solución del problema. Un equipo con estas características ronda en la actualidad en los $5000.

Software: el equipo en el que se ejecute el sistema deberá contar con la JVM instalada además de Matlab, existen versiones Open Source que realizan funciones similares e incluyen métodos de resolución de ecuaciones diferenciales ordinarias como el ODE45. Inclusive modificando el módulo de resolución podría optarse por métodos que realicen este proceso en otros lenguajes.

Pero tal como está programado el sistema será necesario contar con una licencia de Matlab para ejecutar la llamada al mismo mediante la ventana de comandos. El costo aproximado es de: u$s89.

De acuerdo al análisis realizado los costos irán en la alternativa más barata de unos $3500 a unos $10000 (adquiriendo el hardware de procesamiento más avanzando).

- **Beneficios:**

Se obtendrá una mejora considerable en el proceso de planificación de la producción reduciendo tiempos muertos, mejorando la capacidad de control, aumentando la productividad, facilitando la toma de decisiones, y el cálculo de las necesidades de material (MRP), resultará más sencillo por parte del encargado de elaborar el plan de producción establecer planes ante contingencias, y generará mayor flexibilidad empresarial.

Se generarán mayores ingresos y captación de clientes, al cumplir con los mismos en los tiempos pautados.

Se reducirá el pago de horas extras a los empleados y se aprovecharán todos los recursos con los que cuenta la empresa. Como así también los honorarios a personal externo de la empresa encargada del control y la elaboración de planes.

Favorecerá la explotación de nuevos métodos de producción, mejorando la gestión, como el Just in Time, la rotación de stock.

5. Concreción del modelo

Resulta muy complejo estimar cuál será el impacto financiero de estos beneficios, pero se calcula que las mejoras en cuanto a la reducción de horas extras, y el aumento de productividad, como así también la mejor gestión de los recursos, la administración de los materiales, y del control productivo podrían generar mejoras en los ingresos cercanas al 5%.

Considerando una PyME con una producción mensual que genera ingresos por: $30000 obtendrá: $1500 más cada mes, es decir que al cabo de unos 3 meses la inversión podrá ser cubierta.

5.1.3 Factibilidad operativa

En este análisis se busca determinar la posibilidad de la implementación de la solución diseñada, satisfaciendo los requerimientos planteados al inicio.

En este punto se analizará:

- La simpleza de utilización

- La adopción de mismo por parte de los usuarios

- La probabilidad de obsolescencia.

En cuanto a la simpleza, la interfaz ha sido de forma que el usuario tenga la posibilidad de ingresar unos pocos variables conocidos y obtener una programación adecuada de la tarea, como así también un diagrama de Gantt de la producción para facilitar el control. Será necesario remarcar las ventajas que acarreará la incorporación de la solución provista en el proceso de toma de decisiones y en el aumento de la productividad.

El encargado de producción (usuario del sistema) conoce lo complejo de la elaboración de un plan productivo, en empresas de Mediana dimensión muchas veces no se cuenta con el know how para desarrollar esta actividad y se realiza de forma intuitiva generando dificultades posteriores, como horas extras o tiempos muertos. Con lo cual se supone que al implementar la solución todos los involucrados se encontrarán plenamente satisfechos al lograr automatizar una tarea tan engorrosa y crítica para la organización.

En puntos posteriores se realiza un análisis comparativo entre este nuevo método y otros existentes, tal vez este solucionador sea remplazado por uno más eficiente, que utilice la teoría del caos (al igual que este), o tal vez métodos heurísticos o de inteligencia artificial. Pero lo relevante en esta investigación fue el abordaje del tema de Job Shop Scheduling mediante un problema de Satisfacción de Restricciones, que se resuelve utilizando un solu-

cionador de tiempo continuo que persigue encontrar los momentos en el que caos sea igual a 0 para todas las variables y así hallar un solución factible. Con lo cual tal vez no sea ni el método más eficiente, tampoco el más sencillo y rápido de ejecutar, pero sí es una solución aplicable a empresas que actualmente no cuentan con una planificación automatizada y es otra alternativa para intentar solucionar este problema del tipo NP-Completo que desde hace más de 50 años se ha ido perfeccionando y buscando nuevas propuestas de solución.

Sin dudas el impacto de la implementación del modelo será positivo en la organización generando beneficios operativos, financieros y logrando marcar claras diferencias entre una organización que utiliza métodos de gestión y una que improvisa día a día.

De acuerdo a todo lo expuesto se deja constancia que es totalmente factible desarrollar y operar en una empresa tanto PyME como de mayor tamaño el sistema de Optimización de planificación propuesto.

5.2 Prueba de hipótesis

En la introducción y durante el análisis de la situación planteada quedó claramente definido que el objetivo de esta investigación fue plantear un nuevo método de optimización de la planificación de la producción haciendo uso de un algoritmo que soluciona mediante el planteo de un sistema dinámico determinista un problema de satisfacción de restricciones.
El algoritmo a su vez utiliza características de la teoría del caos (conceptos como la Energía, las variables de giro, entre otros) para buscar determinar la solución que genera el valor de todas las variables cumpliendo con las restricciones impuestas.
Las variables y restricciones se obtienen luego de un proceso de conversión del problema expresado en forma sencilla. Y posterior a la simulación realizada con una función de Matlab denominada ode45 (que permite resolver ecuaciones diferenciales ordinarias con un método adaptativo), permite la presentación de los tiempos de inicio y finalización de cada actividad.
Además se ha incorporado un módulo que permite analizar la complejidad del problema planteado, por un lado se analiza el resultado de la conversión a forma normal conjuntiva poniendo atención en el coeficiente denominado densidad de restricción, que no es más que el cociente entre restricciones y variables. De allí surge una escala de dificultad dependiendo la cantidad de máquinas y trabajos involucrados con el problema. (Tabla 16)

V=Trabajos * Máquinas	Valor densidad de restricción	Dificultad
V<6	α<1.5	Baja
	1.5<α<2	Media
	2<α	Alta
6<V<10	α<2	Baja
	2<α<2.5	Media
	2.5<α	Alta
10<V	α<2.5	Baja
	2.5<α<3	Media
	3<α	Alta

Tabla 15 Escala dificultad

Otro valor relevante a la hora de determinar la dificultad es una medida que se obtuvo luego de resolver más de 100 problemas. En la que se intentó determinar la "dureza" de resolución enmarcando el problema dentro de una "escala de Richter", denominada de esta manera ya que lo que se mide es la energía demandada para alcanzar la solución. Esta medida es posible de medir debido a que la trayectoria caótica de los sistemas hiperbólicos decae exponencialmente. En base a esto lo que se calcula *p(t)* que representa la probabilidad de que el sistema no encuentre solución el tiempo t. En base a este valor se puede obtener el ratio de escape del sistema denominado k, que viene dado por la siguiente fórmula: $p(t) \sim e^{-kt}$.

Usando los valores obtenidos de la resolución de distintos problemas se ajusta la función e^{-kt} y así se puede determinar el ratio de escape para cuantificar la dificultad del problema.

5. Concreción del modelo

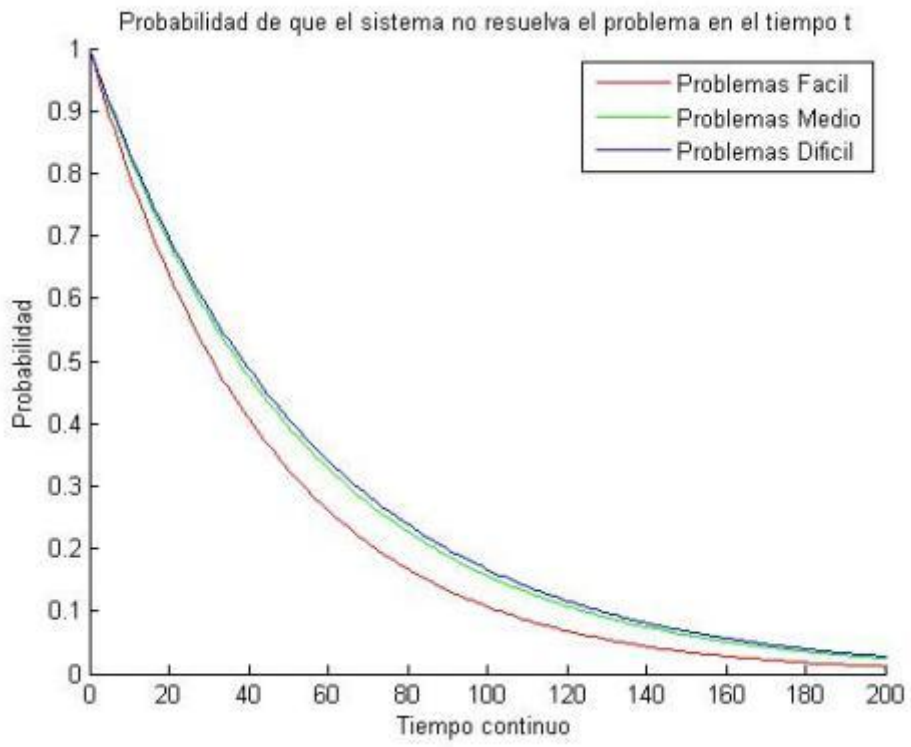


Ilustración 40 Probabilidades p(t) para problemas 3 x 3

A continuación se presenta la tabla con el ratio de escapa y el valor en la escala de dificultad ($-\log_{10}k$):

Dificultad	**k**	**$-\log_{10}k$**
Fácil	0.0223	1.65
Medio	0.0185	1.73
Difícil	0.0179	1.75

Tabla 16 Ratios de escape y nivel de dificultad

En conclusión en base al ratio de escape k de el sistema dinámico utilizado para resolver la clase dada de un problema, junto con su logaritmo negativo índice el nivel de dificultad dentro de una escala construida a base de resolver varios problemas de la misma clase.

De la resolución de los problemas para la obtención de la escala de dificultad, se analizó también la validez de la solución propuesta, hallando en todos ellos soluciones veraces que satisfacían todas las restricciones.

El paradigma utilizado para llevar a cabo la investigación es el empírico-analítico, debido a que se busca corroborar la hipótesis considerando la vía hipotético-deductiva.

La hipótesis establecida era la factibilidad de utilizar un algoritmo dinámico determinista basado en la teoría del caos para llevar a cabo la optimización y automatización de la planificación industrial.

5. Concreción del modelo

Para concluir con la corroboración de la hipótesis se llevó a cabo un proceso investigativo especificado en el marco teórico, donde se analizaron los problemas de alta dificultad computacional (NP completos), los modelos de planificación, los problemas de satisfacción de restricciones, y los actuales métodos de resolución. Para posteriormente con esta información elaborar un algoritmo que enmarque todo el proceso de planificación, desde el planteo del problema, hasta la presentación de la solución, incorporando un módulo que permite el análisis de dificultad de la situación planteada.

Luego de la creación e implementación del modelo en lenguaje computacional (Java y Matlab) se llevaron adelante más de 100 ejecuciones de la solución propuesta obteniendo siempre resultados positivos. Presentándose algunos problemas debido a la escasez de memoria del equipo utilizado cuando la magnitud del programa generaba luego de su conversión en forma normal conjuntivo un número de variables y restricciones inmanejables para Matlab.

Se verificó que los tiempos de resolución dependen de la capacidad de cálculo del equipamiento informático utilizado, generando muchas veces tiempos de resolución excesivos comparados con otros modelos existentes, pero lo que cabe destacar es el aporte teórico que significa el planteo del problema de Job Shop Scheduling como un problema de Satisfacción Booleana (realizado por Crawford y Baker), la codificación del mismo utilizando la moderna codificación por orden planteada en 2008 por Tamura, para posteriormente hayar la solución mediante un nuevo método algorítmico publicado en 2012 por Ercsey-Ravasz y Torozckai que utiliza el caos determinista para hallar una solución y medir la dificultad de cálculo.

Los sistemas análogos continuos que se introdujeron para la resolución de problemas del tipo NP-Completo generan interés teórico debido al nuevo punto de vista con el que se estudian sus propiedades, el sistema dinámico propuesto por Ercsey-Ravasz y Toroczkai para la resolución de Sudokus analiza la configuración de números de partida en el tablero y evoluciona desde una condición inicial aleatoria, en la presente investigación el punto de partida es una matriz con el orden de procesamiento de cada máquina dentro de un trabajo y su correspondiente duración. Es decir que ambos persiguen la búsqueda la solución que de existir y ser única se acaba alcanzando.

Trabajar y caracterizar los sistemas caóticos es muy complejo debido a la irregularidad del comportamiento, por lo que generalmente se introducen parámetros que permiten caracterizarlo. En base a lo propuesto por Ercesey-Ravasz y Toroczkai, se utiliza un parámetro

denominado k (ratio de escape) que permite medir la duración del caos transitorio, que al alcanzar la convergencia desaparece dando lugar a la solución. La probabilidad de que no se obtenga solución en un tiempo dado (p(t)), sigue una distribución exponencial e^{-kt}.

5.3 Resultados

Luego de la investigación, la elaboración del algoritmo y el modelo de resolución, se consiguió elaborar un sistema que planifique y optimice el problema del Job Shop Scheduling, expresado como k-SAT en Forma Normal Conjuntiva.

Se lograron determinar los pasos para convertir el problema planteado en una interfaz sencilla, mediante una librería de Java denominada Sugar, luego se elaboró el archivo de resolución aprovechando las ventajas de procesamiento y la función ode45 que posibilita obtener las soluciones de un sistema de ecuaciones mediante el método Runge-Kutta de 4^{to} – 5^{to} orden adaptativo. Se presentaron las soluciones y se calcularon las fechas correspondientes, como así también se elaboró un diagrama de Gantt que posibilita el control de lo calculado.

Por último se realizaron más de 100 ejecuciones de la solución propuesta, hasta obtener valores de probabilidad de resolución y así calcular coeficientes como el del ratio de escape del caos.

En base a estos cálculos se elaboró un módulo de análisis de la complejidad del proceso de solución, clasificando de acuerdo a la conversión realizada a forma normal conjuntiva el grado de dificultad del problema.

A continuación se realiza un análisis de la conversión del problema, luego del método de resolución y la medición de la dificultad.

5.3.1 Conversión

El método seleccionado para llevar a cabo la conversión del problema dado en formato informal a uno en forma Normal Conjuntiva fue la codificación por orden. La misma fue publicada por Tamura, Taga, Kitagawa y Banbara en 2008, y está basada en la metodología propuesta por Crawford y Baker en 1994.

Este método codifica problemas escritos en CSP en el formato que sugiere la competencia internacional de resolución de SAT a CNF, la comparación x <= a se codifica mediante una variable booleana diferente para cada variable entera x y cada valor entero a.

5. Concreción del modelo

Los autores han probado la efectividad de esta conversión realizando la codificación de 192 problemas del tipo Open Shop Scheduling y posteriormente lo han resuelto utilizando MiniSat para todos ellas han alcanzado los resultados óptimos.

Para verificar la aplicabilidad del método propuesto los autores realizaron los problemas de Guéret y Prins (80 problemas), los de Taillard (60 problemas) y los de Brucker (52 problemas). Algunos de estos problemas presentan una gran dificultad de resolución, en la tabla que se presenta a continuación se presenta el número de instancias resultas en función del tiempo de CPU:

Tiempo de CPU	Número de instancias resueltas
De 0 a 1 minuto	96
De 1 minuto a 10	77
De 10 minutos a 1 hora	14
De 1 hora a 3	3
No resueltos	2

Tabla 17 Número de instancias resueltas

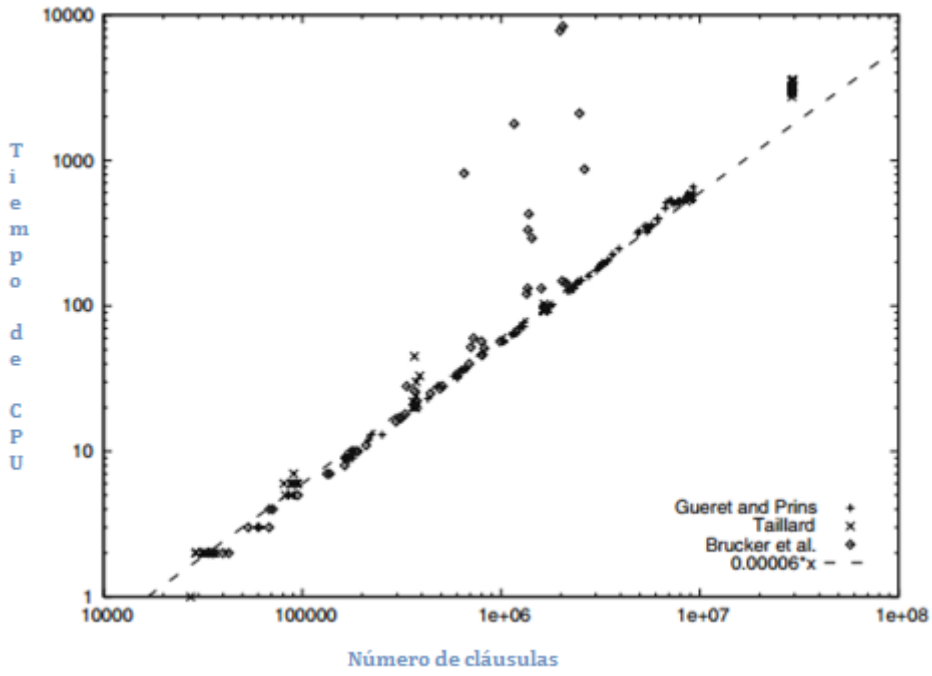


Ilustración 41 Número de clausulas y Tiempo de CPU

Existen otras formas de codificar problemas expresados en CSP a CNF, algunas de ellas como la codificación lineal, la de multivalor, por soporte, Towards, la compacta, si bien todas ellas elaboran codificaciones se ha optado por la codificación por orden por su senci-

lla adaptación dado que la librería Sugar se ha desarrollado en lenguaje JAVA, además de su efectividad a la hora de convertir problemas de diversas escalas y su capacidad de analizar si el problema planteado tiene solución o no.

A continuación se presenta una tabla mostrando resultados de la conversión y la posterior resolución de 1797 instancias puras de problemas de satisfactibilidad booleana:

Ranking	Solucionador	Resueltos	%	Tiempo de CPU
1	Mistral	1375	86	61248.25
2	**Sugar**+picosat	1354	85	71212.80
3	Choco 2.1.1	1317	82	80035.51
4	Sugar+ minisat	1314	82	71839.86
5	Choco 2.1.1b	1303	81	91613.23
6	Bpsolver	1084	68	169126.56

Tabla 18 Ranking de solucionadores

De acuerdo al rendimiento presentado y las características que ofrece Sugar como librería se optó por esta para realizar la conversión del problema que se expresa previamente en un archivo con formato csp.

Como resultado de la conversión se obtiene: el archivo .cnf que se convierte en el input de la función en Matlab, otro archivo de mapeo con extensión .map que sirve para presentar los resultados obtenidos y calcular posteriormente los tiempos de inicio. Y también devolverá un mensaje de error si el problema planteado no es satisfacible, lo que conducirá a ampliar el límite superior del makespan.

En cuanto a los límites de cálculo para el makespan en este caso el ingreso es por parte del usuario, se han analizado otras alternativas, como que el cálculo se realice por parte del sistema, pero con los métodos conocidos este proceso genera retardos muy similares al tiempo de resolución, por lo tanto se opto por el ingreso manual.

En 1963 Fisher y Thompson plantearon una seria de problemas de Job Shop Scheduling los resultados de conversión del problema FT06 (Tabla 20) mediante lo propuesto en este informe son: 1467 variables y 6718 restricciones, lo que provee un mejor resultado que la conversión mediante CSP2SAT que arroja 2579 con 16354, no solo un número muy superior sino que además la densidad de restricción también lo es.

1	1	2	1
	2	0	3
	3	1	6
	4	3	7
	5	5	3
	6	4	6
2	1	1	8
	2	2	5
	3	4	10
	4	5	10
	5	0	10
	6	3	4
3	1	2	5
	2	3	4
	3	5	8
	4	0	9
	5	1	1
	6	4	7

4	1	1	5
	2	0	5
	3	2	5
	4	3	3
	5	4	8
	6	5	9
5	1	2	9
	2	1	3
	3	4	5
	4	5	4
	5	0	3
	6	3	1
6	1	1	3
	2	3	3
	3	5	9
	4	0	10
	5	4	4
	6	2	1

Tabla 19 Fisher y Thompson 6 x 6 (FT06)

5.3.2 Simulación y dificultad

Para la resolución del problema se realizó una adaptación del algoritmo propuesto por Ercsey-Ravasz y Toroczkai en 2012 en su publicación “The chaos within Sudoku”.

Mediante este método se utilizan propiedades que caracterizan el caos para solucionar problemas de satisfacción de restricciones, como es el caso del rompecabezas Sudoku. Esta propuesta surgió luego de analizar la trayectoria del sistema de tiempo continuo determinista en el que se logró especificar que a medida que aumenta la dificulta de resolución, también lo hace en apariencia el caos.

Posterior a la conversión y obtención de un archivo que expresa el problema en forma normal conjuntiva se procede a realizar una lectura del mismo, para posteriormente determinar el número de variables, el de cláusulas, el de variables por cláusula y luego elaborar una matriz denominada C que dependiendo los valores de las variables tomará los valores -1, 1 o 0. Para cada una de las variables existe además una “variable de giro” asociada denominada s, que tomará valores entre -1 y 1, a su vez a cada cláusula de la matriz C se le

asocia la función de energía $K_m(s)=2^{-k_m}\prod_{j=1}^{N}(1-c_{mj}\,s_j)$. Dónde K_m toma valores entre 0 y 1 para todas las s, esta función es conocida como la de energía para la cláusula C_m y su valor de estado fundamental ($K_m=0$) se alcanza cuando C_m es satisfecha.

De este modo la simulación a realizar terminarán cuando todas las cláusulas estén satisfechas, es decir que todas las funciones $K_m=0$.

Además del cálculo del valor aleatorio de s, se debe calcular una variable de ajuste denominada a_m, es una variable de ajuste que crece exponencialmente con el ratio K_m, cuanto más crece este valor, más rápido lo hace a_m.

La simulación es de dos funciones, por un lado la ecuación de la derivada de la ecuación de giro: $\frac{ds_i}{dt}=\sum_{m=1}^{M}2a_m c_{mi}K_{mi}(s)K_m(s),\ \ i=1,\ldots,N$ y por otro lado el gradiente descendente de sobre el escape de energía: $\frac{da_m}{dt}=a_m K_m(s),\quad m=1,\ldots,M$.

Si el problema no es satisfacible el sistema genera un dinámica caótica en $[-1,1]^N$ indefinidamente.

En la implementación realizada en lenguaje Matlab se utilizó el método adaptativo para la resolución de ecuaciones diferenciales Runge-Kutta de cuarto – quinto orden (ode45), el error relativo para la función será de: 1e-3, finalizará cuando el valor del gradiente para todos los puntos fijos sea 0, las condiciones iniciales para s son arbitrarias pero para a deben ser positivas.

En la ilustración 44 se presenta el resultado de una simulación realizada para 133 variables y 338 restricciones:

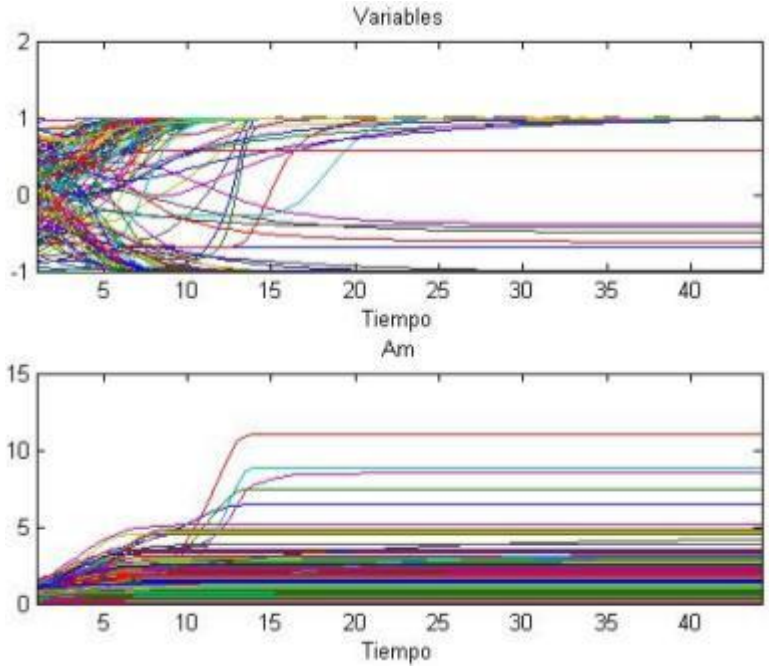


Ilustración 42 Simulación para 133 variables y 338 restricciones

Se incorporó en el cálculo de la simulación dos valores interesantes para analizar la complejidad de la resolución ellos son: la cantidad de pasos que tomó alcanzar la solución y el otro es el tiempo de CPU requerido.

En la tabla 21 se presenta el valor de pasos y tiempo de CPU para algunos de los problemas resueltos:

Problema (trabajos –máquinas)	Variables N	Restricciones M	Alfa =M/N	Pasos	Tiempo CPU
2 – 2	172	324	1.88	54	26.45
2 – 2	332	682	2.05	110	54.06
3 – 3	518	1165	2.48	111	54.65
3 – 3	508	1259	2.47	173	85.50
4 – 4	610	1790	2.93	106	52.47
4 – 4	350	1158	3.31	144	71.40
5 – 5	413	1459	3.53	94	46.27
5 – 5	949	3338	3.51	100	49.34
6 – 6	503	1683	3.35	76	37.35
6 – 6	629	3024	4.73	149	73.51

Tabla 20 Resultados de resolución

Realizando un análisis de la tabla 20 se puede observar que el tiempo de CPU correspondiente a cada paso de resolución es 0,49, y que no sigue una proporción la cantidad de restricciones y variables con el tiempo de resolución demandada.

Estos valores son importantes, dado que una vez determinado el nivel de complejidad (asociado al ratio de escape), se puede obtener un promedio de los pasos necesarios para hallar

la solución, y conociendo que el tiempo de CPU correspondiente a cada paso es 0,49 obtener el tiempo promedio de resolución de CPU.

Otro punto relevante durante la implementación del solucionador fue la incorporación del análisis de complejidad para llevarlo a cabo se debieron resolver varios ejercicios (más de 100) de distintos grado de dificultad, en este caso se adjuntan los datos relevados para problemas de distintas grados de dificultad para 3 trabajos y 3 máquinas.

Para obtener la probabilidad de que el problema no sea resuelto en el tiempo (t) se procesaron los resultados de 30 ejercicios de esta clase que previamente se clasificaron de acuerdo a su densidad de restricción (alfa número de restricciones/número de variables) en Fáciles, Medios y Difíciles de la siguiente forma:

Trabajos x Máquinas	Dificultad	Límite densidad de restricción
Entre 0 y 4	Fácil	0 a 1
	Medio	1 a 1,5
	Difícil	1,5 a 2
Entre 4 y 9	Fácil	0 a 1,5
	Medio	1,5 a 2
	Difícil	2 a 2,5
Entre 9 y 16	Fácil	0 a 2
	Medio	2 a 2,5
	Difícil	2,5 a 3

Tabla 21 Rangos de dificultad

Es decir que se estableció una escala en la que de acuerdo a la multiplicación de la cantidad de trabajos y máquina, y luego para cada una de las categorías se va añadiendo 0,5 a los límites de rango a medida que aumenta la dificultad.

En base a esta escala que se construyó luego de la resolución mediante ode45 con una tolerancia relativa al error de 10^{-3} de varios problemas, curvas para cada tipo de problema que permiten luego medir el ratio de escape del caos. Lo que se hizo es en base a los resultados obtenidos se calcularon las probabilidades de no resolución en tiempos determinados, y se construyeron así tres curvas para cada uno de los niveles de dificultad, como se puede observar en el gráfico 45 la probabilidad de no resolución decae exponencialmente. Dado que p(t) $p(t) \sim e^{-kt}$ donde k es el ratio de escape, con los valores obtenidos se pudo ajustar la función y obtener así los valores correspondientes a k para cada nivel de dificultad. Y posteriormente aplicándole a k el logaritmo negativo en base de 10 se obtuvo el nivel de dificultad correspondiente de acuerdo al tiempo que el sistema pasa por una trayectoria caóti-

ca, que como ya se expresó mientras más grande sea más dificultoso es el proceso de encontrar la solución.

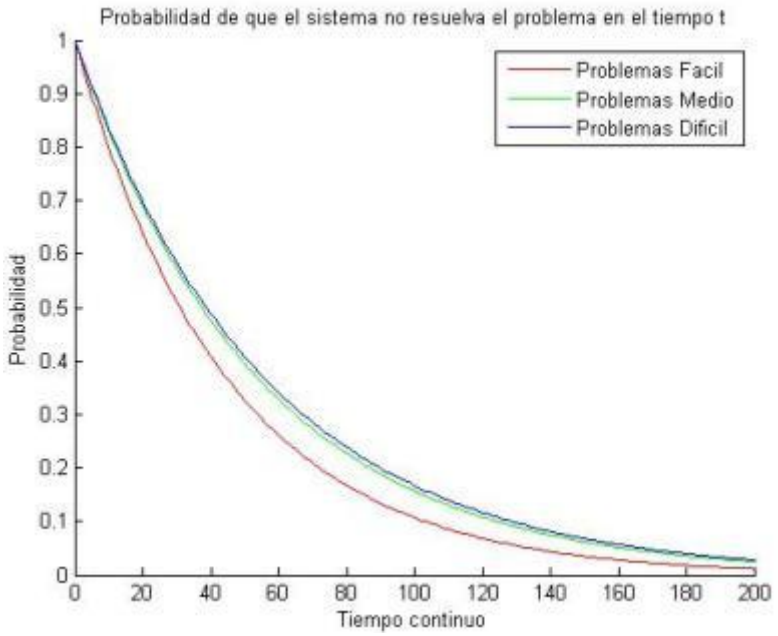


Ilustración 43 Probabilidad de no resolución en t

En la tabla 23 se presentan los valores del ratio de escape para cada nivel de los problemas de 3 trabajos y 3 máquinas y también su valor de dificultad:

Dificultad	**k**	**$-\log_{10}k$**
Fácil	0.0223	1.65
Medio	0.0185	1.73
Difícil	0.0179	1.77

Tabla 22 Dificultad para problemas 3x3

Si bien actualmente existen otras formas de clasificar la dificultad de problemas de Job Shop Scheduling esta nueva forma parece ajustarse mejor a la lógica de cálculo utilizada, proveyendo un novedoso método para determinar tanto el tiempo de CPU promedio para la resolución de un problema con una dificultad dada y que si bien no presenta una relación lineal con el ratio alfa (densidad de restricción), sí denota valores similares en la escala de dificultad.

5.4 Comparación con otros modelos

Tal como se introdujo en el marco teórico existen distintos modelos para abordar la solución de los problemas de Job Shop Scheduling, en este apartado se realiza una comparación con otros modelos de resolución en los que se los aborda como problemas de Satisfacción de Restricciones Booleanas.

En la tabla 24 se muestra la comparación de los resultados obtenidos para el problema FT06 (tabla 20).

Solucionador	Makespan obtenido	Tiempo de resolución
Optimizar (el desarrollado)	66	64
Back tracking	60	30
Random	60	29
Análisis simulado (SA)	55	29
Taboo	58	30
Inter block backtracking (IBB)	60	30

Tabla 23 Comparación de modelos de resolución

Estos resultados se obtuvieron mediante la ejecución en primer lugar del software desarrollado observándose en la figura 47 la naturaleza compleja de la resolución, que queda bien definida al ver la trayectoria caótica densa en el gráfico de variables. Luego para analizar los demás modelos se utilizó el software elaborado por Tamura denominado Cream FT06[29] donde se permite obtener los resultados para distintos modelos de resolución, partiendo todos del mismo problema expresado en forma de matriz, y que se convierte utilizando la misma librería Sugar que se utiliza en el software desarrollado (Optimizar). La interfaz de aplicación de Cream es compleja, y tiene un mero sentido educativo, debido a que se trata de un archivo de extensión java que no ofrece la posibilidad de ingresar datos.

[29] Cream FT06: http://bach.istc.kobe-u.ac.jp/cream/cream106/src/examples/FT06.java

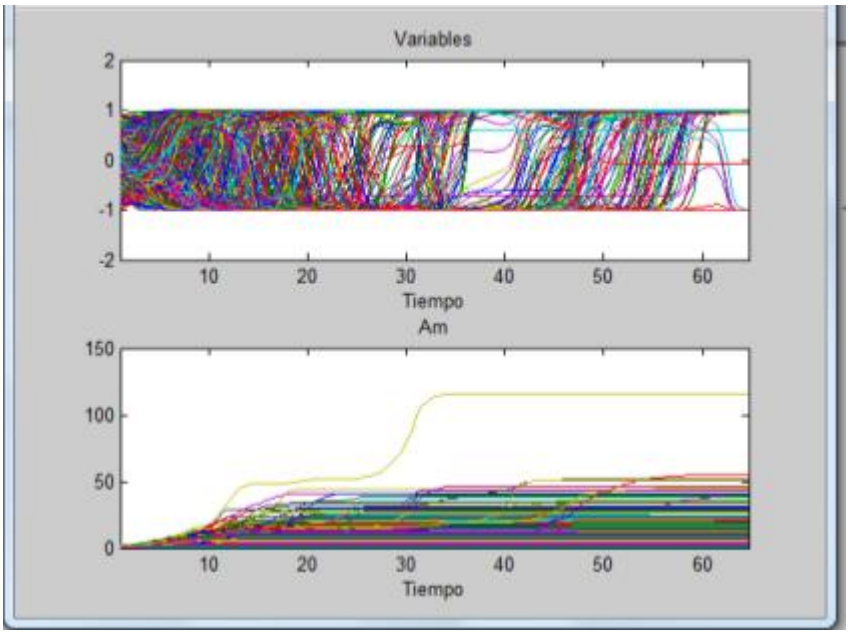


Ilustración 44 Resolución de FT06

Como puede desprenderse del análisis de la tabla 24 Optimizar ofrece el peor rendimiento, no sólo por su lentitud de resolución, sino que además presenta el makespan más grande. En la ilustración 47 se muestra el proceso de resolución más veloz y con el menor makespan que coincide que el mejor valor obtenido para este problema, en este caso se ha alcanzado mediante el Análisis Simulado o Recorrido simulado (SA, Simulated Annealing) este método desarrollado por Kirkpatrick en 1984 genera soluciones aleatorias y calcula los costos de esa solución, el algoritmo evoluciona realizando desplazamientos unitarios que son aceptados si su coste es inferior a la solución anterior.

Ilustración 45 Resolución de FT06 con SA

5. Concreción del modelo

Queda expuesto que el método desarrollado no presenta las mejores prestaciones, pero como se enunció anteriormente combinado con la interfaz de uso sencilla, es una solución muy eficiente para empresas de menor tamaño que no cuenten con más de 25 operaciones, como es el caso de las PyMES.

Además de las potencialidades de desarrollo y expansión que el modelo diseñado presenta, tendrá una relevancia teórica que aún no se encuentra plenamente explotada ya que se trata de un tema actual que continua siendo investigado y para el que se siguen descubriendo aplicaciones.

6 Conclusiones

El objetivo de la presente tesis ha sido el desarrollo de un modelo de optimización de la planificación de la producción utilizando un solver publicado en 2012 en la revista de divulgación científica: Nature - Scientific Reports por Mària Ercsey-Ravasz y Zoltán Toroczkai para la resolución del rompecabezas Sudoku.

Se abordó el complejo problema del Job Shop Scheduling mediante la optimización combinatoria utilizando sistemas dinámicos, con especial interés en la resolución de problemas booleanos denominados k-SAT. Para realizarlo fue necesario seleccionar un método de conversión del problema, ingresado por el usuario mediante una interfaz sencilla, a la forma normal conjuntiva que sirve de "*input"* para la función que calcula los tiempos de inicio de cada operación desarrollada en Matlab. Se optó por un método de conversión desarrollado por los japoneses Tamura y Taga denominado Sugar que implementa en una librería de Java la codificación por orden que es muy similar a la propuesta en 1994 por Crawford y Baker para el Job Shop Scheduling.

Los métodos propuestos actualmente para resolver la planificación industrial utilizan desde algoritmos genéticos hasta modelos de inteligencia artificial, en algunos casos el problema es planteado como la satisfactibilidad de restricciones, pero en ningún caso de los estudiados, se explotaba la optimización combinatoria utilizando sistemas dinámicos, basado en principios de la teoría del caos, como es el que realiza el algoritmo de resolución propuesto.

El modelo que se desarrolló para solucionar el problema discreto de planificación de la producción (Optimizar) puede ser aplicado en diversas situaciones complejas en las que la situación problemática no puede ser resuelta en tiempo polinómico y que puede expresarse mediante expresiones booleanas con variables y sin cuantificadores. Por todo esto es importante la extensibilidad del modelo desarrollado a otras aplicaciones como la criptografía, problemas de transporte (como el del viajante), empresariales y también del ámbito industrial, siempre y cuando el valor de las variables se encuentre delimitado.

Para implementar la solución propuesta fue necesario:

- Crear una interfaz sencilla donde los usuarios vuelquen los datos del problema.
- Establecer el cálculo del mínimo makespan del problema, para luego en base a estos valores crear el archivo de satisfacción de restricciones (CSP).

- Utilizar la librería Sugar para realizar una conversión a Forma Normal Conjuntiva del problema planteado y un archivo de mapeo para luego presentar la solución.
- Crear una función en Matlab capaz de leer el archivo CNF y plantearlo en una matriz, para luego calcular mediante el solucionador de tiempo continuo determinista los valores para cada variable, recopilando los resultados obtenidos de la simulación realizada mediante ode45 de un sistema de ecuaciones que analiza la trayectoria caótica por la que pasa el problema hasta que todos los valores del vector gradiente de energía son iguales a 0.
- Con los resultados obtenidos de la simulación calcula junto al archivo de mapeo los tiempos de inicio de cada operación, los de finalización, y en base a las fecha de inicio calcula la programación de todas las tareas.
- Mediante la librería jFreeChart se elabora un diagrama de Gantt que permite graficar los resultados obtenidos luego de la ejecución de la función anteriormente descrita.
- Por último se incorporó un módulo de análisis para analizar la complejidad del problema planteado, mostrando la cantidad de variables, restricciones y la densidad de restricción (alfa) obtenidas luego de la codificación, la cantidad de pasos y de tiempo de CPU que demandó la solución, la clasificación correspondiente al problema, que fue obtenida luego de resolver más de 100 JSSP, para los que se hallaron las probabilidades de que no se encuentra solución en un determinado tiempo y en base a esos valores ajustados se determinó el ratio de escape (k) para distinta clases de problemas, a los que aplicándole el logaritmo negado se pudo construir una escala de valores positivos que crece a medida que lo hace la dificultad.

En la última etapa de esta investigación, la de concreción del modelo, se comparó el método desarrollado con otros algoritmos existentes y se notó que para problemas de complejidad media o superior, la eficiencia no es apropiada, existiendo mejores beneficios al implementar alguno de los solucionadores SAT como el de recorrido asistido. Pero también hay que destacar que la interfaz de ingreso del método propuesto en este trabajo es muy sencilla de utilizar por los encargados de producción, incorporando la posibilidad de determinar todos los tiempos de inicio y finalización y la gráfica de la misma. No es así el caso de ninguno de los métodos comparativos analizados, durante la etapa de investigación se observó que algunos de los programas pagos sí contaban con estas características que

alientan a la introducción del sistema en la empresa, facilitando el uso para los responsables de la planificación.

Otra característica relevante de la solución propuesta es que utilizando características del sistema dinámico sin límites se pudieron calcular tasas de evaluación de la dificultad de los problemas planteados, quedando claramente determinado que mientras mayor es el tiempo en que pertenece en la trayectoria caótica, mayor lo es también su dificultad de resolución. A estos resultados se arribó luego de calcular probabilidades y analizar las soluciones de más de 100 problemas de Job Shop Scheduling.

Realizando la implementación han surgido inconvenientes asociados a la naturaleza compleja de resolución del problema a solucionar, por ejemplo la cantidad de memoria necesaria para almacenar la gran cantidad de variables y restricciones que surge luego de la codificación de planificaciones con más de 100 operaciones involucradas, lo que demanda para la correcta ejecución equipos con sistemas operativos de 64 bits, al igual que el programa Matlab y que posean memorias de 4 gb o superior. Este problema no surge cuando el tamaño del problema es menor, posibilitando la ejecución en equipos con recursos inferiores. Por estos motivos resulta aconsejable la implementación para empresas de pequeña o mediana envergadura en su proceso productivo, de lo contrario será fundamental contar con el equipamiento anteriormente descripto para alcanzar los resultados esperados.

Estos son los resultados de meses de investigación acerca de la optimización, la satisfacción de restricciones, la planificación industrial, la complejidad computacional, los modelos que se desprenden de la teoría del caos, la conversión a diferentes formas y la implementación en lenguaje Matlab. Con lo que los objetivos planteados en el anteproyecto de este documento se han alcanzados porque se logro adaptar el algoritmo solucionador de tiempo continuo determinista desarrollado por Ercsey-Ravasz y Torczkai para resolver Sudokus a un complejo software capaz de determinar los tiempos de inicio y finalización de cada operación involucrado en el proceso productivo dentro de una industria.

Analizando los resultados obtenidos luego de la implementación y prueba de Optimizar se logra presentar un modelo que si bien resulta ineficiente en comparación con otros vigentes, explota mecanismos de resolución que antes no habían sido siquiera evaluados, además de proveer ventajas en el momento de la implementación industrial debido a sus características de uso simple.

A la luz de los resultados puede determinarse que si bien este modelo propuesto resulta en algunas ocasiones demasiado complejo para la resolución de la planificación industrial podrá tener diversas aplicaciones en campos de estudio muy variados, generando avances en la forma de concebir la resolución de problemas de satisfacción enmarcados en la clase de complejidad NP-Completa. Por esto si bien aquí concluye el trabajo desarrollado se considera que puede ser base sólida para condensar nuevas aplicaciones y ampliar los modelos propuestos para el abordaje de la planificación industrial.

7 Referencia Bibliográfica

1. Applegate D, Cook W. In A computational study of the job-shop scheduling problem: ORSA Journal of Computing; 1991. p. 149-156.

2. Brucker P,JBySB. In A branch and bound algorithm for the job shop scheduling problem: Discrete Applied Mathematics; 1994. p. 107-127.

3. Carlier J, Pinson E. In An algorithm for solving the job-shop problem: Management Science; 1989. p. 164-176.

4. Crawford J, Baker A. In Experimental Results on the Application of Satisfiability Algorithms to Scheduling Problems.: AAAI Press; 1994. p. 1092-1097.

5. Karp R. In Reducibility among combinatorial problems. In Complexity of Computer Computations. New York: Plenum; 1972. p. 85-103.

6. Barahona F. In On the computational complexity of Ising spin glass models.: J. Phys. A: Math; 1982. p. 3421-3253.

7. Ercsey-Ravasz M, Toroczkai Z. The chaos within Sudoku. [Online].; 2012 [cited 2013 Abril 01. Available from:

   http://www.nature.com/srep/2012/121011/srep00725/full/srep00725.html.

8. Tamura N, Taga A, Kitagawa S, Banbara M. Compiling finite linear CSP into SAT. [Online].; 2008 [cited 2013 Julio 01. Available from: http://link.springer.com/article/10.1007%2Fs10601-008-9061-0.

9. Vázquez Gómez J. Análisis y diseño de algoritmos. [Online].; 2012 [cited 2013 Octubre Available from:

   http://www.aliatuniversidades.com.mx/bibliotecasdigitales/pdf/sistemas/Analisis_y_disenio_de_algoritmos.pdf.

10. Ercset-Ravasz M, Toroczkai Z. Optimization hardness as transient chaos in an analog approach to constraint satisfaction. [Online].; 2012 [cited 2013 Julio 1. Available from: http://arxiv.org/pdf/1208.0526.pdf.

11. Barber , Salido. Introducción a la programación de restricciones. [Online].; 2003

[cited 2013 01 Julio. Available from: http://users.dsic.upv.es/~msalido/papers/aepia-introduccion.pdf.

12. Cheng C, Smith S. Applying Constraint Satisfaction techniques to Job Shop Scheduling. [Online].; 1995 [cited 2013 Agosto 1. Available from: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.72.5444&rep=rep1&type=pdf.

13. Cook S. The Complexity of Theorem Proving Procedure. In.: ACM Symposium on Thoery of Computing,; 1971. p. 151-158.

14. Cvitanovic. The chaos book. [Online].; 2013 [cited 2013 Septiembre 01. Available from: http://chaosbook.org/.

15. Giunchiglia E. In Constraints and AI Planning.: IEEE Intelligent Systems; 2005. p. 67-72.

16. Gu J, Du Q. On optimizing the satisfiability (SAT) problem. [Online].; 1999 [cited 2013 Julio 1. Available from: http://link.springer.com/article/10.1007%2FBF02952482.

17. Kan A, Lawler E, Lenstra J, Shmoys D. Sequency and Scheduling Algortihms and complexity. [Online].; 1993 [cited 2013 Agosto 01. Available from: http://pages.cs.wisc.edu/~dluu/ps/Lawler,%20Lenstra,%20Kan,%20and%20Shmoys%20-%20Sequencing%20and%20Scheduling%20Algorithms%20and%20Complexity.pdf.

18. Lorenz E. Deterministic nonperiodic flow. [Online].; 1963 [cited 2013 Agosto 01. Available from: http://journals.ametsoc.org/doi/pdf/10.1175/1520-0469(1963)020%3C0130%3ADNF%3E2.0.CO%3B2.

19. Lynce I, Ouaknine J. Sudoku as SAT problem. [Online].; 2006 [cited 2013 Julio 01. Available from: http://www.cs.ox.ac.uk/joel.ouaknine/publications/sudoku05.pdf.

20. Peña V, Zumelzu L. Estado del Arte del Job Shop Scheduling Problem. [Online].; 2006 [cited 2013 Abril 01. Available from: http://www.alumnos.inf.utfsm.cl/~vpena/ramos/ili295/ia-jobshop.pdf.

21. Schaefer T. The complexity of satisfiability problems. [Online].; 1978 [cited 2013

Julio 01. Available from: http://www.ccs.neu.edu/home/lieber/courses/csg260/f06/materials/papers/max-sat/p216-schaefer.pdf.

22. Siegelman H. Computation Beyond the Turing Limit. [Online].; 1995 [cited 2013 Julio 1. Available from: http://binds.cs.umass.edu/papers/1995_Siegelmann_Science.pdf.

23. Tsang E. Foundations of Constraint Satisfaction. [Online].; 1970 [cited 2013 Agosto 01. Available from: http://www.bracil.net/CSP/papers/Tsang-Fcs1993.pdf/Tsang-Fcs1993-Toc.pdf.

24. Valverde J. Sudoku un problema NP-completo. [Online].; 2011 [cited 2013 Junio 01. Available from: http://seccperu.org/files/sudoku%20is%20NP.pdf.

25. Von der Becke C. Caos. [Online].; 2001 [cited 2013 Agosto 01. Available from: http://vonderbecke.org/caoscog.html.

26. Yato T. Complexity and completeness of fiding another solution and its application to puzzles. [Online].; 2003 [cited 2013 Julio 01. Available from: http://www-imai.is.s.u-tokyo.ac.jp/~yato/data2/MasterThesis.pdf.

8 Bibliografía

1. Aho A, Hopcroft J, Ullman J. Estructuras de datos y algoritmos: Addison-Wesley Iberoamericana; 1988.

2. Ander Egg E. Introducción a la planificación: El cid; 1978.

3. Baker T, Gill J, Solovat R. Relativizatons of the P = NP Question: SIAM; 1975.

4. Campos D. Determinismo, Caos e impredictibilidad: Academia Colombiana Ciencias; 2002.

5. Cardona S, Alonso S, Jaramillo S. Lógica matemática para Ingeniería de Sistemas y Computación: Elizcom; 2010.

6. Chakraborty U. Computational Intelligence in Flow Shop and Job Shop Scheduling: Springer; 2009.

7. Conway R, Maxwell W, Miller L. Theory of scheduling: Courier Dover Publications; 1967.

8. Elser. Searching with iterated maps: Proceedings of the Nation Academy of Sciences ; 2007.

9. Fisher H, Thompson GL. In Probabilistic learning combinations of local job-shop scheduling rules in Industrial Scheduling. Englewood Cliffs: Prentice-Hal; 1963. p. 225-251.

10. Fogarty D, Blackstone J, Hoffman T. Production & inventory management: Co, South-Western Pub.; 1991.

11. Gagne R, Briggs M. La Planificación de la Enseñanza: Trillas; 1987.

12. Gaither N, Frazier G. Administración de producción y operaciones: Cengage Learning ; 2000.

13. Garey M, Johnson D. Computers and Intractability: A Guide to the Theory of NP-Completeness: W H Freeman ; 1979.

14. Harel D, Kozen D, Tiuryn J. Dynamic Logic: MIT Press; 2009.

15. Joyanes Aguilar L. Fundamentos de la programación: Mc Graw Hill; 1999.

16. Kumar. Theory Of Automata: Mc Graw Hill; 2010.

17. Lai Y. Transient Chaos: Complex Dynamics on Finite-Time Scales: Springer; 2011.

18. Levin L. Universal Search Problems: Problems of Information Transmission; 1973.

19. Matus C. Estrategia y plan: Editorial universitaria; 1972.

20. Ogata K. Ingeniería de Control Moderna: Pearson Education; 2003.

21. Ott E. Chaos in Dynamical Systems: Cambridge University Press; 2002.

22. Pinedo M. Scheduling: theory, algorithms and systems: Springer; 2012.

23. Poincaré H. Science and Method: Maitland; 1914.

24. Sametband M. Entre el orden y el caos: la complejidad: Fondo de cultura económica USA; 1999.

25. Sanchez G. Técnicas participativas para la planeación: Fundación ICA; 2003.

26. Silva O. Planificacion eficiente: Lulu Enterprises Inc; 2007.

27. Smith P. El caos: Cambridge University Press; 2001.

28. Solé R, Marubia S. Orden y caos en sistemas complejo: Edicions UPC; 2001.

29. Yamada T, Nakano R. Genetic algorithms in engineering systems: IEE Control Engineering series; 1997.

9 Glosario

- *JSSP:* Job Shop Scheduling Problem (Problema de planificación industrial) es un problema de optimización en el marco computacional y de la investigación de operaciones en el cual cada trabajo se asocia a un recurso en un tiempo particular.
- *NP:* Clasificación de problemas en la lógica computación que no pueden ser resueltos en tiempo polinomial.
- *SAT:* Problema de Satisfactibilidad Booleana, fue el primer problema identificado como perteneciente a la clase de complejidad NP-completo.
- *CNF:* forma de expresar una fórmula como una conjunción de cláusulas, donde una cláusula es una disyunción de literales, donde un literal y su complemento no pueden aparecer en la misma cláusula.
- *CSP:* son problemas matemáticos definidos como una seria de objetos con estados que deben ser satisfechos de acuerdo a una serie de restricciones o limitaciones.
- *ODE:* ecuación diferencial ordinaria matemática que contiene una función desconocida de una variable independiente y relaciona con sus derivadas: una *sola* variable independiente (a diferencia de las ecuaciones diferenciales parciales que involucran derivadas parciales de varias variables), y una o más de sus derivadas respecto de tal variable.
- *CTDS:* solucionador que brinda un modelo donde las mismas entradas producen siempre el mismo estado y las mismas salidas. En otras palabras, el azar no juega ningún papel en el modelo. El tiempo es una entrada del sistema (que causa efecto en el mismo), es decir, los valores internos del modelo cambian con el tiempo. Y que permite conocer los valores de salida en todos los instantes del intervalo de tiempo. Los modelos dinámicos continuos suelen estar basados en ecuaciones diferenciales, tanto ordinarias (como es el caso del método propuesto) como en derivadas parciales.
- *RK:* conjunto de métodos genéricos iterativos, explícitos e implícitos, de resolución numérica de ecuaciones diferenciales. En este trabajo se utilizó el método Runge Kutta adaptativo de cuarto – quinto orden, este combina ambos modelos mediante una técnica que utiliza tramos de tamaño variable eligiendo el tamaño de tramo en cada fase para alcanzar un grado determinado de precisión (es un método adaptativo).

- *ZOPP:* sistema de técnicas y procedimientos para la planeación de proyectos orientados a la acción. El nombre proviene de la denominación alemana Ziel Orientiere Projekt Planung (Planeación de Proyectos Orientada a Objetivos).

10 Anexos

10.1 Apéndice A: COCOMO

Este es un modelo matemático de base empírica que permite estimar los costos asociados a la construcción de un software, incluyendo distintos niveles de detalles.

Cálculo puntos de función:

Archivos lógicos internos: máquinas – trabajos –operaciones – demo_jssp - conversión

Archivos de interface: problema – complejidad

Entradas externas: Ingresar problema – Ingresar máximo makespan – Ingresar fecha inicio

Salidas: archivo csp – archivo map – archivo cnf – archivo de texto resultados -inicio de cada operación – fin de cada operación – fechas inicio operación – fecha fin de cada operación – makespan – complejidad – diagrama de Gantt

Consultas: convertir problema - obtener resultados – calcular complejidad

Clasificación de las funciones:

- Lógicos:

Función	Grado de la función
Máquinas	SIMPLE
Trabajos	SIMPLE
Operaciones	SIMPLE
Demo_jssp	MEDIO
Conversión	SIMPLE

- Archivos interface:

Función	Grado
Problema	SIMPLE
Complejidad	SIMPLE

- Archivos de entrada:

Función	Grado
Ingresar problema	SIMPLE
Máximo makespan	SIMPLE
Fecha inicio	SIMPLE

- Salidas:

Función	Grado
Inicio de cada operación	SIMPLE
Fin de cada operación	SIMPLE
Fechas inicio operación	SIMPLE
Fecha fin de cada operación	SIMPLE
Makespan	SIMPLE
Complejidad	SIMPLE
Diagrama de Gantt	SIMPLE
Archivo csp	SIMPLE
Archivo map	SIMPLE
Archivo cnf	SIMPLE
Archivo de texto resultados	SIMPLE

- Consultas:

Función	Grado
Convertir problema	SIMPLE
Obtener resultados	DIFICIL
Calcular complejidad	SIMPLE
Consultar factura	SIMPLE
Consulta equipo	SIMPLE
Consulta trabajos realizados	SIMPLE

Cálculo: Puntos sin ajuste

Simple Promedio Complejo	Simple	Promedio	Complejo	Total
Entradas	3 * 3	* 4	* 6	9
Salidas	11 * 4	* 5	* 7	44
Almacenamientos	4 * 7	1 * 10	* 15	38
Interfaces	2 * 5	* 7	* 10	10
Consultas	5 * 3	* 4	1 * 6	21

Total de puntos sin ajustar (PSA) = 122

10. Anexos

COCOMO II permite estimar el coste, esfuerzo y tiempo cuando se planifica una actividad de desarrollo software. Este sistema será implementado en el lenguaje Java, las líneas de código por cada punto de función equivalen a 53. De este modo se puede calcular el esfuerzo, hallando la variable KDLC.

KDLC=(122*53)/1000= 6,466

Debido a que el número de líneas de código no supera los 50 KLDC el tipo utilizado será el orgánico, por consiguiente, los coeficientes a utilizar serán las siguientes:

Proyecto Software	**A**	**E**	**C**	**D**
Orgánico	3,2	1,05	2,5	0,83
Semi acoplado	3,0	1,12	2,5	0,35
Empotrado	2,8	1,2	2,5	0,35

La FAE se obtiene mediante la multiplicación de los valores evaluados para 15 conductores de coste:

CONDUCTORES DE COSTE

Conductores de coste	**Valoración**					
	Muy bajo	Bajo	No-minal	Alto	Muy alto	Extr alto
Fiabilidad requerida del software	0,75	0,88	1	1,15	1,40	-
Tamaño de la base de datos		0,94	1	1,08	1,16	-
Complejidad del producto	0,7	0,85	1	1,15	1,30	1,65
Restricciones del tiempo de ejecución	-	-	1	1,11	1,30	1,66
Restricciones del almacenamiento principal	-	-	1	1,06	1,21	1,56
Volatilidad de la máquina virtual	-	0,87	1	1,15	1,30	-
Tiempo de respuesta del or-	-	0,87	1	1,07	1,15	-

denador						
Capacidad del analista	1,46	1,19	1	0,86	0,71	-
Experiencia en la aplicación	1,29	1,13	1	0,91	0,82	-
Capacidad de los programa-dores	1,42	1,17	1	0,86	0,70	-
Experiencia en S.O. utilizado	1,21	1,10	1	0,90	-	-
Experiencia en el lenguaje de programación	1,14	1,07	1	0,95	-	-
Experiencia en el lenguaje de programación	1,14	1,07	1	0,95	-	-
Prácticas de programación modernas	1,24	1,10	1	0,91	0,82	-
Utilización de herramientas software	1,24	1,10	1	0,91	0,83	-
Limitaciones de planificación	1,23	1,08	1	1,04	1,10	-

FAE=1,4 . 0,94 . 1,3 . 1 . 1 . 1 . 1,15 . 0,71 . 0,82 . 0,7 . 0,95 . 0,91 . 0,91 . 1,08= 0,7486

E=a KLDC.FAE=3,2 . (6,466)^1,05 . 0,7486 =17

Cálculo tiempo de desarrollo: T= c . Esfuerzo ^d= 2,5 . 17^0,38= 7,33

Productividad: PR=LDC/Esfuerzo =6466/17=380 LDC/personas mes

Personal promedio: P=E/T=17/7,33=2,31 personas

Según estas cifras será necesario un equipo de 3 personas trabajando alrededor de 6 meses.

10.2 Apéndice B: Función demo_jssp.m

A continuación se presenta la función demo_jssp elaborada para lenguaje Matlab, que es la que efectúa el proceso de resolución del problema y devuelve una gráfica de la simulación y los resultados que luego se mapean para obtener los tiempos de inicio.

```
function out = demo_jssp()
Fs = read_cnf('jss.cnf');
cla
N = max(max(Fs));       % numero de variables
M = length(Fs(:,1));    % numero de clausulas
K = sum(abs(Fs)>0,2);   % numero de variables en cada clausula
% Matriz Cmi
C = zeros(M,N);
for clause=1:M
  for varindex=1:K(clause)
    var = abs(Fs(clause,varindex));
    C(clause,var) = sign(Fs(clause,varindex));
  end
end
Ss      = rand(1,N)*2-1; % valor aleatorio "spin" entre -1 y 1
As      = rand(1,M); % valor aleatorio para la clausulas am entre 0 y 1
Kms = zeros(M,1);
dys = zeros(N,1);
dirs = sign(Fs);
ctn=0;
options = odeset('RelTol',1e-3, 'Events',@event_function);
[T,Y] = ode45(@simulate,[0:0.5:175],[Ss, As],options);
s = sum(abs(Y(:,1:N)),2);
s(s==Inf)=0;
[~,ind] = max(s(:));
sol = sign(Y(ind,1:N));
valid = valid_solution(Fs,sol);
steps = ind;
cTime = 1000;
cTime = T(ind);
    % Dibujar
   figure(1);
   subplot(2,1,1);
   plot(T,Y(:,1:N));
   xlabel('Tiempo');
   xlim([1 cTime]);
   title('Variables');
   subplot(2,1,2);
   plot(T,Y(:,N+1:M+N));
   xlabel('Tiempo');
   xlim([1 cTime]);
   title('Am');
if valid
   display('Solución encontrada')
   sol
   steps
   cTime
   N
   M
   for i=1:N
   solu(i)=sol(i)*i;
end
```

```
  str='SAT';
  filename = 'C:\JSSP\matlab\resultado.txt';
  fid = fopen(filename,'w');            %# Abrir
  if fid ~= -1
   fprintf(fid,'%s\n',str);       %# Imprimir
   fclose(fid);                   %# Cerrar
  end
  dlmwrite(filename,[solu],'-append', 'newline','pc','delimiter','\t');
  saveas(figure(1),'grafico.jpg');
  com=M/N;
  xlswrite( 'C:\JSSP\matlab\complejidad.xls', { N , M, com , steps, cTime } );
else
  display('No se encontró solución en el tiempo límite')
  sol = -1;
end
 % Primeros N parámetros son los valores Si para las variables
 % Los siguientes M parámetros son los valores am para las clausulas
    function dy = simulate(tt,y)
   Kms;
   sis = y(1:N); % Variables
   ams = y(N+1:M+N); % Clausulas
   kms = Km(sis);
   dy  = zeros(N+M,1);
   ctn = ctn + 1;
   for i=1:N % Para todas las variables
      div = (1- (C(:,i)*sis(i)));
      kmis   = kms ./ (div);
      kmis((isnan(kmis) + isinf(kmis)) > 0) = 0;
      dy(i)  = sum(2 * ams .* C(:,i) .* kmis .* kms);
   end
  dy(N+1:M+N) = ams .* kms; % dam/dt
  dys = dy;
 end
% Km
 function ret = Km(s)
    ss = repmat(s',M,1);
    ret = 2 .^ (-K) .* prod(1 - (C.*ss),2);
 end
   function [value,isterminal,direction] = event_function(t,y)
   value = +~all(abs(dys) <= 0.01); % Detenerse cuando nada cambia más
   isterminal = 1; % Terminar después del primer evento
   direction = 0;  % Obtener todos los zeros
 end
end
```

10.3 Apéndice C: código Java

El programa desarrollado en Java cuenta con una clase principal en la que se muestra el formulario para el ingreso del problema y donde posteriormente se representa la solución, además existe otro formulario donde se analiza la complejidad.

Y mediante otras clases se realiza la conversión del problema a forma normal conjuntiva, y posteriormente se decodifican los resultados.

Las librerías utilizadas son:

- Sugar-v2-0-0
- Jcalendar
- Jcommon-1.0.20
- Jfreechart-1.0.14
- matlabcontrol-4.1.0

Los formularios jFrame:

- Inicio

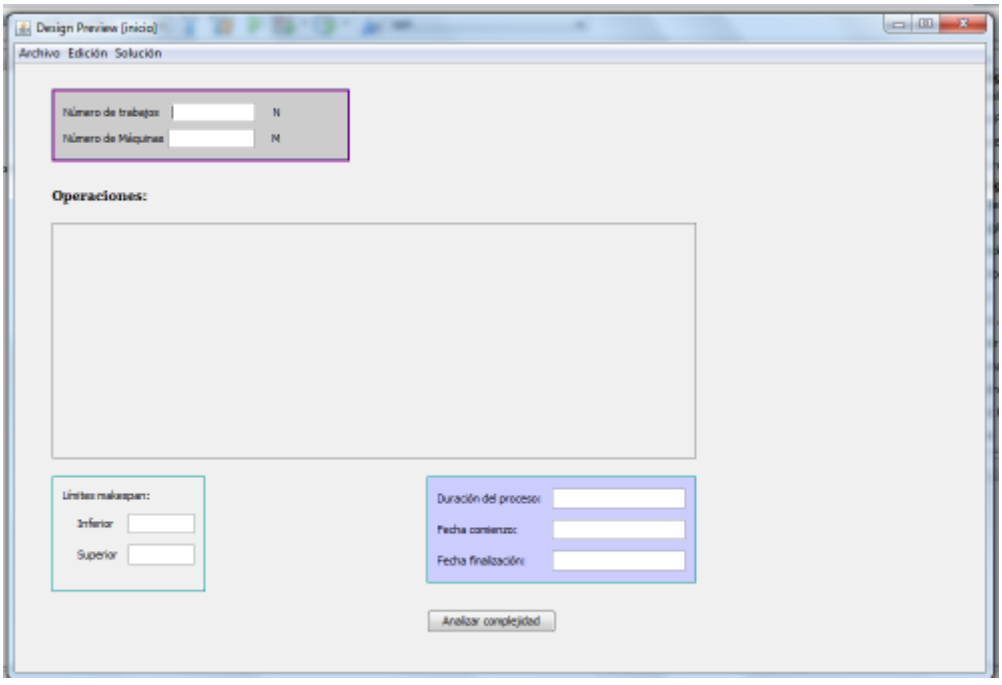


Ilustración 46 Formulario inicio

- Complejidad

Ilustración 47 Formulario complejidad

Las clases son:

- inicio.java
- excelTableExporter.java
- complejidad.java
- convertir.java

A continuación se presenta el contenido de las clases:

Inicio:

```
public class inicio extends javax.swing.JFrame {
static int numberOfJobs;
static int numberOfMachines;
double alfa;
String dif="";
double k;
double niv;
int tp;
int m=0;
int [][] j_m;
int [][]j_p;
int [][] m_j;
int [][] m_p;
int [][]ordenada;
int [][]op;
int [][]dur;
```

```
String sFichero;
String fichero;
IntVariable makespan;

public inicio() {
  initComponents();
  jTable1.setVisible(false);
  jButton1.setVisible(false);
}
private void leerJss(){
  numberOfJobs= Integer.parseInt(jTextField1.getText());
  numberOfMachines=Integer.parseInt(jTextField2.getText());
  DefaultTableModel mi=(DefaultTableModel)this.jTable1.getModel();
  j_m= new int [numberOfJobs][numberOfMachines];
  j_p= new int [numberOfJobs][numberOfMachines];
  int fi=0;
  FileWriter fichero = null;
  PrintWriter pw = null;
  try
  {
   fichero = new FileWriter("c:\\JSSP\\conv\\problema.jss");
   pw = new PrintWriter(fichero);
   for (int i=0; i<numberOfJobs;i++){
    for (int j=0; j<numberOfMachines; j++){
     String ope=  (String) mi.getValueAt(fi,2);  //op j del trabajo i
     String du=  (String) mi.getValueAt(fi, 3);  //dur de la operacion
     j_m[i][j]=Integer.parseInt(ope);
     j_p[i][j]=Integer.parseInt(du);
     fi++;
     pw.print(j_m[i][j] + " "+ j_p [i][j] + " ");
     }
    pw.println();
   }
  } catch (Exception e) {
    e.printStackTrace();
  } finally {
  try {
   if (null != fichero)
     fichero.close();
  } catch (Exception e2) {
   e2.printStackTrace();
  }
}
  m_j= new int [numberOfJobs][numberOfMachines];
  j_p= new int [numberOfJobs][numberOfMachines];
  for (int i=0; i< numberOfJobs; i++){
   for (int j=0; j< numberOfMachines;j++){
    m=j_m[i][j];
    m_j[m][j] = j;
    m_p[m][i] = j_p[i][j];
   }
  }
}
private void jMenuItem1ActionPerformed(java.awt.event.ActionEvent evt) {
 numberOfJobs= Integer.parseInt(jTextField1.getText());
 numberOfMachines=Integer.parseInt(jTextField2.getText());
 String[] columnNames = {"Trabajo", "Operación", "Máquina", "Duración"};
 DefaultTableModel modelo = new DefaultTableModel();
 modelo.setColumnCount(4);
 int jo;
```

```
 int op;
 modelo.setColumnIdentifiers(columnNames);
 int filas= numberOfJobs*numberOfMachines;
 modelo.setRowCount(filas);
 int fi=0;
 for (int i=0; i<numberOfJobs;i++){
  jo=i+1;
  modelo.setValueAt(jo, fi, 0);
  for (int j=0; j<numberOfMachines;j++){
   op=j+1;
   modelo.setValueAt(op,fi,1);
   fi++;
  }
 }
 jTable1.setModel(modelo);
 jTable1.setVisible(true);
 }
}
private static boolean Fecha_Valida_Formato(String fecha) {
 try {
  SimpleDateFormat formatoFecha = new SimpleDateFormat("dd-MM-yyyy", Locale.getDefault());
  formatoFecha.setLenient(false);
  formatoFecha.parse(fecha);
 } catch (ParseException e) {
  return false;
 }
  return true;
 }
private void jMenuItem2ActionPerformed(java.awt.event.ActionEvent evt) {
 File fi= new File("C:\\JSSP\\java.xls");
 String nombreTab="JSSP";
 try
  {
  DataOutputStream out=new DataOutputStream(new FileOutputStream(fi));
  WritableWorkbook w = Workbook.createWorkbook(out);
  WritableSheet s = w.createSheet(nombreTab, 0);
  for(int i=0;i< jTable1.getColumnCount();i++){
   for(int j=0;j<jTable1.getRowCount();j++){
    Object objeto=jTable1.getValueAt(j,i);
    if (objeto==null)
    {
     objeto="";}
    s.addCell(new Label(i, j, String.valueOf(objeto))); }
   }
  w.write();
  w.close();
  out.close();
 }catch(IOException ex){ex.printStackTrace();}
 catch(WriteException ex){ex.printStackTrace();} }
private void jMenuItem3ActionPerformed(java.awt.event.ActionEvent evt) {
 File file=new File("C:\\JSSP\\java.xls");
 Vector headers = new Vector();
 Vector data = new Vector();
 try {
  Workbook workbook = Workbook.getWorkbook(file);
  Sheet sheet = workbook.getSheet(0);
  headers.clear();
  headers.add("Trabajo");
  headers.add("Operación");
  headers.add("Máquina");
```

```
 headers.add("Duración");
 data.clear();
 for (int j = 0; j < sheet.getRows(); j++) {
  Vector d = new Vector();
  for (int i = 0; i < sheet.getColumns(); i++) {
   Cell cell = sheet.getCell(i, j);
   d.add(cell.getContents());
  }
  d.add("\n");
  data.add(d);
 }
}
 catch (Exception e) {
  e.printStackTrace();
 }
 DefaultTableModel model = new DefaultTableModel(data,headers);
 model = new DefaultTableModel(data, headers);
 jTable1.setModel(model);
}
private void jMenuItem7ActionPerformed(java.awt.event.ActionEvent evt) {
 final IntervalCategoryDataset dataset = createDataset();
 final JFreeChart chart = createChart(dataset);
 final ChartPanel chartPanel = new ChartPanel(chart);
 JFrame frame = new JFrame("Gráfico");
 frame.setContentPane(chartPanel);
 frame.setExtendedState(MAXIMIZED_BOTH);
 frame.setVisible(true);
}
private void jMenuItem5ActionPerformed(java.awt.event.ActionEvent evt) {
 convertir co= new convertir();
 co.convertir(sFichero);
 int N=0;
 int M=0;
 File f = new File( "C:\\JSSP\\matlab\\jss.cnf" );
 BufferedReader entrada;
 try{
  entrada = new BufferedReader( new FileReader( f ) );
  String linea;
  linea = entrada.readLine();
  String[] separadas1 = linea.split(" ");
  N= Integer.parseInt(separadas1[2]);
  M=Integer.parseInt(separadas1[3]);
  JFrame frame = new JFrame("Codificación");
  JOptionPane.showMessageDialog(frame,"La cantidad de variables es: "+N +" y la de restricciones: "+M);
  alfa=M/N;
  int op1= Integer.parseInt(jTextField1.getText()) * Integer.parseInt (jTextField2.getText());
  if (op1<6){
   if (alfa < 1.5){
    dif="FÁCIL";
    k=0.0159;
    niv=1.8;
    tp=44;
   }
   else{
    if (alfa <2) {
     dif="Media";
     k=  0.0138;
     niv= 1.87;
     tp=60;
     }
```

```
    else{
     dif="Alta";
     k= 0.0201;
     niv= 1.7;
     tp=61;
    }
   }
  }
 else{
  if (op1<10){
   if (alfa < 2){
    dif="FÁCIL";
    k=0.223;
    niv=0.65;
    tp=41;
   }
   else{
    if (alfa< 2.5){
     dif="MEDIO";
     k=0.0185;
     niv=1.73;
     tp=49;
    }else{
      dif="DIFÍCIL"
      k=0.0179;
      niv=1.75;
      tp=59;
    }
   }
  }else{
    if (alfa < 2.5){
     dif="FÁCIL";
     k=0.0197;
     niv=1.7;
     tp=69;       }
     else{
      if (alfa< 2.5){
       dif="MEDIO";
       k=0.0230;
       niv=1.64;
       tp=51;
      }else{
        dif="DIFÍCIL";
        k=0.0139;
        niv=1.86;
        tp=66;
      }
    }
   }
}
 JFrame fr2=new JFrame("Dificultad");
 JOptionPane.showMessageDialog(fr2,"La dificultad de resolución es: "+ dif+", el valor en la escala de dificultad es: "+ niv +" dado por el ratio de escape del caos: "+ k);
 JOptionPane.showMessageDialog(fr2, "El tiempo de resolución promedio de acuerdo a su nivel de dificultad es: "+tp);
}catch (IOException e) {
 e.printStackTrace();
}
}
private void jMenuItem6ActionPerformed(java.awt.event.ActionEvent evt) {
```

```
try {
 MatlabProxyFactoryOptions options = new MatlabProxyFactoryOptions.Builder()
 .setUsePreviouslyControlledSession(true)
 .setHidden(true)
 .setMatlabLocation(null).build();
 MatlabProxyFactory factory = new MatlabProxyFactory(options);
 MatlabProxy proxy = factory.getProxy();
 proxy.eval("demo_jssp");
 convertir co= new convertir();
 float re[]=co.resultado(sFichero);
 DefaultTableModel modelo = (DefaultTableModel)jTable1.getModel();
 modelo.addColumn("INICIO");
 for (int i=1; i< re.length; i++){
  int g=i-1;
  modelo.setValueAt(re[i], g, 4);
 }
 jTable1.setModel(modelo);
 String nom[][]= new String[numberOfJobs][numberOfMachines];
 Long inicio[][]=new Long [numberOfJobs][numberOfMachines];
 Long fin[][]= new Long[numberOfJobs][numberOfMachines];
 modelo.addColumn("FIN");
 for (int i=0; i<numberOfJobs;i++){
  for (int o=0; o<numberOfMachines;o++){
   nom[i][o]=  (String) jTable1.getValueAt(i*numberOfMachines+o, 2);
   float val= (float) jTable1.getValueAt(i*numberOfMachines+o, 4);
   inicio[i][o]=(long) val;
   String val1=(String) jTable1.getValueAt(i*numberOfMachines+o, 3);
   float val2= Float.parseFloat(val1);
   fin [i][o]=(long) val2 + inicio[i][o];
   modelo.setValueAt(fin[i][o], i*numberOfMachines+o,5);
  }
 }
jTable1.setModel(modelo);
 } catch (MatlabInvocationException ex) {
  Logger.getLogger(inicio.class.getName()).log(Level.SEVERE, null, ex);
 } catch (MatlabConnectionException ex) {
  Logger.getLogger(inicio.class.getName()).log(Level.SEVERE, null, ex);
 }
 jButton1.setVisible(true);
  }
private void jMenuItem9ActionPerformed(java.awt.event.ActionEvent evt) {
 leerTabla();        // TODO add your handling code here:
 }
private void jMenuItem4ActionPerformed(java.awt.event.ActionEvent evt) {
 Date ahora=new Date();
 SimpleDateFormat formateador=new SimpleDateFormat("dd-MM-HH-mm");
 Calendar calendarioAhora = Calendar.getInstance();
 ahora = calendarioAhora.getTime();
 String fh=formateador.format(ahora);
 sFichero = "C:\\JSSP\\problema"+numberOfJobs+"-"+numberOfMachines+"-"+fh+".csp";
 File fichero = new File(sFichero);
  if (fichero.exists()){
   System.out.println("Ya existe");
  }else {
 try{
  BufferedWriter bw =    new BufferedWriter(new FileWriter(sFichero));
  bw.write("; JSS \n");
  bw.write("; "+numberOfJobs+" "+numberOfMachines);
  for (int i=0; i<numberOfJobs;i++){
   bw.write("\n; ");
```

```
   for (int j=0; j<numberOfMachines; j++){
    bw.write (op[i][j]+" "+dur[i][j]+ " ");
   }
  }
  bw.write("\n");
  bw.write("(int makespan "+jTextField3.getText()+" "+   jTextField4.getText()+")\n(objective minimize
makespan)\n");
  for (int i=0; i<numberOfJobs;i++){
   for (int o=0; o<numberOfMachines;o++){
    bw.write("(int s_"+i+"_"+o+" 0 "+ jTextField4.getText()+") \n");
   }
  }
  for (int i=0; i<numberOfJobs;i++){
   for (int o=0; o<numberOfMachines;o++){
    int y=o+1;
    if (y==numberOfMachines){
     bw.write("(<= (+ s_"+i+"_"+o+" "+dur[i][o]+ ")makespan) \n");
    }else
     bw.write("(<= (+ s_"+i+"_"+o+" "+dur[i][o]+ ") s_"+i+"_"+y+") \n");
    }
   }
   for (int m=0; m<numberOfMachines;m++){
    for (int i0=0; i0<numberOfJobs;i0++){
     for (int i1=i0+1; i1<numberOfJobs;i1++){
      int j0=m_j[m][i0];
      int j1=m_j[m][i1];
      String s0="s_"+i0+"_"+j0;
      String s1="s_"+i1+"_"+j1;
      int p0=dur[i0][j0];
      int p1=dur[i1][j1];
      bw.write("(or (<= (+ "+s0+" "+ p0+") "+s1+") (<= (+ "+s1+" "+p1+") "+s0+"))\n");
    }
   }
  }
  bw.close();} catch (IOException ioe){
  ioe.printStackTrace();
  }
  }
private void jMenuItem8ActionPerformed(java.awt.event.ActionEvent evt) {
 long max=0;
 obtenerFechas();
 java.text.SimpleDateFormat sdf=new java.text.SimpleDateFormat("dd-MM-yyyy");
 DefaultTableModel modelo= (DefaultTableModel)jTable1.getModel();
 String fei1= (String) modelo.getValueAt(0,6);
 String fef1=(String)modelo.getValueAt(0,7);
 Date fei=null;
 Date fef=null;
 try {
  fei = sdf.parse(fei1);
  fef= sdf.parse(fef1);
 } catch (ParseException ex) {
   ex.printStackTrace();
 }
 modelo = (DefaultTableModel)jTable1.getModel();
 for (int i=0; i<modelo.getRowCount();i++){
  try {
   long dura= (long) modelo.getValueAt(i,5);
   String tti1= (String)modelo.getValueAt(i,6);
   Date tti=sdf.parse(tti1);
   String ttf1=(String)modelo.getValueAt(i,7);
```

```
   Date ttf= sdf.parse(ttf1);
   if (dura>max)
    max=dura;
    if (tti.before(fei)){
     fei=tti;
    }
    if (ttf.after(fef)){
     fef=ttf;
    }
   } catch (ParseException ex) {
    Logger.getLogger(inicio.class.getName()).log(Level.SEVERE, null, ex);
   }
  }
  jTextField5.setText(String.valueOf(max));
  jTextField6.setText(sdf.format(fei));
  jTextField7.setText(sdf.format(fef));}
private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
 complejidad com= new complejidad();
 com.jTextField5.setText(jTextField1.getText());
 com.jTextField4.setText(jTextField2.getText());
 com.jTextField3.setText(String.valueOf( Integer.parseInt (jTextField1.getText()) * Integer.parseInt (jText-
Field2.getText())));
 com.jTextField9.setText(String.valueOf(k));
 com.jTextField10.setText(String.valueOf(niv));
 com.jTextField11.setText(String.valueOf(tp));
 com.jTextField12.setText(dif);
 com.setVisible(true);
}
public void obtenerFechas(){
 String ax = JOptionPane.showInputDialog("Ingrese la fecha de comienzo (formato dd-MM-yyyy)");
 boolean val=Fecha_Valida_Formato(ax);
 if (val==false){
  JOptionPane.showMessageDialog(this,"El formato de la fecha ingresada es incorrecto");
 }else{
  SimpleDateFormat formateador = new SimpleDateFormat("dd-MM-yyyy");
  Date fecha=   null;
  try {
   fecha = formateador.parse(ax);
   } catch (ParseException ex) {
     ex.printStackTrace();
   }
   DefaultTableModel modelo = (DefaultTableModel)jTable1.getModel();
   modelo.addColumn("Fecha inicio");
   modelo.addColumn ("Fecha fin");
   Calendar cal=new GregorianCalendar();
   jTable1.setModel(modelo);
   for (int i=0; i<modelo.getRowCount();i++){
    float tti=(float) modelo.getValueAt(i,4);
    long ttf= (long) modelo.getValueAt(i,5);
    int inicio=(int) tti;
    int fin = (int) ttf;
    cal.setTime(fecha);
    cal.add(Calendar.DATE, inicio);
    modelo.setValueAt(formateador.format(cal.getTime()), i, 6);
    cal.setTime(fecha);
    cal.add(Calendar.DATE, fin);
    modelo.setValueAt(formateador.format(cal.getTime()),i,7);
   }
   jTable1.setModel(modelo);
  }
```

```
  }
public IntervalCategoryDataset createDataset() {
 DefaultTableModel modelo = (DefaultTableModel)jTable1.getModel();
 String nom[][]= new String[numberOfJobs][numberOfMachines];
 Date fein[][]=new Date [numberOfJobs][numberOfMachines];
 Date fife[][]= new Date[numberOfJobs][numberOfMachines];
 for (int i=0; i<numberOfJobs;i++){
  for (int o=0; o<numberOfMachines;o++){
   try {
    nom[i][o]=  (String) jTable1.getValueAt(i*numberOfMachines+o, 2);
    String ut1=(String) jTable1.getValueAt (i*numberOfMachines+o, 6);
    DateFormat df = new SimpleDateFormat("dd-MM-yyyy");
    fein[i][o]= df.parse(ut1);
    ut1=(String)jTable1.getValueAt (i*numberOfMachines+o, 7);
    fife[i][o]= df.parse(ut1);
   } catch (ParseException ex) {
    Logger.getLogger(inicio.class.getName()).log(Level.SEVERE, null, ex);
   }
  }}
  final TaskSeries [] s=new TaskSeries [fein.length];
  for (int i=0; i<nom.length;i++){
   int d=i+1;
   s[i]= new TaskSeries("Trabajo "+d);
   for (int j=0; j< nom[i].length;j++){
    String nomb= "Máquina "+nom[i][j];
    s[i].add(new Task (nomb, new SimpleTimePeriod (fein[i][j], fife[i][j])));
   }
   }
   final TaskSeriesCollection collection = new TaskSeriesCollection();
   for (int j=0; j<fein.length;j++ ){
    collection.add(s[j]);
    }
   return collection;
   }
private JFreeChart createChart(final IntervalCategoryDataset dataset) {
  final JFreeChart chart = ChartFactory.createGanttChart(
  "Diagrama de GANTT del Job Shop Scheduling",  // chart title
  "Tareas",             // domain axis label
  "Duración",           // range axis label
  dataset,              // data
  true,                 // include legend
  true,                 // tooltips
  false                 // urls
  );          chart.getCategoryPlot().getDomainAxis().setMaxCategoryLabelWidthRatio(10.0f);
  return chart;
 }
private void leerTabla(){
 numberOfJobs= Integer.parseInt(jTextField1.getText());
 numberOfMachines=Integer.parseInt(jTextField2.getText());
 DefaultTableModel mi=(DefaultTableModel)this.jTable1.getModel();
 op= new int[numberOfJobs][numberOfMachines];
 dur=new int[numberOfJobs][numberOfMachines];
 int  fi=0;
 for (int i=0; i<numberOfJobs;i++){
  for (int j=0; j<numberOfMachines; j++){
   if(mi.getValueAt(fi,2)==null ||mi.getValueAt(fi,3)==null ){
    JOptionPane.showMessageDialog(this,"Debe suministrar toda la informacion solicitada");
    return;
   }
   String ope=  (String) mi.getValueAt(fi,2);
```

```
  String du=  (String) mi.getValueAt(fi, 3);
  if (Integer.parseInt(ope)>numberOfMachines){
   JOptionPane.showMessageDialog(this,"Un valor asignado es superior a las máquinas disponibles");
   return;
  }
  op[i][j]=Integer.parseInt(ope);
  dur[i][j]=Integer.parseInt(du);
  fi++;
  }
 }
 int sum[]=new int[numberOfJobs];
 for (int v=0;v<numberOfJobs;v++){
  for (int d=0;d<numberOfMachines;d++){
   sum[v]=sum[v]+dur[v][d];
  }
 }
 int maximum = sum[0];   // start with the first value
 for (int i=1; i<sum.length; i++) {
  if (sum[i] > maximum) {
   maximum = sum[i];   // new maximum
  }
 }
 int sum1[]=new int[numberOfMachines];
 for (int v=0;v<numberOfMachines;v++){
  for (int d=0;d<numberOfJobs;d++){
   sum1[v]=sum1[v]+dur[d][v];
  }
 }
 for (int i=1; i<sum.length; i++) {
  if (sum1[i] > maximum) {
   maximum = sum1[i];   // new maximum
  }
 }
 jTextField3.setText(String.valueOf(maximum));
 int vm[]=new int [numberOfMachines];
 for (int v=0;v<numberOfMachines;v++){
  for (int d=0;d<numberOfJobs;d++){
   if (vm[v]<dur[d][v]){
    vm[v]=dur[d][v];
   }
  }
 }
 ordenada=new int[numberOfJobs][numberOfMachines];
 for (int i=0; i<numberOfJobs;i++){
  for (int o=0; o<numberOfMachines;o++){
   for (int m=0; m< numberOfMachines;m++){
    if (op[i][m]==(o+1))
    {
     ordenada[i][o]=dur[i][m];
    }
   }
  }
 }
 int[] max=new int[numberOfMachines];
 max[0]=0;
 for (int i=0; i<numberOfMachines;i++){
  for (int j=0; j<numberOfJobs;j++){
   if(ordenada[j][i]>max[i])
   {
   max[i]=ordenada[j][i];
```

```
    }
   }}
  m=0;
  for (int i=0; i<max.length; i++) {
   m= m+max[i];
   }
  m_j=new int[numberOfJobs][numberOfMachines];
  m_p=new int[numberOfJobs][numberOfMachines];
  for (int i=0; i<numberOfJobs;i++) {
   for (int j=0; j<numberOfMachines;j++) {
    int m = op[i][j];
    m_j[m][i] = j;
    m_p[m][i] = dur[i][j];
  } } }
public static void main(String args[]) {
 try {
  for (javax.swing.UIManager.LookAndFeelInfo info : javax.swing.UIManager.getInstalledLookAndFeels())
{
   if ("Nimbus".equals(info.getName())) {
    javax.swing.UIManager.setLookAndFeel(info.getClassName());
    break;
   }
  }
} catch (ClassNotFoundException ex) {
java.util.logging.Logger.getLogger(inicio.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
 } catch (InstantiationException ex) {
 java.util.logging.Logger.getLogger(inicio.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
 } catch (IllegalAccessException ex) {
  Java.util.logging.Logger.getLogger(inicio.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
 } catch (javax.swing.UnsupportedLookAndFeelException ex) {
  java.util.logging.Logger.getLogger(inicio.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
 }
 java.awt.EventQueue.invokeLater(new Runnable() {
public void run() {
 new inicio().setVisible(true);}  });
}
```

Convertir

```
public class convertir {
 CSP csp = null;
 boolean prolog = false;
 boolean maxCSP = false;
 boolean weightedCSP = false;
 boolean competition = false
 boolean incremental = false;
 boolean propagation = true;
 boolean simplify_clauses = true;
 public static int debug = 0;
 public void convertir(String file){
  try {
   String inputFileName=file;
   String nombre=  file.substring(0, file.length()-4);
   String outputFileName=nombre+"-1.csp"
   String satFileName="C:\\JSSP\\matlab\\jss.cnf";
   File fichero = new File(satFileName);
   fichero.delete();
   String mapFileName=nombre+".map";
   String cspFileName = outputFileName;
   String outputHook=null;
```

```
  String format="csp";
  outputCSP(inputFileName, outputFileName,format,outputHook );
  encode(cspFileName, satFileName, mapFileName);
  Logger.status();
 } catch (SugarException ex) {
  java.util.logging.Logger.getLogger(convertir.class.getName()).log(Level.SEVERE, null, ex)
 } catch (IOException ex) {
  java.util.logging.Logger.getLogger(convertir.class.getName()).log(Level.SEVERE, null, ex);
} catch (ClassNotFoundException ex) {
  java.util.logging.Logger.getLogger(convertir.class.getName()).log(Level.SEVERE, null, ex);
 } catch (InstantiationException ex) {
 java.util.logging.Logger.getLogger(convertir.class.getName()).log(Level.SEVERE, null, ex);
} catch (IllegalAccessException ex) {
 java.util.logging.Logger.getLogger(convertir.class.getName()).log(Level.SEVERE, null, ex); }}
public float[] resultado (String file){
 float res[]=null;
 try {
  String nombre=  file.substring(0, file.length()-4)
  String outFileName="C:\\JSSP\\matlab\\resultado.txt";
  String mapFileName=nombre+".map";
  res=decode(outFileName, mapFileName);
  } catch (SugarException ex) {
  java.util.logging.Logger.getLogger(convertir.class.getName()).log(Level.SEVERE, null, ex);
  } catch (IOException ex) {
  java.util.logging.Logger.getLogger(convertir.class.getName()).log(Level.SEVERE, null, ex)
  }
  return res;
  }
public void translate(String cspFileName) throws SugarException, IOException {
Logger.fine("Parsing " + cspFileName);
InputStream in;
if (cspFileName.endsWith(".gz")) {
 in = new GZIPInputStream(new FileInputStream(cspFileName));
} else {
  in = new FileInputStream(cspFileName);}
 BufferedReader reader = new BufferedReader(new InputStreamReader(in, "UTF-8"));
 Parser parser = new Parser(reader, prolog);
 List<Expression> expressions = parser.parse();
 reader.close();
 reader = null;
 in.close();
 in = null;
 parser = null;
 Runtime.getRuntime().gc();
 Logger.info("parsed " + expressions.size() + " expressions");
 if (debug > 0) {
  for (Expression x : expressions)
   System.out.println("c " + x);
  }
 Logger.status();
       // Conversion
 Logger.fine("Converting to clausal form CSP");
 csp = new CSP();
 Converter converter = new Converter(csp);
 Converter.INCREMENTAL_PROPAGATION = propagation;
 converter.convert(expressions);
 converter = null;
 expressions = null;
 Expression.clear();
 Runtime.getRuntime().gc();
```

```
 Logger.fine("CSP : " + csp.summary());
 Logger.status();
 if(propagation) {
 // Propagation
  Logger.fine("Propagation in CSP");
  csp.propagate();
  Logger.fine("CSP : " + csp.summary());
  if (debug > 0) {
  csp.output(System.out, "c ");
  }
  Logger.status();
}
  if (csp.isUnsatisfiable()) {
   Logger.info("CSP is unsatisfiable after propagation");
   JOptionPane.showMessageDialog(null, "El problema no es satisfactible");
   Logger.println("s UNSATISFIABLE");
            }
            if (simplify_clauses) {
               // Simplification
   Logger.fine("Simplifing CSP clauses by introducing new Boolean variables");
  csp.simplify();
  Logger.info("CSP : " + csp.summary());
  if (debug > 0) {
   csp.output(System.out, "c ");
}
   Logger.status();
 }
   Runtime.getRuntime().gc();        }
public void encode(String cspFileName, String satFileName, String mapFileName)  throws SugarException,
IOException {
  translate(cspFileName);
  if (csp.isUnsatisfiable()) {
   Logger.info("CSP is unsatisfiable after propagation");
   Logger.println("s UNSATISFIABLE");
    return;
  }
   Logger.fine("Encoding CSP to SAT : " + satFileName);
   Encoder encoder = new Encoder(csp);
   encoder.encode(satFileName);
   Logger.fine("Writing map file : " + mapFileName);
    encoder.outputMap(mapFileName);
   Logger.status();
   Logger.info("SAT : " + encoder.summary());
   }
public float[] decode(String outFileName, String mapFileName)
throws SugarException, IOException {
 float[] sn = null;
 String [] sv;
 int cont=0;
 CSP csp = new CSP();
 List<String> objectiveVariableNames = null;
 BufferedReader rd = new BufferedReader(
 new InputStreamReader(new FileInputStream(mapFileName), "UTF-8"));
 while (true) {
  String line = rd.readLine();
  if (line == null)
   break;
  String[] s = line.split("\\s+");
  if (s[0].equals("objective")) {
  if (s[1].equals(SugarConstants.MINIMIZE)) {
```

```
 csp.setObjective(CSP.Objective.MINIMIZE);
 } else if (s[1].equals(SugarConstants.MAXIMIZE)) {
  csp.setObjective(CSP.Objective.MAXIMIZE);
}
 objectiveVariableNames = new ArrayList<String>();
 for (int i = 2; i < s.length; i++)
  objectiveVariableNames.add(s[i]);
 } else if (s[0].equals("int")) {
 String name = s[1];
 int code = Integer.parseInt(s[2]);
 IntegerDomain domain = null;
 if (s.length == 4) {
  int lb;
  int ub;
  int pos = s[3].indexOf("..");
  if (pos < 0) {
   lb = ub = Integer.parseInt(s[3]);
  } else {
   lb = Integer.parseInt(s[3].substring(0, pos));
   ub = Integer.parseInt(s[3].substring(pos+2));
}
  domain = new IntegerDomain(lb, ub);
} else {
  SortedSet<Integer> d = new TreeSet<Integer>();
  for (int i = 3; i < s.length; i++) {
   int lb;
   int ub;
   int pos = s[i].indexOf("..");
   (pos < 0) {
   lb = ub = Integer.parseInt(s[i]);
  } else {
   lb = Integer.parseInt(s[i].substring(0, pos));
   ub = Integer.parseInt(s[i].substring(pos+2));
 }
  for (int value = lb; value <= ub; value++) {
   d.add(value);}}
  domain = new IntegerDomain(d);
}
  IntegerVariable v = new IntegerVariable(name, domain);
  v.setCode(code);
  csp.add(v);
  cont++;
  } else if (s[0].equals("bool")) {
  String name = s[1];
  int code = Integer.parseInt(s[2]);
  booleanVariable v = new BooleanVariable(name);
  v.setCode(code);
  csp.add(v);
 }}
 rd.close();
 if (objectiveVariableNames != null) {
  List<IntegerVariable> vs = new ArrayList<IntegerVariable>();
  for (String name : objectiveVariableNames) {
   IntegerVariable v = csp.getIntegerVariable(name);
   if (v == null)
    throw new SugarException("Unknown objective variable " + name);
    vs.add(v);
 }
  csp.setObjectiveVariables(vs);}
  Encoder encoder = new Encoder(csp);
```

```
 if (encoder.decode(outFileName)) {
  sn=new float [cont];
  if (csp.getObjectiveVariables() == null) {
 } else {
  String s = "o";
  int vu=0;
  for (IntegerVariable v : csp.getObjectiveVariables()) {
   String name = v.getName();
   int value = v.getValue();
   s += " " + value;
  }}
 if (competition) {
  for (IntegerVariable v : csp.getIntegerVariables()) {
   if (! v.isAux() && ! v.getName().startsWith("_")) {
 }}
 } else {
  int vu=0;
  for (IntegerVariable v : csp.getIntegerVariables()) ;
   if (! v.isAux() && ! v.getName().startsWith("_")) {
    sn[vu]=v.getValue();
    vu++;
   }}}
 } else {}
 return sn;
}
public void outputCSP(String inputFileName, String outputFileName, String format, String outputHook)
throws SugarException, IOException, ClassNotFoundException, InstantiationException, IllegalAccessEx-
ception {
 Logger.info("Translate CSP to " + format);
 translate(inputFileName);
 PrintWriter out = null;
 if (outputFileName != null)
 out = new PrintWriter(new BufferedWriter(new FileWriter(outputFileName)));
 OutputInterface output;
 if (outputHook == null) {
  output = new Output();
 } else {
 Class<?> clazz = Class.forName(outputHook);
 output = (OutputInterface)clazz.newInstance();
 }
 output.setCSP(csp);
 output.setOut(out);
 output.setFormat(format);
 output.output();
 Logger.status();
 }}
```

Complejidad

```
public class complejidad extends javax.swing.JFrame {

    /**
     * Creates new form complejidad
     */
    public complejidad() {
        initComponents();
        leerDatos();
    }
private void leerDatos(){
    File file=new File("C:\\JSSP\\matlab\\complejidad.xls");
```

```
Vector data = new Vector();
try {
Workbook workbook = Workbook.getWorkbook(file);
Sheet sheet = workbook.getSheet(0);
data.clear();
String datos[]= new String[5];
for (int j = 0; j < sheet.getRows(); j++) {
Vector d = new Vector();
for (int i = 0; i < sheet.getColumns(); i++) {
Cell cell = sheet.getCell(i, j);
datos[i]=String.valueOf(cell.getContents());
}
d.add("\n");
data.add(d);
}
 jTextField2.setText(datos[0]);
 jTextField1.setText(datos[1]);
 jTextField6.setText(datos[2]);
 jTextField7.setText(datos[3]);
 jTextField8.setText(datos[4]);
 }
 catch (Exception e) {
  e.printStackTrace();
}
}
public void run() {
  new complejidad().setVisible(true); }
 });
}
```

10.4 Apéndice D: Conversión de un problema

En esta sección se muestran los distintos archivos que surgen luego de la conversión y resolución del siguiente problema:

Trabajo	Operación	Máquina	Duración
1	1	1	2
	2	0	1
	3	2	2
2	1	1	3
	2	2	1
	3	0	2
3	1	0	4
	2	2	1
	3	1	2

Límites Makespan: [7,12]

Conversión a CSP:

```
; JSS
; 3 3
; 1 2 0 1 2 2
; 1 3 2 1 0 2
; 0 4 2 1 1 2
(int makespan 7 12)
(objective minimize makespan)
(int s_0_0 0 12)
(int s_0_1 0 12)
(int s_0_2 0 12)
(int s_1_0 0 12)
(int s_1_1 0 12)
(int s_1_2 0 12)
(int s_2_0 0 12)
(int s_2_1 0 12)
(int s_2_2 0 12)
(<= (+ s_0_0 2) s_0_1)
(<= (+ s_0_1 1) s_0_2)
(<= (+ s_0_2 2)makespan)
(<= (+ s_1_0 3) s_1_1)
(<= (+ s_1_1 1) s_1_2)
(<= (+ s_1_2 2)makespan)
(<= (+ s_2_0 4) s_2_1)
(<= (+ s_2_1 1) s_2_2)
(<= (+ s_2_2 2)makespan)
(or (<= (+ s_0_1 1) s_1_2) (<= (+ s_1_2 2) s_0_1))
(or (<= (+ s_0_1 1) s_2_0) (<= (+ s_2_0 4) s_0_1))
(or (<= (+ s_1_2 2) s_2_0) (<= (+ s_2_0 4) s_1_2))
(or (<= (+ s_0_0 2) s_1_0) (<= (+ s_1_0 3) s_0_0))
(or (<= (+ s_0_0 2) s_2_2) (<= (+ s_2_2 2) s_0_0))
(or (<= (+ s_1_0 3) s_2_2) (<= (+ s_2_2 2) s_1_0))
(or (<= (+ s_0_2 2) s_1_1) (<= (+ s_1_1 1) s_0_2))
(or (<= (+ s_0_2 2) s_2_1) (<= (+ s_2_1 1) s_0_2))
(or (<= (+ s_1_1 1) s_2_1) (<= (+ s_2_1 1) s_1_1))
```

Archivo de mapeo:

```
objective minimize makespan
int makespan 1 7..12
int s_0_0 6 0..7
int s_0_1 13 2..9
int s_0_2 20 3..10
int s_1_0 27 0..6
int s_1_1 33 3..9
int s_1_2 39 4..10
int s_2_0 45 0..5
int s_2_1 50 4..9
int s_2_2 55 5..10
```

Archivo en CNF

```
p cnf 77 196
-1 2 0
-2 3 0
-3 4 0
-4 5 0
-6 7 0
-7 8 0
-8 9 0
-9 10 0
-10 11 0
-11 12 0
-13 14 0
-14 15 0
-15 16 0
-16 17 0
-17 18 0
-18 19 0
-20 21 0
-21 22 0
-22 23 0
-23 24 0
-24 25 0
-25 26 0
-27 28 0
-28 29 0
-29 30 0
-30 31 0
-31 32 0
-33 34 0
-34 35 0
-35 36 0
-36 37 0
-37 38 0
-39 40 0
-40 41 0
-41 42 0
-42 43 0
-43 44 0
-45 46 0
-46 47 0
-47 48 0
-48 49 0
-50 51 0
-51 52 0
```

```
-52 53 0
-53 54 0
-55 56 0
-56 57 0
-57 58 0
-58 59 0
6 -13 0
7 -14 0
8 -15 0
9 -16 0
10 -17 0
11 -18 0
12 -19 0
13 -20 0
14 -21 0
15 -22 0
16 -23 0
17 -24 0
18 -25 0
19 -26 0
-1 22 0
-2 23 0
-3 24 0
-4 25 0
-5 26 0
27 -33 0
28 -34 0
29 -35 0
30 -36 0
31 -37 0
32 -38 0
33 -39 0
34 -40 0
35 -41 0
36 -42 0
37 -43 0
38 -44 0
-1 40 0
-2 41 0
-3 42 0
-4 43 0
-5 44 0
45 -50 0
46 -51 0
47 -52 0
48 -53 0
49 -54 0
50 -55 0
51 -56 0
52 -57 0
53 -58 0
54 -59 0
-1 55 0
-2 56 0
-3 57 0
-4 58 0
-5 59 0
-39 14 -60 0
-40 15 -60 0
-41 16 -60 0
```

```
-42 17 -60 0
-43 18 -60 0
-44 19 -60 0
-16 -61 0
39 -17 -61 0
40 -18 -61 0
41 -19 -61 0
42 -61 0
60 61 0
-47 -62 0
-48 13 -62 0
-49 14 -62 0
15 -62 0
-14 -63 0
45 -15 -63 0
46 -16 -63 0
47 -17 -63 0
48 -18 -63 0
49 -19 -63 0
62 63 0
-64 0
45 -39 -65 0
46 -40 -65 0
47 -41 -65 0
48 -42 -65 0
49 -43 -65 0
64 65 0
-28 -66 0
-29 6 -66 0
-30 7 -66 0
-31 8 -66 0
-32 9 -66 0
10 -66 0
-8 -67 0
27 -9 -67 0
28 -10 -67 0
29 -11 -67 0
30 -12 -67 0
31 -67 0
66 67 0
-55 9 -68 0
-56 10 -68 0
-57 11 -68 0
-58 12 -68 0
-12 -69 0
55 -69 0
68 69 0
-55 29 -70 0
-56 30 -70 0
-57 31 -70 0
-58 32 -70 0
-71 0
70 71 0
-34 -72 0
-35 20 -72 0
-36 21 -72 0
-37 22 -72 0
-38 23 -72 0
24 -72 0
-20 -73 0
```

```
33 -21 -73 0
34 -22 -73 0
35 -23 -73 0
36 -24 -73 0
37 -25 -73 0
38 -26 -73 0
72 73 0
-50 -74 0
-51 20 -74 0
-52 21 -74 0
-53 22 -74 0
-54 23 -74 0
24 -74 0
-21 -75 0
50 -22 -75 0
51 -23 -75 0
52 -24 -75 0
53 -25 -75 0
54 -26 -75 0
74 75 0
-50 33 -76 0
-51 34 -76 0
-52 35 -76 0
-53 36 -76 0
-54 37 -76 0
38 -76 0
-34 -77 0
50 -35 -77 0
51 -36 -77 0
52 -37 -77 0
53 -38 -77 0
54 -77 0
76 77 0
```

Archivo de resultados:

```
SAT
-1 -2 -3 4 5 -6 7 8 9 10 11 12
-13 -14 15 16 17 18 19 -20 -21 -22 -23 24
25 26 -27 -28 -29 30 31 32 -33 -34 -35 36
37 38 -39 -40 -41 -42 43 44 45 46 47 48
49 -50 51 52 53 54 -55 -56 57 58 59 60
-61 -62 63 -64 65 66 -67 68 -69 70 -71 -72
73 -74 75 -76 77
```