Instituto Superior Politécnico José Antonio Echeverría
Facultad de Ingeniería Industrial

Módulo de planificación de secuencias de ensamble para el sistema Galba-CAD

Tesis presentada en opción al título de

Máster en Tecnologías de Apoyo a la Toma de Decisiones

Autor: Ing. Maybel Díaz Capote

Tutores: Dra. Edith Martínez Delgado

Dra. Rosario Garza Ríos

MSc. Hassán Lombera Rodríguez

La Habana, 2013

"Lo que está bien puesto en el juicio será, de seguro, bien puesto en los labios".

José Martí

Dedicatoria

A mis padres quienes sin escatimar esfuerzo sacrificaron su vida para educarme y hacerme feliz. A mi hermano, para quien la ilusión de su vida ha sido verme convertida en una profesional.

Agradecimientos

Quisiera agradecer en primer lugar a Hassán por ilustrarme el camino de la superación y estar a mi lado para ayudarme a transitarlo. Por su confianza, por ser mi ejemplo a seguir como profesional, por ser mi tutor personal y por acompañarme en cada momento de debilidad. Le estaré eternamente agradecida por las interminables noches corrigiendo artículos y documentos, por sus críticas, que aunque a veces fueron duras me sirvieron para tomar el camino correcto. Por su amor incondicional y su paciencia.

A mis tutoras Edith y Rosario, por su confianza, sus sugerencias, sus conocimientos y su tiempo. Por no desistir ante tanta indecisión y cambios de temas y por su disposición en asesorar esta nueva investigación.

A todos los profesores de la CUAE que tuvieron que ver con mi formación.

A mi mamá y a mi papá, artífices de mi vida. Sé que no existirá forma alguna de agradecer una vida de sacrificios, esfuerzos y amor; gracias por ser tan preocupados y estar tan pendientes de mí en todo momento. Con su apoyo y aliento hoy he logrado uno de mis más grandes anhelos. A mi hermano, por ser incondicional y apoyarme en todas mis decisiones. A estas tres grandes personas de mi vida, sé que jamás encontraré la forma de agradecer su constante apoyo y confianza. Solo espero que comprendan que mis ideales, esfuerzos y logros han sido también suyos e inspirados en ustedes. Gracias a todos por permitir que este sueño se haya convertido en una realidad.

Maybel Díaz Capote

Resumen

El proyecto Centro de Diseño y Simulación de Estructuras Mecánicas de la Universidad de las Ciencias Informáticas construye actualmente una infraestructura tecnológica denominada Galba-CAD. Esta infraestructura, basada en *software* libre, brinda soporte al funcionamiento de la Gerencia de Ingeniería y Diseño de la Empresa Socialista de Capital Mixto Industria China Venezolana de Taladros, S.A. Dentro de los procesos de producción de la Gerencia de Ingeniería y Diseño se encuentra con alta relevancia la Planificación de Secuencias de Ensamble de taladros de perforación de petróleo. Cuando se realiza el ensamble de un taladro sin un estudio previo el tiempo de ensamble aumenta, debido fundamentalmente, al número elevado de reorientaciones de piezas y cambios de herramientas que pueden ocurrir. De aquí, que el estudio y las acciones que se adopten con el fin de atender este problema, tengan una gran importancia para esta nueva empresa. La presente investigación persigue así, desarrollar un módulo que minimice la cantidad de reorientaciones y los cambios de herramientas para disminuir el número total de operaciones durante el proceso de ensamble. Las primeras constituyen las restricciones de optimización fundamentales abordadas en el trabajo. Además, se consideran en la investigación seis direcciones posibles de ensamble a lo largo de los tres ejes principales ($\pm x, \pm y, \pm z$), conjuntamente con dos tipos de restricciones absolutas: geométricas y de precedencia. Cada componente se desensambla con una sola herramienta. El principal aporte del trabajo lo constituye la resolución del problema planteado mediante la técnica metaheurística Sistema de Hormigas $\mathcal{MAX}$-$\mathcal{MIN}$. De igual manera constituye un aporte la inclusión de los cambios de herramientas en la regla probabilística que utiliza esta técnica, así como la nueva interpretación de un término en la regla de actualización. Se comprueba el desempeño del algoritmo propuesto mediante casos de estudio recogidos en revistas referenciadas del tema. Adicionalmente, se utilizan casos de estudio relativos a un taladro de perforación de petróleo de 2000 HP, cortesía del proyecto. Como conclusión fundamental se deriva que la inclusión de los cambios de herramientas en la regla probabilística de la la técnica metaheurística Sistema de Hormigas $\mathcal{MAX}$-$\mathcal{MIN}$ resulta factible. Asimismo, se concluye que el sentido práctico de las secuencias obtenidas con el algoritmo propuesto puede mejorar cuando se tienen en cuenta las restricciones de cambios de herramientas.

Palabras clave: cambios de herramientas, metaheurísticas, número de reorientaciones, optimización con restricciones, Planificación de Secuencias de Ensamble, Sistema de Hormigas $\mathcal{MAX}$-$\mathcal{MIN}$

Índice general

Índice de figuras

Índice de tablas

La República Bolivariana de Venezuela se ha propuesto lograr una independencia petrolera y tecnológica que le permita explotar sus reservas de hidrocarburos de forma soberana. Uno de los frentes de trabajo es la producción de taladros de perforación de petróleo, por los elevados costos de alquiler y mantenimiento de estos equipos en el mercado internacional. Para ello se ha creado la Empresa Socialista de Capital Mixto Industria China Venezolana de Taladros, S.A (ICVT) que permitirá producir estos taladros en el territorio venezolano. El objetivo general de la ICVT consiste en diseñar, fabricar, ensamblar y probar taladros con sus componentes para perforación en tierra. Estos taladros contarán con capacidades desde 750 hasta 3000 HP. La Gerencia de Ingeniería y Diseño (GID) de la ICVT tiene la misión de proponer variantes de diseño a los modelos de taladros originales, de forma que se adapten a las condiciones específicas del entorno venezolano. Para estas actividades se emplean comúnmente tecnologías de Diseño Asistido por Computadoras (CAD) e Ingeniería Asistida por Computadoras (CAE). En el proyecto Centro de Diseño y Simulación de Estructuras Mecánicas (CDSEM) de la Universidad de las Ciencias Informáticas (UCI) se construye actualmente una infraestructura tecnológica denominada Galba-CAD. Esta infraestructura, basada en *software* libre, brinda soporte al funcionamiento de la GID. Sobre esta plataforma se modela actualmente y se comprobará en un futuro el funcionamiento de los componentes de los taladros de perforación que serán fabricados en Venezuela.

El ciclo básico de necesidades de la GID comprende el diseño de partes, ensambles, análisis numérico y la generación de los planos técnicos. Una vez que este ciclo concluye, el flujo de trabajo demanda una etapa donde se planifican los procesos de producción. Dentro de estos procesos se encuentra con alta relevancia la Planificación de Secuencias de Ensamble (ASP) de estructuras mecánicas. Esta planificación influye directamente en los tiempos y costos de producción, principalmente cuando cientos de piezas están involucradas en el proceso.

Situación problémica

Actualmente en la ICVT se ensamblan taladros de perforación de petróleo fabricados en su totalidad en China. El procedimiento de ensamble es con frecuencia un alto consumidor de tiempo debido a las siguientes deficiencias:

- La planificación del ensamble se realiza manualmente, por lo que resulta poco eficiente.
- Se realiza un número elevado de reorientaciones de piezas y subensambles. Cada reorientación supone un tiempo adicional, debido a la utilización de maquinarias pesadas en el proceso.
- Se realizan innecesariamente cambios de herramientas, cuestión que retrasa el proceso de ensamble.

- Se desensamblan piezas innecesarias durante la extracción de partes defectuosas.
- Se viola involuntariamente la precedencia de piezas durante los procesos de ensamble y desensamble.
- Se omiten secuencias de ensamble óptimas o cercanas a las óptimas.
- El proceso de planificación es altamente dependiente del conocimiento de un operario.

Por todo lo anteriormente expuesto se deriva el siguiente **problema científico**:

La cantidad de reorientaciones y el número de cambios de herramientas afectan el número total de operaciones durante el proceso de ensamble.

En tal sentido, se determina como **objeto de estudio**:

Los métodos de optimización aplicados al problema de planificación de secuencias de ensamble.

Para solucionar el problema planteado se define como **objetivo general**:

Desarrollar un módulo que minimice la cantidad de reorientaciones y los cambios de herramientas para disminuir el número total de operaciones durante el proceso de ensamble.

Como **hipótesis** de la investigación se plantea:

La utilización de un módulo basado en colonia de hormigas que minimice la cantidad de reorientaciones y el número de cambios de herramientas permitirá disminuir el número total de operaciones durante el proceso de ensamble.

Variables independientes:

- La cantidad de reorientaciones.
- El número de cambios de herramientas.

Variable dependiente:

- El número total de operaciones durante el proceso de ensamble.

Se define como **campo de acción**:

Los métodos de optimización basados en colonia de hormigas aplicados al problema de planificación de secuencias de ensamble.

Para lograr el objetivo propuesto, se realizaron las siguientes **tareas de la investigación**:

- Elaboración del marco teórico de la investigación a partir del estado del arte existente actualmente.

- Caracterización del problema de planificación de secuencias de ensamble para determinar qué elementos intervienen en él y cómo representarlo.
- Caracterización de las metaheurísticas aplicadas a los problemas de planificación de secuencias de ensamble para determinar aspectos relevantes que puedan ser utilizados en la solución.
- Análisis de los algoritmos de optimización basados en colonia de hormigas aplicados a los problemas de planificación de secuencias de ensamble para determinar cuál es el apropiado para la solución.
- Implementación del algoritmo elegido de optimización basado en colonia de hormigas, aplicado a la planificación de secuencias de ensamble, para resolver el problema en cuestión.
- Comprobación del desempeño del algoritmo propuesto mediante casos de estudios que verifiquen la efectividad de este.

Para lograr el cumplimiento del objetivo propuesto se utilizaron los siguientes **métodos científicos:**

Métodos generales: el histórico-lógico y el dialéctico, para el estudio crítico del estado del arte sobre los métodos de optimización que se han aplicado para resolver los problemas de planificación de secuencias de ensamble. El sistémico para la elaboración de un módulo que cumpla con los requisitos del objeto de estudio.

Métodos lógicos: el analítico-sintético, para estudiar cómo afecta la cantidad de reorientaciones y los cambios de herramientas en la planificación de secuencias de ensambles.

Métodos empíricos: el experimental, para comprobar y fundamentar con casos de estudio que el uso de un módulo que minimice la cantidad de reorientaciones y el número de cambios de herramientas permitirá disminuir el número de operaciones durante la planificación de secuencias de ensamble en la ICVT.

Alcance

En una primera etapa de desarrollo la GID precisa priorizar una solución capaz de minimizar el número de reorientaciones y los cambios de herramientas. Además, debe cumplir con las restricciones de precedencia y las restricciones geométricas. Para el presente trabajo se consideran seis direcciones de ensamble a lo largo de los tres ejes principales $(\pm x, \pm y, \pm z)$ y una única herramienta para desensamblar cada componente.

El documento está estructurado en tres capítulos. En el Capítulo §1 se exponen las características de los problemas ASP y los tipos de restricciones que intervienen en él. Se presentan las principales metaheurísticas existentes en la literatura para tratar los problemas ASP y se justifica la elección de la optimización basada en colonia de hormigas. Finalmente se selecciona el algoritmo Sistema de Hormigas $\mathcal{MAX}$-$\mathcal{MIN}$ ($\mathcal{MM}$AS) para el desarrollo de la solución. Se comenta además sobre los sistemas CAD comerciales de apoyo al problema ASP. En el Capítulo §2 se realiza la propuesta de solución y en el Capítulo §3 se exponen los resultados y la discusión del tema.

CAPÍTULO 1

Fundamentación teórica

1.1. Introducción

En este capítulo se realiza un estudio sobre el problema ASP y sus restricciones. Además, se abordan las características fundamentales de las técnicas metaheurísticas aplicadas a los problemas ASP, haciendo mayor énfasis en la Optimización basada en Colonia de Hormigas (ACO). De igual forma se comenta sobre los sistemas CAD comerciales de apoyo a la ASP.

1.2. Planificación de Secuencias de Ensamble

El ensamble se considera uno de los procesos más importantes en la manufactura. Este consume hasta un 50 % del tiempo total de producción y se responsabiliza por más de un 20 % del total de los costos (Pan, 2005). Un ensamble es un objeto compuesto de partes individuales en determinadas ubicaciones relativas, de tal manera que no se superponen, y donde cada parte está en contacto con un subconjunto del ensamble. La secuencia de ensamble calcula un orden de operaciones libres de colisiones para juntar las partes. Esta tarea se realiza dada una descripción geométrica de las posiciones de las partes en el ensamble del producto final (Jiménez, 2013).

La optimización del ensamble ayuda a aumentar la velocidad de los procesos de ensamble y reduce los costos. Así, la ASP es un componente importante en la planificación de ensambles. Esta se refiere a una tarea para la cual los planificadores, sobre la base de sus heurísticas particulares, organizan una secuencia de ensamble específica de acuerdo con la descripción de diseño de un ensamble (Rashid, Hutabarat & Tiwari, 2012).

El resultado de la ASP es un aspecto crucial, pues determina varias características del ensamble como los cambios de herramientas y las direcciones posibles de movimiento. La secuencia de ensamble también influye en la productividad completamente, pues determina qué tan rápido y exacto es el ensamble de un producto (Rashid et al., 2012).

Por otra parte, los productos mecánicos usualmente se componen de múltiples partes o subensambles. Por tanto, existe siempre más de una forma para ensamblar un producto. Las partes, al ser ensambladas, pueden ser muy diferentes en términos de la geometría del componente, la precedencia, la accesibilidad y otros tipos de limitaciones. Así, el problema ASP se considera como un problema de optimización combinatoria donde el espacio de búsqueda se incrementa cuando aumenta el número de componentes. Es un problema a gran escala por los posibles movimientos a considerar para cada componente. Además, el problema es altamente restringido ya que está sujeto a restricciones

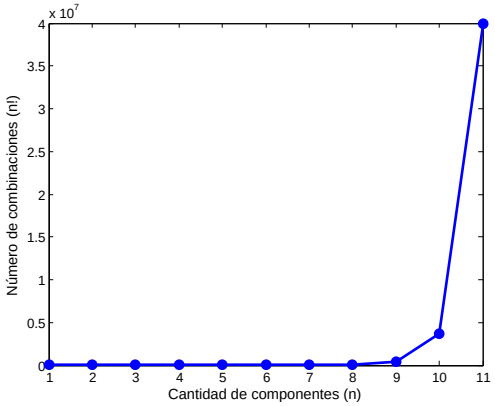


Figura 1.1: Relación entre el número de combinaciones y el número de componentes de un ensamble

geométricas, de precedencia, de cambios de herramientas, de número de reorientaciones, de cambios de puestos de trabajo, de accesibilidad, entre otras. Aun cuando las restricciones limitan el número de secuencias, existe un número considerable de posibles combinaciones de estas (Homem de Mello & Sanderson, 1991). El número de secuencias posibles es una función factorial del número de componentes, ver Figura 1.1. El problema ASP se encuentra dentro de los problemas NP-*hard*, para los cuales hasta el momento, no se ha derivado una solución exacta en tiempo polinomial.

Solucionar el problema ASP conlleva a analizar una gran cantidad de combinaciones de alternativas, para verificar y seleccionar la mejor secuencia de ensamble. Por consiguiente, se puede alcanzar un tiempo de cálculo inaceptable, incluso en el caso de utilizar equipos de cómputo con altas prestaciones.

1.3. Restricciones en la Planificación de Secuencias de Ensamble

Existen dos tipos de restricciones en el proceso de ensamble. Estas son las restricciones absolutas y las restricciones de optimización (Marian, Luong & Abhary, 2003). Las absolutas son aquellas que si son violadas conducen a secuencias de ensamble infactibles, mientras que las de optimización son las restricciones que cuando son violadas conducen a disminuir la calidad de la secuencia de ensamble.

En los problemas ASP, las restricciones absolutas que se consideran usualmente son las restricciones de precedencia y las restricciones geométricas. Las primeras muestran las relaciones de los componentes predecesores y sucesores en el proceso de ensamble y pueden ser representadas mediante un diagrama de precedencia o en forma matricial (Rashid et al., 2012). La matriz (1.3.1) muestra la precedencia entre los componentes de la Figura 1.2. En esta matriz, cuando la parte i debe ser ensamblada después de la parte j, $P(i,j) = 1$. En caso contrario, $P(i,j) = 0$.

$$\begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \end{array} \begin{array}{c} \begin{array}{cccc} 1 & 2 & 3 & 4 \end{array} \\ \begin{bmatrix} 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 \end{bmatrix} \end{array} \tag{1.3.1}$$

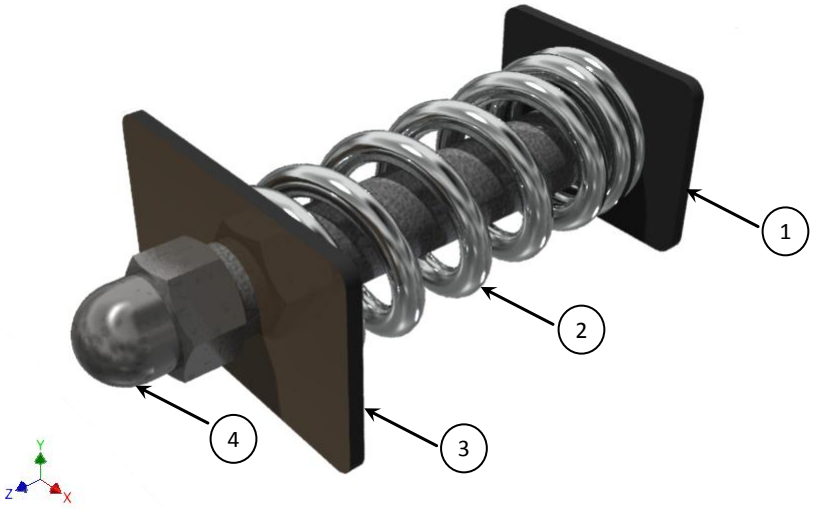


Figura 1.2: Ejemplo ilustrativo

Las restricciones geométricas en ASP determinan el ensamble de los componentes. Para realizar el acoplamiento de dos partes, debe existir al menos una trayectoria libre de colisiones que permita que todos los componentes estén en contacto. Todas las secuencias de ensamble válidas deben cumplir las restricciones geométricas para una estructura determinada. Las restricciones geométricas al igual que las de precedencia pueden ser representadas mediante una matriz (Rashid et al., 2012). En dicha matriz para cada par de componentes (P_i, P_j), se registran las direcciones en las que P_i se puede ensamblar sin colisionar con P_j (Chen & Liu, 2001).

Por otro lado, las restricciones que son clasificadas como restricciones de optimización están relacionadas con problemas de optimización multiobjetivo. En esta categoría las restricciones que más se destacan son: las restricciones de cambios de herramientas, el número de reorientaciones y los cambios de puestos de trabajo (Rashid et al., 2012).

En este trabajo se hace una fusión para representar las restricciones de precedencia y geométricas en una misma matriz. Esta estrategia fue utilizada antes por (Wang, Liu & Zhong, 2005). En relación a las restricciones de optimización se abordan el número de reorientaciones y los cambios de herramientas.

1.4. Métodos de optimización para el problema ASP

La optimización de un problema combinatorio y de generación de secuencias por métodos clásicos de optimización es altamente compleja debido al número de variables y restricciones a tener en cuenta. Luego de varios intentos de resolución sin éxito del problema anterior linealizado y lo reportado por otros autores con problemas similares *"no se continuaron haciendo más intentos para la solución de este problema por métodos clásicos de optimización"* (apud Tomás García, 2009, p. 16).

Homem de Mello y Sanderson (1991) desarrollaron los grafos AND/OR para lograr una representación más completa de una secuencia de ensamble. Sin embargo, esta técnica presenta una desventaja en su habilidad para representar completamente las secuencias, dado que el grafo resulta muy grande y difícil de manipular con el incremento del número de componentes. Ko y Lee (1987) desarrollaron un método de ordenamiento de partes el cual explora las condiciones de empalme de

caras[1] e interacciones entre los diferentes componentes. El método asegura que no existan colisiones entre las partes durante el proceso de ensamble. Ben-Arieh y Kramer (1994) desarrollaron un modelo para generar secuencias de ensamble a partir de considerar relaciones de precedencia y de diferentes combinaciones de subensambles. Huang y Lee (1991), al igual que Dini y Santochi (1992) desarrollaron algunos modelos matemáticos para representar la información requerida en la generación de secuencias de ensamble.

En la optimización de ASP las técnicas de *soft computing* son más aceptadas ya que tienen la habilidad para tratar con problemas complejos, de grandes dimensiones y de numerosas restricciones. Lendak, Erdeljan, Čapko y Vukmirovic (2010) definieron los métodos de *soft computing* como una estrategia que se caracteriza por el uso de soluciones inexactas en problemas NP-*hard*.

Varios métodos de *soft computing* han sido utilizados en la optimización del problema ASP, tales como: Recocido Simulado, Redes Petri, Algoritmos Meméticos, Algoritmos Inmunes, Redes Neuronales y Búsqueda Tabú. También se ha utilizado la combinación de algunos de estos tales como: Algoritmos Genéticos (AG) con el método de Recocido Simulado (Lin, Che, Chiang, Che & Chiang, 2009), Optimización por Enjambre de Partículas (PSO) y Recocido Simulado (Shan, Zhou & Sun, 2009), AG y Búsqueda Tabú y ACO con Recocido Simulado (Hui, Yuan & Kai-fu, 2009). Vale destacar que también se desarrolló una estrategia teórica utilizando una red neuronal con *back-propagation* y una base de datos de conocimientos para asistir a los ingenieros de ensamble en la predicción temprana de secuencias cercanas a las óptimas (Chen, Hsu, Hsieh & Tai, 2010; Hsu, Tai, Wang & Chen, 2011). En los últimos diez años los métodos más usados para resolver este tipo de problema han sido: AG, ACO y PSO; véase (Rashid et al., 2012) y las referencias dentro de este.

1.4.1. Algoritmos Genéticos

Los AG fueron introducidos por John Henry Holland en 1975. Estos son programas evolutivos y adaptativos de búsqueda. Además, son algoritmos de optimización basados en la mecánica de la genética y la selección natural. Los AG son una técnica de búsqueda y optimización que opera en el principio Darwiniano de la supervivencia del más apto. De esta forma los individuos débiles mueren antes de reproducirse y los más fuertes pueden sobrevivir, tener muchos descendientes y descendientes de especie, que a menudo heredan cualidades que son, en muchos casos, superiores a las cualidades de sus padres. Los AG evolucionan con una población de cadenas creadas al azar (Kumar, Murthy & Chandrashekara, 2012). Estos algoritmos comienzan con un juego de cromosomas llamado población inicial y pasan a través de tres operaciones, la reproducción o selección, cruce y mutación, para obtener una solución óptima heurísticamente (Ahmed, 2013).

Para utilizar los AG es necesario encontrar una posible estructura para representar las soluciones. Pensando este asunto como el problema de buscar en un espacio de estados, una instancia de esta estructura representa un punto o un estado en el espacio de búsqueda de todas las posibles soluciones. Así, una estructura de datos en el AG consistirá en uno o más cromosomas (frecuentemente uno), el cual se representa comúnmente como una cadena de *bits*. Cada cromosoma (cadena) es una concatenación de un número de subcomponentes llamados genes. La posición de un gen en el cromosoma se conoce como locus y sus valores como alelos. En la representación como cadena de *bits*, un gen es un *bit* o una cadena de *bits*, un locus es su posición en la cadena y un alelo es su valor (0 o 1 si es un *bit*) (Gálvez, 1998).

Al optimizar una estructura usando un AG se necesita una medida de la calidad de cada estruc-

[1] en inglés, *mate conditions*

tura en el espacio de búsqueda. La función de adaptabilidad es la encargada de esta tarea. En una maximización de funciones, la función objetivo frecuentemente actúa como la función de adaptabilidad. Los AG realizan una maximización por defecto, para los problemas de minimización los valores de la función objetivo pueden ser negados y trasladados con vistas a tomar valores positivos para producir así la adaptabilidad (Gálvez, 1998).

El modo de trabajo de un AG canónico puede resumirse en el esquema mostrado en la Figura 1.3.

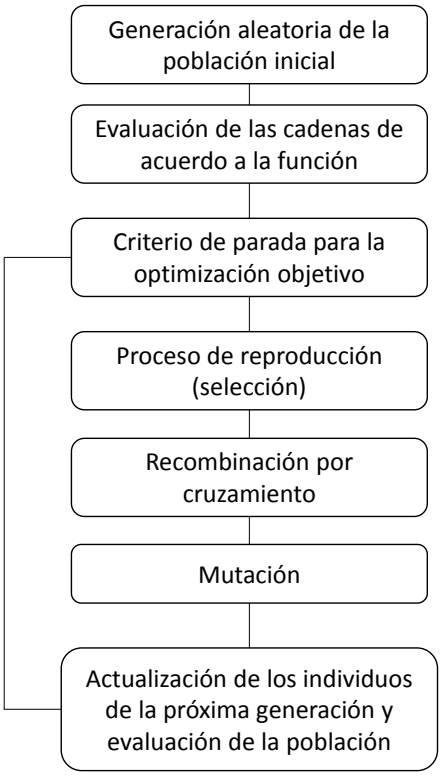


Figura 1.3: Diagrama funcional de un algoritmo genético (Gálvez, 1998)

- El AG canónico genera aleatoriamente una población de n estructuras (cadenas, cromosomas o individuos).
- Se realiza una evaluación de las cadenas de acuerdo a una función de aptitud.
- Sobre la población actúan los operadores transformando la población. Una vez completada la acción de los tres operadores se dice que ha transcurrido un ciclo generacional.
- Luego se repite el paso anterior mientras no se garantice el criterio de parada del AG.

Hay tres operadores principales en los AG: selección, cruce y mutación. La selección es un operador en el cual las cadenas del individuo son copiadas de acuerdo a sus valores objetivos. Este operador trae como resultado individuos altamente adaptados y otros individuos menos débiles en la piscina de apareamiento intermedia. El operador de selección es seguido por el operador de cruce que se realiza en dos pasos: en primer lugar, los miembros de la piscina de apareamiento se acoplan al azar;

en segundo lugar, cada par de individuos acoplados experimentan el cruzamiento con respecto a uno o más puntos de cruce. Estas porciones de cadenas son intercambiadas entre pares. El mecanismo de selección y cruzamiento, si bien parece ser simple, su acción combinada proporciona a los AG gran parte de su poder de búsqueda. La mutación desempeña un papel secundario en los AG y trata de asegurar las soluciones potenciales (individuos). Los operadores de selección y cruce proporcionan la solución de convergencia, evitando óptimos locales. El resultado de la nueva población es entonces evaluado y probado para favorecer su terminación. Los criterios de terminación se diseñan basándose en el tiempo de respuesta disponible dentro del cual se obtiene la solución o en un esperado nivel de rendimiento. Si el criterio de terminación no se cumple, la población se evalúa nuevamente por la función de aptitud. Este procedimiento continúa y se repite hasta que el criterio de terminación se cumple (Kumar et al., 2012).

En los AG el número de soluciones a considerar es reducido comparado con los algoritmos exactos. Además, los AG pueden tratar problemas complejos y con múltiples restricciones. Sin embargo, el AG canónico es poco adecuado para ser usado directamente en la resolución y optimización de problemas ASP. La primera razón es que la cadena binaria original de cromosomas es menos adecuada para problemas complejos combinatorios como es el ASP (Rashid et al., 2012). La segunda razón es respecto a la factibilidad de los cromosomas en la manipulación de las restricciones de precedencia del ensamble. El AG canónico tiende a generar descendencias infactibles que violan las restricciones de precedencia debido a los cruzamientos y operadores de mutación (Rashid et al., 2012).

Aun cuando los AG han sido implementados en ASP, los investigadores resaltan algunos aspectos respecto a este algoritmo. El principal aspecto es que el AG canónico es susceptible a convergencias prematura (Shan, Li, Gong & Lou, 2006; Shu-xia & Hong-bo, 2008). La llamada convergencia prematura se manifiesta cuando todos los individuos de la población tienen un valor de adaptación o ajuste muy cercano y la mejoría de una generación a otra es muy poca. Si al principio de una corrida, se toma por el operador de selección una proporción significativa de la población en una generación, puede conducir a una convergencia prematura. Al final de una corrida, el ajuste promedio de la población puede estar cercano al mejor ajuste de la población, lo que puede conducir a un movimiento aleatorio entre los mediocres (Gálvez, 1998). Los AG ordenan rápidamente áreas interesantes de un espacio, pero son un método débil, sin la garantía de procedimientos más convergentes. En este caso si para el problema que se está resolviendo se conocen métodos locales pero convergentes, la solución es aplicar un esquema híbrido. De esta forma cuando el AG encierra las mejores regiones, se aplican los métodos locales (Gálvez, 1998). En adición, vale destacar los altos costos computacionales de este algoritmo (Moon, Logendran & Lee, 2009).

Las investigaciones de ASP basadas en AG están principalmente enfocadas en la codificación de los cromosomas en la secuencia de ensamble, en la forma de las operaciones genéticas, en la evaluación multiobjetivo y en la mejora de la capacidad de búsqueda local. En estas investigaciones la matriz de precedencia se utiliza frecuentemente para satisfacer las restricciones de precedencia, pero esta no puede describir cuáles direcciones de precedencia existen. Los AG también tienden a generar descendencias infactibles que violan las restricciones de precedencia geométricas ya que los operadores de cruce y mutación y las cadenas binarias en los cromosomas son menos apropiadas para problemas combinatorios de alta complejidad (Yu & Wang, 2013).

De manera general para resolver las deficiencias del AG canónico, se han propuesto diferentes tipos de estrategias, de penalidad y de reparación (Rashid et al., 2012). En Yu y Yin (2010) los autores propusieron un AG adaptativo para resolver los asuntos de convergencia prematura y los altos costos computacionales.

Por otra parte para mejorar los operadores del AG canónico, se han combinado los AG con otros algoritmos de *soft computing*. En general, la combinación de AG con Recocido Simulado (Shan et al., 2006; Shu-xia & Hong-bo, 2008), Búsqueda Tabú (Li, Khoo & Tor, 2003) y Optimización basada en Colonia de Hormigas (Hui et al., 2009) han tenido mejores rendimientos comparados con el AG original (Rashid et al., 2012). En este sentido vale destacar los trabajos de De Lit, Latinne, Rekiek y Delchambre (2001), quienes desarrollaron una estrategia basada en AG para la generación de un árbol de ensamble de un producto mecánico. Chang, Tseng y Meng (2009) utilizaron posteriormente AG para generar los planes de secuencia de ensamble a través de sistemas artificiales inmunes. Zhou, Zheng, Yan y Wang (2011), propusieron un sistema de planificación de secuencia de ensamble utilizando quimiotaxis bacterial conjuntamente con AG. La ventaja de este sistema radica en su habilidad para mantener la diversidad de las poblaciones durante el proceso de evolución del AG.

1.4.2. Optimización por Enjambre de Partículas

La PSO fue creada por Kennedy y Eberhart (1995). Este algoritmo utiliza un propósito general, es iterativo y de búsqueda heurística. Se considera una población de individuos para probar las regiones prometedoras del espacio de búsqueda de una manera efectiva. En este contexto, la población de las soluciones se llama enjambre y los individuos se llaman partículas. Cada partícula se mueve dentro del espacio de búsqueda y retiene en su memoria la mejor posición y la mejor posición general que se ha encontrado. La velocidad de cada partícula se ajusta durante cada iteración hacia la mejor posición personal, así como la mejor posición de conjunto, imitando así la inteligencia de enjambre (Meier, Brandt, Pisabarro, Baldauf & Sippl, 2008).

En un problema de optimización de minimización, "mejor" significa simplemente la posición con el menor valor objetivo. Los miembros de un enjambre comunican las mejores posiciones entre sí y ajustan su posición y velocidad personal basados en estas mejores posiciones. La mejor posición de una partícula individual se llama "mejor local" y la mejor posición obtenida por todas las partículas se llama "mejor global". Por lo tanto, es la posición mejor global la que todas las partículas tienden a seguir (Ch & Mathur, 2010).

Una población de partículas se inicializa con posiciones aleatorias W_i y velocidad V_i de d dimensiones. El valor de la función de aptitud F, se evalúa por el uso de sus posiciones actuales como una entrada para cada partícula. El valor de la función de aptitud se compara en cada posición de la partícula actual con la posición de la posible mejor aptitud almacenada hasta ese momento. Si la aptitud actual calculada es mejor que una de las almacenadas hasta ese momento, entonces se almacena como $P_{best} = P_i^1, P_i^2$. La diferencia ponderada entre P_{best} y W_i se adiciona a la velocidad de la partícula. Del mismo modo, el mejor valor actual de aptitud se compara con los valores de aptitudes actuales de todos los demás valores. Los mejores de ellos se almacenan como $P_{global} = P_g^1, P_g^2$. La diferencia ponderada estocásticamente entre W_i y P_{global} también se almacena a la velocidad de la partícula. Si γ_1 y γ_2 representan números aleatorios entre 0 y 1 y ψ_1 y ψ_2 representan constantes de aceleración, entonces se sigue la ecuación (1.4.1):

$$V_i^{k+1}(m,n) = V_i^k(m,n) + \psi_1\gamma_1(P_{best}^k(m,n) - W_i^k(m,n)) + \psi_2\gamma_2(P_{global}^k(m,n) - W_i^k(m,n)). \tag{1.4.1}$$

La posición se actualiza según la ecuación (1.4.2):

$$W_i^{j+1} = W_i^j + V_i^j. \tag{1.4.2}$$

donde $k+1$ y $j+1$ denotan el nivel de iteración respectivamente (Ch & Mathur, 2010).

La PSO es muy similar a AG en la cual el sistema se inicializa con una población de soluciones aleatorias. Sin embargo, distinto a AG, la PSO no tiene evolución de operadores tales como el cruzamiento y la mutación. Las soluciones potenciales, llamadas partículas, vuelan a través del espacio del problema siguiendo las partículas óptimas actuales (Sivanandam & Deepa, 2008).

La PSO es relativamente un algoritmo nuevo comparado con AG y ACO. Además, PSO es un algoritmo sencillo ya que solamente usa una simple fórmula de velocidad para desarrollarse (Lv, Lu & Zha, 2010). Por lo tanto, el algoritmo PSO es fácil de implementar y requiere de menos recursos computacionales comparado con AG. Sin embargo, al igual que AG el algoritmo PSO original no es apropiado para ser aplicado directamente a los problemas ASP. En adición al asunto de las restricciones de precedencia, el algoritmo PSO original está diseñado para problemas continuos, donde la solución es un espacio de valor real mientras que las soluciones de ASP residen en un espacio discreto entero (Lv & Lu, 2010; Wang & Liu, 2010). Otro asunto importante del algoritmo PSO original es que es fácilmente atrapado en óptimos locales (Wang & Liu, 2010).

1.4.3. Optimización basada en Colonia de Hormigas

La ACO está inspirada en el comportamiento de las hormigas reales, el cual se distingue por la búsqueda de alimentos, trazando el camino más corto entre el hormiguero y la fuente de comida. El eje de esa búsqueda está basado en la modificación del ambiente local depositando una sustancia química llamada feromona. Esta sustancia sirve como rastro y orienta el recorrido sin necesidad del sentido de la vista. Es por ello que les permite comunicarse de manera indirecta con otros miembros de la colonia de hormigas.

Estos insectos tienden a seguir la mayor concentración de feromona. En principio, ellas siguen rutas aleatorias. Por este motivo, si no existe inicialmente ningún rastro de feromona en el medio, cuando una hormiga llega a una bifurcación, elige al azar una de las vías posibles. La feromona se evapora a medida que pasa el tiempo, así que el rastro de un camino más corto tendrá más concentración de feromona que otro más largo. De esta manera, la probabilidad de que una hormiga escoja el camino más corto aumenta progresivamente. Al final, el recorrido de la colonia converge al más corto de todos los caminos posibles (Wang et al., 2005).

Finalmente, el proceso se caracteriza por una retroalimentación positiva, donde la probabilidad con la que una hormiga escoge un camino aumenta con el número de hormigas que previamente hayan elegido el mismo camino.

Para aplicar ACO, el problema de optimización es transformado en un problema para descubrir la mejor ruta en un grafo ponderado. Las hormigas artificiales construyen soluciones aceleradamente por movimientos en el grafo. El proceso de construcción de la solución es probabilístico (Rashid et al., 2012).

Para guiar los movimientos en el grafo las hormigas se basan en dos tipos de información que se encuentran en cada arista o tramo del grafo.

- Información heurística, mide la preferencia heurística de moverse desde un nodo a otro. Las hormigas no modifican esta información durante la ejecución del algoritmo.

- Información de los rastros de feromona artificiales, mide la "deseabilidad aprendida" del movimiento de un nodo a otro. Imita a la feromona real que depositan las hormigas naturales. Esta información se modifica durante la ejecución del algoritmo dependiendo de las soluciones encontradas por las hormigas (Dorzán, Gagliardi, Leguizamón, Taranilla & Hernández-Penalver, 2009).

Uno de los inconvenientes que fueron detectados por los investigadores en el algoritmo ACO original es con relación al sistema de retroalimentación positiva que sólo acumula buenas soluciones. De esta forma se dejan de evaluar soluciones que pueden estar a pocas iteraciones del óptimo global. En este algoritmo la mejor solución será la que deposita mayor cantidad de feromona. Sin embargo, cuando se genera una solución mala el rastro de feromona para todas las rutas se fija para ser evaporado, lo que causa una convergencia prematura (Rashid et al., 2012).

Las tres técnicas antes mencionadas (ACO, AG y PSO) no pueden ser aplicadas directamente en su forma básica para resolver los problemas ASP (Rashid et al., 2012). Por esta razón se han creado implementaciones específicas para cada una de ellas dando origen a nuevas estrategias de estos algoritmos.

A continuación se presentan algunos de los algoritmo de ACO más reconocidos en la literatura.

Sistema de Hormigas (AS)

El primer algoritmo de ACO, fue propuesto por Colorni, Dorigo y Maniezzo. Este algoritmo recibe el nombre de AS y fue aplicado por primera vez al Problema del Agente Viajero (TSP).

Inicialmente, se presentaron 3 variantes distintas: AS-densidad, AS-cantidad y AS-ciclo. Estos algoritmos se diferencian en la manera en que se actualizan los rastros de feromona. En los dos primeros, las hormigas depositan feromona mientras construyen sus soluciones, con la diferencia de que la cantidad de feromona depositada en el AS-densidad es constante, mientras que la depositada en AS-cantidad depende directamente de la información heurística. Por último, en AS-ciclo, la deposición de feromona se lleva a cabo una vez que la solución está completa. Esta última variante es la que obtiene mejores resultados y es por tanto la que se conoce como AS en la literatura (Alonso, Cordón, Fernández & Herrera, 2004).

En el algoritmo AS, se utiliza la regla probabilística de transición entre estados (1.4.3) para la selección de los nodos que conforman la ruta completa (Wu, Hong & Lee, 2012).

$$p_{ij}^k(t) = \frac{[\tau_{ij}(t)]^\alpha [\eta_{ij}(t)]^\beta}{\sum_{l \in N_i^k} [\tau_{il}(t)]^\alpha [\eta_{il}(t)]^\beta}, \quad \text{con} \quad j \in N_i^k, \tag{1.4.3}$$

donde $p_{ij}^k(t)$ es la probabilidad con la que, en una iteración t, la hormiga k, situada actualmente en el nodo i, elige al nodo j como próxima parada; $\tau_{ij}(t)$ es la cantidad de feromona acumulada sobre el arco (i, j), en la iteración t; η_{ij} es la información heurística; N_i^k es el conjunto de nodos no visitados por la hormiga k.

Esta regla probabilística presenta los parámetros configurables α y β que determinan la importancia relativa del rastro de feromona y de la información heurística respectivamente. Si $\alpha = 0$, aquellos nodos con una preferencia heurística tienen una mayor probabilidad de ser escogidos. Sin embargo, si $\beta = 0$, sólo se tienen en cuenta los rastros de feromona para guiar el proceso constructivo, lo que puede causar un rápido estancamiento. En la situación anterior los rastros de feromona asociados a una solución son ligeramente superiores que el resto, lo que provoca que las hormigas

siempre construyan las mismas soluciones, normalmente óptimos locales. Por tanto, es preciso establecer una adecuada proporción entre la información heurística y la información de los rastros de feromona (Alonso et al., 2004).

Cuando todas las hormigas han construido una solución debe actualizarse la feromona en cada arco. Siendo ρ el coeficiente de evaporación de la feromona, se sigue la ecuación (1.4.4) para la actualización de dicha sustancia:

$$\tau_{ij}(t+1) = (1-\rho)\tau_{ij}(t) + \Delta\tau_{ij}^{best}, \quad \Delta\tau_{ij}^{best} = \begin{cases} \frac{1}{L^{best}} & : \text{si el arco } (i,j) \in T^{best} \\ 0 & : \text{en caso contrario} \end{cases} \tag{1.4.4}$$

donde, T^{best} es la mejor solución encontrada en la iteración; L^{best} es la longitud de la solución T^{best}. Finalmente se itera el mismo proceso.

Al comparar el rendimiento relativo a AS con respecto a otras metaheurísticas este tiende a decrementarse dramáticamente cuando se incrementa el tamaño de las instancias del problema. Por tal motivo, una cantidad importante de investigadores en ACO se han enfocado en cómo mejorar el AS (Dorigo & Stützle, 2004).

Sistema de Hormigas Elitista (EAS)

Con el fin de mejorar la eficiencia, el algoritmo AS ha sufrido frecuentes modificaciones. Una primera mejora en el AS inicial es la llamada estrategia elitista. Así, el AS se convirtió en el EAS. En el algortimo EAS cada hormiga que encuentra una mejor solución tiene la oportunidad de depositar más feromona. Así, una vez que las hormigas han depositado feromona en las conexiones asociadas a sus respectivas soluciones, se realiza una deposición adicional de feromona en las aristas que pertenecen a la mejor solución global de la iteración. La cantidad de feromona depositada, que depende de la calidad de la mejor solución global, se incrementa en un factor e, que se corresponde con el número de hormigas elitistas consideradas (Dorigo & Stützle, 2004).

La actualización de la feromona se realiza mediante la ecuación (1.4.5):

$$\tau_{ij}(t+1) = (1-\rho)\tau_{ij}(t) + e\Delta\tau_{ij}^{best}, \quad \Delta\tau_{ij}^{best} = \begin{cases} \frac{1}{L^{best}} & : \text{si el arco } (i,j) \in T^{best} \\ 0 & : \text{en caso contrario} \end{cases} \tag{1.4.5}$$

donde, T^{best} es la mejor solución encontrada en la iteración; L^{best} es la longitud de la solución T^{best}.

Los resultados computacionales sugieren que el uso de la estrategia elitista con un valor adecuado para el parámetro e permite, con respecto a AS, encontrar mejores excursiones y encontrarlas en un menor número de iteraciones.

Sistema de Hormigas basado en el rango (ASrank)

Otra mejora con respecto al AS es la versión ASrank, propuesto por Bullnheimer. En el ASrank cada hormiga deposita una cantidad de feromona que decrece con su rango. Además, como en EAS, la mejor hormiga en la iteración deposita la mayor cantidad de feromona.

Antes de actualizar los rastros de feromona, las hormigas son ordenadas por la longitud de sus trayectorias de manera creciente. Luego, la cantidad de feromona que depositan las hormigas se pondera de acuerdo al rango r de cada una.

En cada iteración solamente las $(w-1)$ hormigas de mejor rango y la hormiga que produjo la mejor trayectoria son las que pueden depositar feromona.

La mejor trayectoria hasta el momento da la más fuerte retroalimentación con peso w. La mejor hormiga r-ésima de la iteración actual contribuye a la actualización de feromona con el valor $\frac{1}{C^r}$ multiplicado por un peso dado por el $max(0, w-r)$. Así, la regla de actualización de la feromona en ASrank es (1.4.6):

$$\tau_{ij} = \tau_{ij} + \sum_{r=1}^{w-1}(w-r)\Delta\tau_{ij}^{r} + w\Delta\tau_{ij}^{best}, \tag{1.4.6}$$

donde $\Delta\tau_{ij}^{r} = \frac{1}{C^r}$ y $\Delta\tau_{ij}^{best} = \frac{1}{L^{best}}$; L^{best} es la longitud de la mejor solución obtenida en la iteración; C^r es la longitud de la solución T^r construida por la hormiga r-ésima; w es un cierto número de hormigas, las cuales se han escogido como mejores para realizar el depósito de feromona.

Todas las soluciones se clasifican de acuerdo a su longitud. La cantidad de feromona depositada se pondera para cada solución, de tal manera que las soluciones con los caminos más cortos depositan más feromonas que las soluciones con los caminos más largos.

Una evaluación experimental de los resultados realizada por Bullnheimer sugirieron que ASrank presentó rendimientos ligeramente mejores que EAS y significativamente mejores que AS (Dorigo & Stützle, 2004).

Sistema de Colonia de Hormigas (ACS)

Colorni, Dorigo y Maniezzo proponen el algoritmo ACS como una mejora al algoritmo AS. Así, ACS logra mejores rendimientos debido a la introducción de nuevos mecanismos basados en ideas no incluidas en el algoritmo original. El algoritmo ACS tiene tres cambios importantes en las reglas utilizadas en el algoritmo AS (Yousefikhoshbakht, Khorram & Hamedan, 2012):

1. Una regla novedosa de transición entre estados proporciona una forma directa de equilibrio entre exploración de nuevos caminos y explotación de la información acumulada. El nodo j que es el próximo nodo de i, entre el conjunto de nodos no visitados N_i^k, es seleccionado por la hormiga k en la ruta. De acuerdo con la regla de transición (1.4.7), la probabilidad con la que cada nodo se visita es:

$$p_{ij}^{k}(t) = \begin{cases} 1 \quad si \quad q \leq q_0 \quad y \quad j = j^* \\ 0 \quad si \quad q \leq q_0 \quad y \quad j \neq j^* \\ \frac{[\tau_{ij}(t)]^{\alpha}[\eta_{ij}(t)]^{\beta}}{\sum_{l \in N_i^k}[\tau_{il}(t)]^{\alpha}[\eta_{il}(t)]^{\beta}}, \quad \text{en caso contrario} \end{cases} \tag{1.4.7}$$

   donde j^* es el nodo j no visitado en N_i^k para el cual $[\tau_{il}(t)]^{\alpha}[\eta_{il}(t)]^{\beta}]$ es maximizado; q es una variable aleatoria entre $[0, 1]$; q_0 es un determinado parámetro arbitrario fijado con anterioridad, de manera que cuando q es menor que q_0 la hormiga emplea explotación para seleccionar al nodo j^* como el siguiente nodo en su trayectoria, mientras que si q excede a q_0, la hormiga utiliza exploración probabilística para seleccionar el siguiente nodo en su trayectoria.

2. Mientras las hormigas construyen una solución se aplica una regla de actualización local de

feromona mediante la ecuación (1.4.8):

$$\tau_{ij}(t+1) = (1-\rho)\tau_{ij}(t) + \rho\tau_0, \quad si \quad (i,j) \in T^{best}, \tag{1.4.8}$$

donde τ_0 es la cantidad inicial de feromona en cada arista y se calcula mediante la ecuación (1.4.9):

$$\tau_0 = (nL_i)^{-1}, \tag{1.4.9}$$

donde n es el número de nodos ; L_i es la longitud de la trayectoria inicial producida por la construcción heurística. Con esta operación, la feromona asociada a un arco disminuye cada vez que una hormiga lo visita. Los arcos ya visitados son menos prometedores según los recorren más hormigas en la iteración actual, lo que favorece la exploración de arcos no visitados. De esta forma, las hormigas tienden a no converger a soluciones parecidas en la iteración actual (Sorin & Costin, 2013).

3. La regla de actualización global es únicamente aplicada a los caminos pertenecientes a la mejor ruta y se realiza mediante la ecuación (1.4.10):

$$\tau_{ij}(t+1) = (1-\rho)\tau_{ij}(t) + \rho\frac{1}{L^{best}}, \quad si \quad (i,j) \in T^{best} \tag{1.4.10}$$

donde L^{best} es la longitud de T^{best}; T^{best} es la mejor solución encontrada.

$\mathcal{MM}$**AS**

Stüzle y Hoos propusieron el algoritmo $\mathcal{MM}$AS como una mejora del AS inicial para superar el estancamiento y la rápida velocidad de convergencia (Qin, Pan, Chen & Chen, 2006). Sus características principales son que únicamente la mejor hormiga actualiza el rastro de feromona y que el valor de feromona está acotado, tanto superior como inferiormente. Estas cotas son típicamente obtenidas de forma empírica y ajustadas según el problema específico (Socha, Knowles & Sampels, 2002).

Siendo α y β la importancia concedida a la feromona y a la heurística respectivamente, se sigue la regla probabilística (1.4.11):

$$p_{ij}^k(t) = \frac{[\tau_{ij}(t)]^\alpha [\eta_{ij}]^\beta}{\sum_{l \in N_i^k} [\tau_{il}(t)]^\alpha [\eta_{il}]^\beta}, \quad \text{con} j \in N_i^k, \tag{1.4.11}$$

donde $p_{ij}^k(t)$ es la probabilidad con la que, en una iteración t, la hormiga k, situada actualmente en el nodo i, elige al nodo j como próximo a visitar; $\tau_{ij}(t)$ es la cantidad de feromona acumulada sobre el arco (i,j), en la iteración t; η_{ij} es la información heurística; N_i^k es el conjunto de nodos no visitados por la hormiga k.

Cuando todas las hormigas han construido una solución debe actualizarse la feromona en cada arco. Sea ρ el coeficiente de evaporación de la feromona, se sigue la ecuación (1.4.12) para actualizar dicha sustancia.

$$\tau_{ij}(t+1) = (1-\rho)\tau_{ij}(t) + \Delta\tau_{ij}^{best}, \quad \Delta\tau_{ij}^{best} = \begin{cases} \frac{1}{L^{best}} & : \text{si el arco } (i,j) \in T^{best} \\ 0 & : \text{en caso contrario} \end{cases} \tag{1.4.12}$$

donde, T^{best} es la mejor solución encontrada en la iteración; L^{best} es la longitud de la solución T^{best}.

Vale destacar que para la actualización, la elección de T^{best} como la mejor solución encontrada en la iteración actual, constituye un rasgo distintivo del algoritmo $\mathcal{MM}$AS. Así se garantiza que las aristas que frecuentemente se hallen en esta mejor solución refuercen con un valor más alto su cantidad de feromona. Otros como ACS típicamente utilizan la mejor solución global encontrada. Cuando únicamente se utiliza esta última, la búsqueda puede concentrarse demasiado rápido alrededor de la solución y la exploración de posibles mejores soluciones queda limitada, con el consiguiente peligro de quedar atrapado con soluciones de pobre calidad. Aun así, una elección juiciosa entre la hormiga de la mejor iteración y la mejor hormiga global a la hora de actualizar la feromona puede controlar la forma en que el historial de búsqueda es aprovechado (Stützle & Hoos, 2000).

Independientemente de la elección, ya sea la mejor solución global encontrada o la mejor solución encontrada en la iteración actual, puede ocurrir un estancamiento en la búsqueda de soluciones. El estancamiento define aquella situación en la que todas las hormigas construyen una y otra vez la misma solución, cuestión que debe ser evitada sobre todo al inicio de una corrida. Esto pudiera suceder si para un nodo dado el rastro de feromona es significativamente más grande en una arista que en cualquier otra. Por tal motivo, el algoritmo $\mathcal{MM}$AS obliga a que el nivel de feromona permanezca acotado en un intervalo $[\tau_{min}, \tau_{max}]$.

Estos límites se imponen con el objetivo de influir en la probabilidad de escoger la próxima componente de una solución, cuestión que depende directamente del rastro de feromona y de la información heurística, ver ecuación (1.4.11). Si se limita la influencia del rastro de feromona, se puede evitar que las diferencias relativas entre rastros de feromona sean excesivamente grandes durante la corrida del algoritmo.

Para realizar el cálculo de los límites de la concentración de feromona, se siguen las expresiones (1.4.13) y (1.4.14):

$$\tau_{max} = \frac{1}{\rho}\frac{1}{L^{best}}. \tag{1.4.13}$$

$$\tau_{min} = \frac{\tau_{max}(1 - \sqrt[n]{p_{best}})}{(avg - 1)\sqrt[n]{p_{best}}}, \tag{1.4.14}$$

donde $avg = \frac{n}{2}$, n es la cantidad de vértices del grafo y p_{best} la probabilidad con la que se encuentra la mejor solución.

Precisamente en la ecuación (1.4.13), equivalente a la expresión propuesta en (Stützle & Hoos, 2000), constituye la cota superior del valor máximo de feromona que puede acumularse en una arista.

Proposición 1. *Para cualquier arista τ_{ij} se cumple que:*

$$\lim_{t\to\infty}(\tau_{ij}(t)) = \tau_{ij} \leq \frac{1}{\rho L^{best}}. \tag{1.4.15}$$

Demostración. De la ecuación (1.4.12) es posible notar que la máxima cantidad de feromona que puede depositarse en una arista durante una iteración es:

$$\Delta\tau_{ij}^{best} = \frac{1}{L^{best}} \tag{1.4.16}$$

Así para la iteración 1 se tiene (1.4.17).

$$\tau_{ij}(1) = (1-\rho)\tau_{ij}(0) + \Delta\tau_{ij}^{best}, \tag{1.4.17}$$

Para la iteración 2 se tiene (1.4.18).

$$\tau_{ij}(2) = (1-\rho)\tau_{ij}(1) + \Delta\tau_{ij}^{best}, \tag{1.4.18}$$

Escribiendo (1.4.18) en función de (1.4.17) queda (1.4.19).

$$\tau_{ij}(2) = (1-\rho)^2\tau_{ij}(0) + (1-\rho)\Delta\tau_{ij}^{best} + \Delta\tau_{ij}^{best}, \tag{1.4.19}$$

Para la iteración 3 se tiene (1.4.20).

$$\tau_{ij}(3) = (1-\rho)\tau_{ij}(2) + \Delta\tau_{ij}^{best}, \tag{1.4.20}$$

Escribiendo la ecuación (1.4.20) en función de (1.4.19) queda (1.4.21).

$$\tau_{ij}(3) = (1-\rho)^3\tau_{ij}(0) + (1-\rho)^2\Delta\tau_{ij}^{best} + (1-\rho)\Delta\tau_{ij}^{best} + \Delta\tau_{ij}^{best}, \tag{1.4.21}$$

A partir de este comportamiento es posible arribar a la expresión general (1.4.22) que describe la cantidad de feromona depositada en la arista (i, j) hasta la iteración t.

$$\tau_{ij}(t) = (1-\rho)^t\tau_{ij}(0) + \Delta\tau_{ij}^{best}\sum_{k=1}^{t}(1-\rho)^{t-k}, \tag{1.4.22}$$

El valor máximo que se puede acumular será:

$$\tau_{ij}^{max}(t) = \lim_{t\to\infty}\tau_{ij}(t), \tag{1.4.23}$$

Sustituyendo la ecuación (1.4.22) en la ecuación (1.4.23) se obtiene:

$$\tau_{ij}^{max}(t) = \lim_{t\to\infty}[(1-\rho)^t\tau_{ij}(0) + \Delta\tau_{ij}^{best}\sum_{k=1}^{t}(1-\rho)^{t-k}], \tag{1.4.24}$$

Para calcular el límite es necesario conocer el valor de la suma. Dicha suma puede escribirse convenientemente como $\sum_{i=0}^{t-1}(1-\rho)^i$ que desarrollada resulta:

$$\sum_{i=0}^{t-1}(1-\rho)^i = 1 + (1-\rho)^1 + (1-\rho)^2 + (1-\rho)^3 + \ldots (1-\rho)^{t-1} = s, \tag{1.4.25}$$

Multiplicando la suma s por el factor $(1-\rho)$ queda:

$$(1-\rho)s = (1-\rho)^1 + (1-\rho)^2 + (1-\rho)^3 + \ldots (1-\rho)^t, \tag{1.4.26}$$

Restando la ecuación (1.4.25) y la ecuación (1.4.26) se obtiene:

$$s - (1-\rho)s = 1 + (1-\rho)^t, \tag{1.4.27}$$

Resolviendo para s resulta:

$$s = \frac{1 + (1-\rho)^t}{\rho}, \tag{1.4.28}$$

Dado que $\rho \in (0, 1]$ el primer sumando en el límite resulta 0. Por igual razón la sumatoria en el segundo describe una serie geométrica absolutamente convergente de radio $(1-\rho)$ cuya suma es $\frac{1}{\rho}$. Por tanto el valor del límite será:

$$\tau_{ij}^{max}(t) = \lim_{t\to\infty} \tau_{ij}(t) = \frac{1}{\rho L^{best}}. \tag{1.4.29}$$

□

La determinación de la expresión (1.4.14) se realiza mediante suposiciones y observaciones empíricas que pueden ser estudiadas en (Stützle & Hoos, 2000).

Tras la actualización de la feromona se comienza una nueva iteración. El algoritmo $\mathcal{MM}$AS converge cuando para cada vértice del grafo, una arista de la solución tiene asociada la mayor cantidad de feromona τ_{max}, mientras que todas las demás aristas de las soluciones alternativas tienen el valor de feromona τ_{min}. Si el $\mathcal{MM}$AS converge, la solución que se construye eligiendo las aristas con mayor rastro de feromona corresponderá con la mejor solución encontrada por el algoritmo (Stützle & Hoos, 2000).

Este algoritmo posee una mayor explotación de las mejores soluciones y un mecanismo adicional para evitar el estancamiento de la búsqueda (Stützle & Hoos, 2000).

1.5. Sistemas CAD comerciales de apoyo al problema ASP

El estado del arte actual en la industria para la representación de modelos de ensamble está en la utilización de sistemas CAD comerciales disponibles para este propósito. Tales sistemas CAD pueden mostrar visualmente un ensamble, así como permitir una manipulación fácil de su diseño virtual, con propósitos de planificación y presentación. Entre los sistemas CAD disponibles en el mercado se destacan Creo Parametric[2] (PTC, 2013), CATIA (Systèmes, 2013a), SolidWorks (Systèmes, 2013b), NX (SPLMS, 2013a), Solid Edge (SPLMS, 2013b) e Inventor (Autodesk, 2013). Todos tienen los mismos principios básicos para el modelado de ensambles y presentan las mismas restricciones básicas e.g. Empalme o Alineación, Tangencia, Desplazamiento de ángulo y Desplazamiento de distancia. En estos sistemas también se puede crear un mecanismo funcional que mediante la alteración de las restricciones básicas de un ensamble permita comprobar su funcionamiento. Esto le permite al diseñador comprobar mediante un prototipo virtual el correcto funcionamiento del producto diseñado, corregir algún error y poner a prueba su diseño completo. A pesar de la vasta información relacionada con el ensamble presente en un paquete CAD comercial, es válido destacar que ninguno de ellos actualmente utiliza esa información para generar automáticamente secuencias

[2]anteriormente Pro/Engineer Wildfire

de ensamble (Ou & Xu, 2013). En este sentido es válido notar que los avances en esta área del conocimiento se recogen fundamentalmente en investigaciones teóricas, no siempre disponibles. Por otro lado, las empresas que tienen automatizado el proceso protegen su *know how* celosamente. Razones como estas demandan la creación de un módulo que asista de manera automática la ASP.

1.6. Consideraciones finales

Para la solución del problema ASP se elige la metaheurística ACO por su éxito en la solución de problemas discretos entre los que sobresale el TSP. En este sentido vale destacar que si bien la representación de un problema ASP es un grafo conexo, la solución del problema con algunas transformaciones puede encontrarse en un subgrafo completo de la representación original. Estas razones son las que han motivado a los investigadores a utilizar esta estrategia en los problemas ASP (Rashid et al., 2012). Los algoritmos de ACO que en la práctica presentan mejores resultados son ACS y $\mathcal{MM}$AS (Dorigo & Blum, 2005; Dorigo & Stützle, 2004; Dorigo, Birattari & Stutzle, 2006). De estos, el que mejores resultados ha presentado en el tratamiento de los problemas TSP es el $\mathcal{MM}$AS (Stützle & Hoos, 2000), por lo que se escoge este algoritmo para el desarrollo de la solución.

CAPÍTULO 2

Propuesta de solución

2.1. Introducción

En este capítulo se brindan detalles de la representación de los problemas ASP. De igual forma se comentan las secuencias factibles de la ASP, así como los pormenores de los elementos a tener en cuenta para la construcción de la solución aplicando $\mathcal{MM}$AS. Además, se presentan algunos detalles de Ingeniería de *Software* y un seudocódigo de la propuesta del algoritmo.

2.2. Representación del problema de Planificación de Secuencias de Ensamble

Una secuencia de ensamble se representa como una lista ordenada de operaciones de ensamble. En esta investigación dicha lista, en orden inverso, representa una secuencia de desensamble. Considerando las direcciones de desensamble de los componentes y las herramientas necesarias para el desensamble, se describe una Operación de desensamble (OD) como una terna OD=(N, D, T), donde N es el identificador del componente en el desensamble, D es la dirección de desensamble del componente y T es la herramienta a utilizar en el desensamble. En este trabajo, cada componente tiene seis posibles direcciones de desensamble a lo largo de los tres ejes principales $(\pm x, \pm y, \pm z)$. Si un ensamble está compuesto por n componentes, se puede construir un grafo de desensamble con $6n$ nodos de operaciones de desensamble.

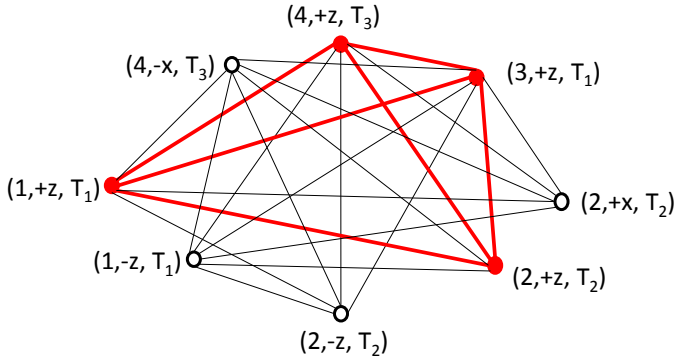


Figura 2.1: Subgrafo de desensamble de la Figura 1.2

La representación del problema ASP mediante un grafo no es más que unir con una arista todo par de nodos que contengan un identificador diferente. A cada una de estas aristas se le hace corresponder un peso, el cual va a tener valor cero si no existe ningún cambio de dirección o de herramienta en el movimiento. Cuando ocurre un cambio de dirección o un cambio de herramienta el peso se incrementa en 1. Encontrar una secuencia de desensamble en el grafo no es más que hallar un ciclo hamiltoniano de menor peso en un subgrafo completo que contenga todos los componentes del ensamble. En la Figura 2.1 se resalta el subgrafo completo donde se encuentra la solución. La búsqueda de la solución en el subgrafo completo es similar al clásico TSP en optimización combinatoria. Vale destacar que el TSP consiste en encontrar la mejor manera de visitar todas las ciudades y volver al punto de partida con la menor distancia de la trayectoria, dado un conjunto de ciudades y la distancia entre cada par de ciudades (Matai, Singh & Mittal, 2010). Las ciudades en el TSP equivalen a los componentes en el ASP y las distancias entre las ciudades equivalen al número total de reorientaciones y cambios de herramientas en el ASP. El objetivo en este último es encontrar un camino con el menor peso. A partir de esta equivalencia es posible utilizar el algoritmo de las hormigas en la solución. En este trabajo el grafo de desensamble completo de un producto no se construye explícitamente en el algoritmo. Mediante la Matriz de Desensamble (MD) en memoria solo se tiene el subgrafo correspondiente a la mitad del número total de nodos del grafo y sus aristas correspondientes. El espacio de solución se genera dinámicamente durante el proceso de búsqueda de las hormigas. De esta forma se reduce la cantidad de memoria necesaria para representar los volúmenes de datos, lo cual tiene un impacto directo en la representación de ensambles complejos.

2.3. Secuencias factibles

Comenzando desde cualquier nodo, una secuencia de desensamble factible o infactible geométricamente se construye visitando todos los nodos que tienen un identificador diferente. Todas estas secuencias conforman el espacio de solución del problema. Una secuencia de ensamble es factible cuando cumple con las restricciones absolutas del problema. Así, cuando se obtiene entre las secuencias de desensamble factible las que sean óptimas, estas a su vez, representan inversamente secuencias de ensamble óptimas.

La Figura 2.1 muestra un subgrafo de desensamble relativo a la Figura 1.2. Desde el nodo $(4, +z, T_3)$, una secuencia de desensamble factible es $\{(4,+z,T_3)$-$(3,+z,T_1)$-$(2,+z,T_2)$-$(1,-z,T_1)\}$.

Cuando las hormigas construyen las secuencias de desensamble usando el grafo de desensamble de los productos, las subsecuencias de operaciones de desensamble que tienen el mismo identificador del componente son evidentemente ilegales ya que un componente no puede ser desensamblado dos veces en un ensamble. Un ejemplo de esto es $\{(1,-z,T_1)$-$(1,+z,T_1)\}$ mostrado en la Figura 2.1. Otra subsecuencia es infactible cuando viola las restricciones geométricas del ensamble, ejemplo $\{(4,+z,T_3)$-$(1,+z,T_1)\}$ mostrado en la Figura 2.1. En el algoritmo se adapta una MD para garantizar la validez y factibilidad de las secuencias. Esta matriz se usa inicialmente para generar las operaciones de desensamble y la lista candidata de operaciones de desensamble. La cantidad de hormigas en el algoritmo es equivalente al número inicial de operaciones de desensamble factibles y estos nodos son fijados como el punto de partida de cada hormiga. Se usa una lista candidata de operaciones de desensamble para restringir la elección de la próxima operación de desensamble durante el proceso de búsqueda de las hormigas.

En este trabajo la MD se utiliza para representar las restricciones geométricas y de precedencia de manera conjunta. Si el componente P_i en el ensamble no interfiere con el componente P_j en la

dirección de los +k-ejes, donde $k \in (x, y, z)$, el componente P_i puede ser desensamblado libremente desde el componente P_j en la dirección de los +k-ejes. De esta forma la MD (2.3.1) se representa como una matriz de n x 3n.

$$\text{MD} = \begin{array}{c} \\ P_1 \\ P_2 \\ \cdots \\ P_n \end{array} \begin{array}{c} \begin{array}{ccccccccccc} x & \overset{P_1}{y} & z & x & \overset{P_2}{y} & z & \cdots & x & \overset{P_n}{y} & z \end{array} \\ \begin{bmatrix} I_{11} & I_{11} & I_{11} & I_{12} & I_{12} & I_{12} & \ldots & I_{1n} & I_{1n} & I_{1n} \\ I_{21} & I_{21} & I_{21} & I_{22} & I_{22} & I_{22} & \ldots & I_{2n} & I_{2n} & I_{2n} \\ & \ldots & & & \ldots & & \ldots & & \ldots & \\ I_{n1} & I_{n1} & I_{n1} & I_{n2} & I_{n2} & I_{n2} & \ldots & I_{nn} & I_{nn} & I_{nn} \end{bmatrix} \end{array}, \tag{2.3.1}$$

donde $I_{ijk} = 1$ si el componente P_i interfiere con el componente P_j durante el movimiento a lo largo de los +k-ejes; en otro caso $I_{ijk} = 0$. La MD inicial del ensamble ilustrativo de la Figura 1.2 es (2.3.2)

$$\text{MD} = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \end{array} \begin{array}{c} \begin{array}{ccccccccccc} x & \overset{1}{y} & z & x & \overset{2}{y} & z & x & \overset{3}{y} & z & x & \overset{4}{y} & z \end{array} \\ \begin{bmatrix} 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \end{array}, \tag{2.3.2}$$

Si el componente P_i puede ser desensamblado libremente desde el componente P_j en la dirección de los +k-ejes, entonces consecuentemente, el componente P_j puede ser desensamblado desde el componente P_i en la dirección de los $-k$-ejes. Si el componente P_i que se mueve en la dirección de los +k-ejes tiene interferencia con otro componente P_j, entonces el componente P_j que se mueve en la dirección de los $-k$-ejes a su vez tendrá interferencia con el componente P_i (Wang et al., 2005). De esta forma se puede inferir conocimiento de la matriz (2.3.2). Si se considera cada terna como una entrada de la matriz, esta tendría dimensiones n x n y su traspuesta brindaría información de la interferencia de cada componente en los $-k$-ejes, e.g.

$$\text{MD}^T = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \end{array} \begin{array}{c} \begin{array}{ccccccccccc} -x & \overset{1}{-y} & -z & -x & \overset{2}{-y} & -z & -x & \overset{3}{-y} & -z & -x & \overset{4}{-y} & -z \end{array} \\ \begin{bmatrix} 0 & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix} \end{array}, \tag{2.3.3}$$

Desde la MD, todas las operaciones de desensamble de cada componente pueden ser derivadas en los pasos de un desensamble. Las operaciones de desensamble factibles del componente P_i a lo largo de los $+k$-ejes ($DO_{i,(+k)}$) se calculan mediante la ecuación (2.3.4).

$$DO_{i,(+k)} = \bigcup_{j=1}^{n} I_{ijk}, \tag{2.3.4}$$

donde $\bigcup$ es el operador booleano OR.

En cuestiones de implementación no es necesario tener la representación de la matriz traspuesta

ya que la información se puede obtener recorriendo en orden inverso la MD. En este sentido las operaciones de desensamble factibles del componente P_i a lo largo de los $-k$-ejes son calculadas mediante las ecuación (2.3.5).

$$DO_{i,(-k)} = \bigcup_{j=1}^{n} I_{jik}, \tag{2.3.5}$$

El resultado puede ser igual a 0 si todos los elementos involucrados en la operación son iguales a 0. De esta manera, el componente P_i puede ser desensamblado libremente en la correspondiente operación obteniéndose así una OD . Si cualquier elemento de los que participan en la operación es igual a 1, el resultado es igual a 1, y el componente P_i no puede ser desensamblado en esa dirección. Para el ensamble mostrado en la Figura 1.2 las operaciones de desensamble factibles calculadas en los $\pm k$-ejes son $(4, +z, T_3)$ y $(1, -z, T_1)$.

2.4. Construcción de la solución aplicando $\mathcal{MM}$AS

2.4.1. Construcción dinámica del grafo

La construcción dinámica del grafo consiste en elegir la próxima OD para desensamblar. Esta elección se realiza a partir de una lista candidata que cada hormiga posee. La lista candidata de cada hormiga contiene todas las operaciones de desensamble que se calculan mediante las ecuaciones (2.3.4) y (2.3.5). La lista candidata inicial del ensamble mostrado en la Figura 1.2 es: $(4, +z, T_3)$ y $(1, -z, T_1)$.

Después que un componente P_i ha sido elegido como el próximo a ser desensamblado, la MD se actualiza. Para realizar la actualización de la MD se toman todos los valores relacionados con el componente desensamblado y se les hace corresponder valor 0. La nueva MD se utiliza para generar la próxima lista candidata a visitar. Si se toma como OD inicial $(1, -z, T_2)$ la MD (2.3.2) actualizada sería (2.4.1).

$$\text{MD} = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \end{array} \begin{array}{c} \begin{array}{cccccccccccc} x & \overset{1}{y} & z & x & \overset{2}{y} & z & x & \overset{3}{y} & z & x & \overset{4}{y} & z \end{array} \\ \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \end{array}. \tag{2.4.1}$$

En consecuencia, la próxima lista candidata de la hormiga que escogió como OD $(1, -z, T_1)$ sería: $(2, +x, T_2)$, $(2, +y, T_2)$, $(3, +x, T_1)$, $(3, +y, T_1)$, $(4, +x, T_3)$, $(4, +y, T_3)$, $(4, +z, T_3)$, $(2, -x, T_2)$, $(2, -y, T_2)$, $(2, -z, T_2)$, $(3, -x, T_1)$, $(3, -y, T_1)$, $(4, -x, T_3)$, $(4, -y, T_3)$. Sin embargo, la hormiga que tiene como OD inicial $(4, +z, T_3)$, actualiza la MD en función de la OD seleccionada y su próxima lista candidata sería: $(3, +z, T_1)$ y $(1, -z, T_1)$.

Para almacenar los componentes que han sido desensamblados, se utiliza una lista tabú, la cual funciona como la memoria de cada hormiga. De esta forma, las operaciones de desensamble que no pertenecen a las dos listas de desensamble citadas anteriormente, no tienen efecto cuando se utiliza la lista candidata para seleccionar la próxima OD.

La MD se actualiza de forma continua conjuntamente con la generación de la lista candidata. Esta última realiza implícitamente la construcción dinámica del grafo de desensamble completo y la

elección de la próxima secuencia de operaciones de desensamble. Al comienzo cada hormiga posee la misma MD. Durante el proceso de búsqueda, cada hormiga genera su lista candidata, su lista tabú, construye su secuencia y actualiza su MD independientemente.

2.4.2. Regla probabilística y actualización de feromona

Durante el proceso de búsqueda cada hormiga construye una secuencia seleccionando la próxima OD de la lista candidata. Esta selección se realiza mediante una regla probabilística de transición entre estados. En este trabajo la regla probabilística se transforma para tratar de manera conjunta las reorientaciones y los cambios de herramientas. Sean α, β y γ la importancia concedida a la feromona, a las reorientaciones y a los cambios de herramientas respectivamente, se sigue la regla probabilística (2.4.2).

$$p_{ij}^{k}(t) = \frac{[\tau_{ij}(t)]^{\alpha}[\eta_{ij}]^{\beta}[\varphi_{ij}]^{\gamma}}{\sum_{l \in N_i^k}[\tau_{il}(t)]^{\alpha}[\eta_{il}]^{\beta}[\varphi_{il}]^{\gamma}}, \quad \text{con} j \in N_i^k, \tag{2.4.2}$$

donde $p_{ij}^{k}(t)$ es la probabilidad con la que, en una iteración t, la hormiga k, situada actualmente en la OD_i, elige OD_j como próximo componente a desensamblar; N_i^k es la lista candidata de la hormiga k generada por la MD, después de que el componente i ha sido desensamblado; $\tau_{ij}(t)$ es la cantidad de feromona acumulada sobre el arco (i, j), en la iteración t; η_{ij} es la información heurística correspondiente al número de reorientaciones desde la OD_i a la OD_j y se toma como (2.4.3),

$$\eta_{ij} = \begin{cases} \eta_1 & : \text{si no hubo reorientación} \\ \eta_2 & : \text{en caso contrario} \end{cases} \tag{2.4.3}$$

donde η_1 y η_2 son constantes positivas. En este trabajo, $\eta_1 = 1$ y $\eta_2 = 0.4$; φ_{ij} es la información heurística correspondiente al número de cambios de herramientas desde la OD_i a la OD_j y se toma como (2.4.4),

$$\varphi_{ij} = \begin{cases} \varphi_1 & : \text{si no hubo cambio de herramienta} \\ \varphi_2 & : \text{en caso contrario} \end{cases} \tag{2.4.4}$$

donde φ_1 y φ_2 son constantes. En este trabajo, $\varphi_1 = 1$ y $\varphi_2 = 0.2$, dando de esta forma más importancia a las reorientaciones.

Cuando todas las hormigas han construido una solución se actualiza en la matriz de feromona el nivel de feromona de cada arista. Sea ρ el coeficiente de evaporación de la feromona, se sigue la ecuación (2.4.5) para actualizar dicha sustancia.

$$\tau_{ij}(t+1) = (1-\rho)\tau_{ij}(t) + \Delta\tau_{ij}^{best}, \quad \Delta\tau_{ij}^{best} = \begin{cases} \frac{1}{L^{best}} & : \text{si el arco } (i,j) \in T^{best} \\ 0 & : \text{en caso contrario} \end{cases} \tag{2.4.5}$$

donde, T^{best} es la mejor solución encontrada en la iteración y L^{best} es la suma de todas las reorientaciones y cambios de herramientas de la solución T^{best}. Vale destacar que el término L^{best} cambia su significado para el problema ASP tratado.

2.4.3. Matriz de feromona

En los algoritmos de ACO, se utiliza una matriz de feromona para representar la memoria compartida de todas las hormigas. Esta guarda las cantidades de feromona depositadas en todas las aristas para guiar la búsqueda de las hormigas. En este trabajo, la matriz de feromona se expresa como una matriz de 6n x 6n ya que el grafo de desensamble completo de un producto tiene 6n nodos. Un elemento de la matriz de feromona denotado como $\tau(i, j)$ indica la preferencia heurística de moverse desde la OD_i a la OD_j. Evidentemente, la matriz de feromona es una matriz asimétrica debido a las restricciones de precedencia entre los componentes de un producto. Inicialmente la matriz de feromona contiene el máximo valor de feromona posible.

2.4.4. Especificación de fichero

La entrada al módulo es un fichero con una estructura bien definida. Así, el fichero está compuesto por una cabecera y dos bloques de información.

Cabecera

En la cabecera aparecen los siguientes parámetros:

NAME: especifica el nombre del caso de estudio.

TYPE: especifica el tipo de problema.

COMMENT: describe el contenido del fichero.

DIMENSION: define la dimensión espacial de los datos.

EXCLUDED_DIRECTION: especifica una dirección excluida de la solución. El valor 0 excluye la dirección del eje x, 1 la dirección del eje y, 2 la dirección del eje z. El valor -1 especifica que los tres ejes están contemplados dentro del espacio de solución.

SIZE: determina la cantidad de partes del ensamble.

Bloques de información

Los bloques de información que contienen el fichero son: la matriz de desensamble del producto y las herramientas utilizadas para ensamblar o desensamblar cada parte.

DISASSEMBLY_MATRIX_SECTION: especifica la matriz de desensamble del producto. Se utiliza durante la búsqueda de la solución.

TOOLS_SECTION: define la herramienta utilizada para el ensamble o el desensamble de cada una de las partes.

Para finalizar el bloque se incluye el caracter de fin de línea **EOF**. El Ejemplo 2.4.4 muestra la especificación de fichero para un ensamble ilustrativo.

Ejemplo 1: Ensamble Ilustrativo

```
NAME: EnsambleIlustrativo.asp
TYPE: ASP
COMMENT: Ejemplo de ensamble ilustrativo tomado del artículo "A novel
    ant colony algorithm for assembly sequence planning"
DIMENSION: 3
EXCLUDED_DIRECTION: -1
SIZE: 4
DISASSEMBLY_MATRIX_SECTION
000 110 110 110
111 000 000 000
111 001 000 000
111 001 001 000
TOOLS_SECTION
2 1 1 3
EOF
```

2.4.5. Propuesta del algoritmo

En general, todos los algoritmos de optimización basados en colonia de hormigas siguen un esquema algorítmico específico, descrito ampliamente en (Maniezzo, Gambardella & De Luigi, 2004; Stützle & Hoos, 2000). El Algoritmo 1 describe la llamada principal para obtener una secuencia de ensamble utilizando la estrategia $\mathcal{MM}$AS.

Algoritmo 1 Obtención de una secuencia de ensamble mediante la estrategia $\mathcal{MM}$AS

```
1: procedure ComputeAssemblySequence()
2:     it ← 0
3:     bestAnt ← 0
4:     globalBestAntSoFar ← 0
5:     Preprocess()
6:     while it < MAX & ¬HasConverged(bestAnt) do
7:         ClearAllTabuLists()
8:         for all ant do
9:             ant.FindSolution()
10:        end for
11:        bestAnt ← antLeastChangesInPath()
12:        GlobalUpdate(bestAnt)
13:        it ← it + 1
14:    end while
15:    if it = MAX then return emptyList                ▷ It didn´t converge
16:    end if
17:    return globalBestAntSoFar.reverseTabu()
18: end procedure
```

El Algoritmo 1 comienza con configuraciones iniciales, representadas en las líneas 2-4. Luego

continúa con una etapa de preprocesamiento en la línea 5. Dentro de esta rutina, se construye una hormiga por cada OD factible que pueda iniciar una secuencia de desensamble. Estas operaciones se determinan mediante las ecuaciones (2.3.4) y (2.3.5). Cada hormiga adiciona su correspondiente OD factible al conjunto de operaciones realizadas, su lista tabu, con el objetivo de recordarlas. Estas listas se utilizan para mantener una memoria en las hormigas de las OD ya visitadas y prohibir que puedan volver sobre estas en una misma iteración.

El algoritmo entra en un bucle en la línea 6 donde: mientras no se exceda de una cantidad de ciclos máxima definida (MAX) y el algoritmo no haya convergido, se ejecutará el bloque principal. En la línea 7 se eliminan las entradas de todas las listas tabu de las hormigas. Vale destacar que el proceso de eliminación no incluye la primera OD de cada lista tabu, pues esta constituye una operación inicial invariante para cada hormiga durante la ejecución del algoritmo.

Luego en la línea 9, cada hormiga realiza su búsqueda de solución al problema planteado de manera probabilística, guiadas por la ecuación (2.4.2). Cuando todas las hormigas han encontrado su solución el algoritmo elige, en la línea 11, la mejor hormiga de la iteración; aquella que ha encontrado la secuencia de desensamble con la menor cantidad total de reorientaciones y cambios de herramientas. Si en ese instante se detecta una hormiga mejor que la mejor hormiga global hallada hasta la iteración en curso (globalBestAntSoFar), esta se actualiza y con ella el límite máximo de feromona. Vale destacar que la comparación es estricta. Con la mejor hormiga de la iteración se actualizan, en la línea 12, los rastros de feromona de todo el grafo siguiendo la ecuación (2.4.5). El bucle continúa hasta tanto no se cumpla alguna condición de parada: se alcance la cantidad máxima de ciclos permisibles o el algoritmo converja de manera natural, ver Sección §1.4.3.

2.5. Ingeniería de *Software*

2.5.1. Diagrama de clases del módulo

La Figura 2.2 muestra el diagrama de clases del módulo. Dicho diagrama recoge todas las clases involucradas en el algoritmo y las relaciones entre ellas.

La clase ASPMMASystem tiene la responsabilidad de ejecutar las operaciones principales del algoritmo. Esta clase se encarga del cálculo de la secuencia de ensamble, de la actualización y de la convergencia. En el caso de la clase ASPAnts representa la hormiga artificial la cual contiene todos los atributos necesarios para la búsqueda de la solución. Adicionalmente, se definieron las clases DisassemblyMatrix y PheromoneMatrix con el objetivo de facilitar el acceso y la modificación de la información almacenada respecto a la MD y a la matriz de feromona. Se representa además, cada OD como un conjunto formado por el identificador de un componente, una dirección de desensamble y la herramienta utilizada en la operación.

2.5.2. Diagrama de actividades del algoritmo

El diagrama de actividades de la Figura 2.3 ilustra los pasos principales del algoritmo propuesto.

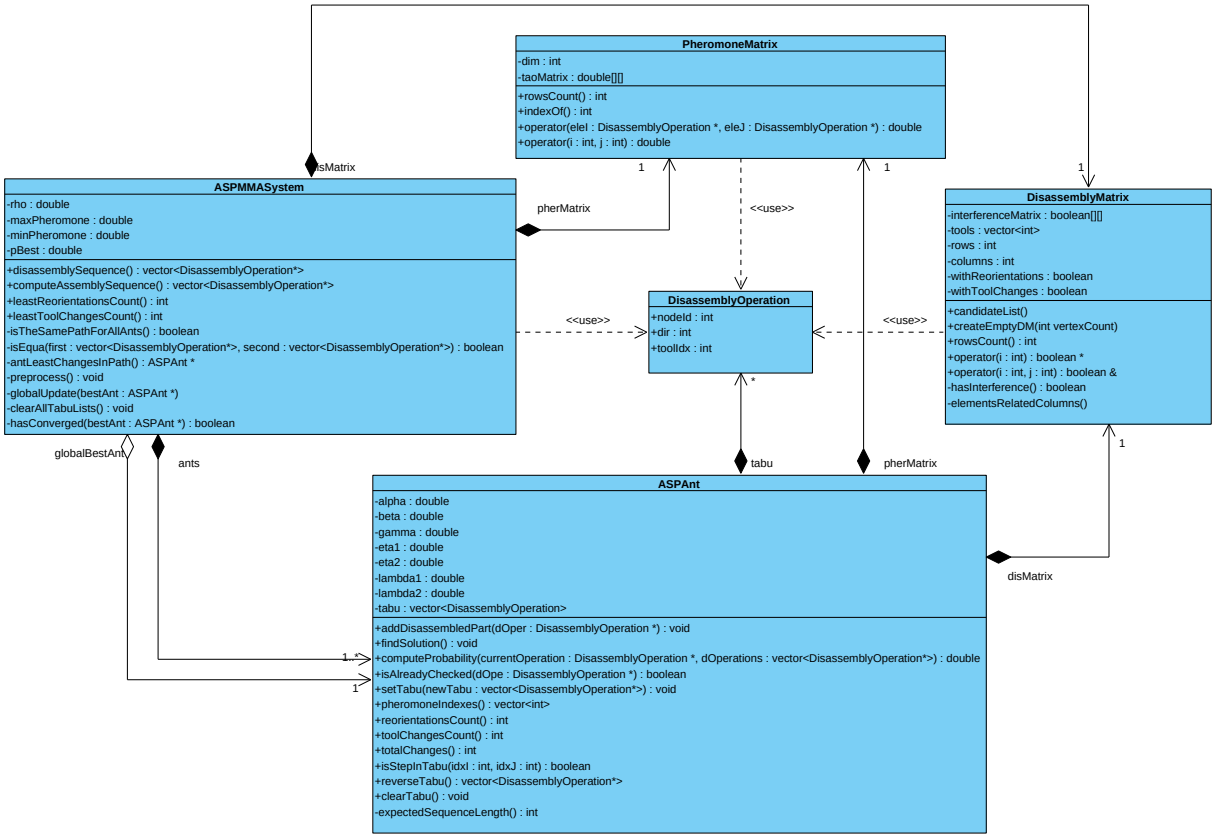


Figura 2.2: Diagrama de clases del algoritmo

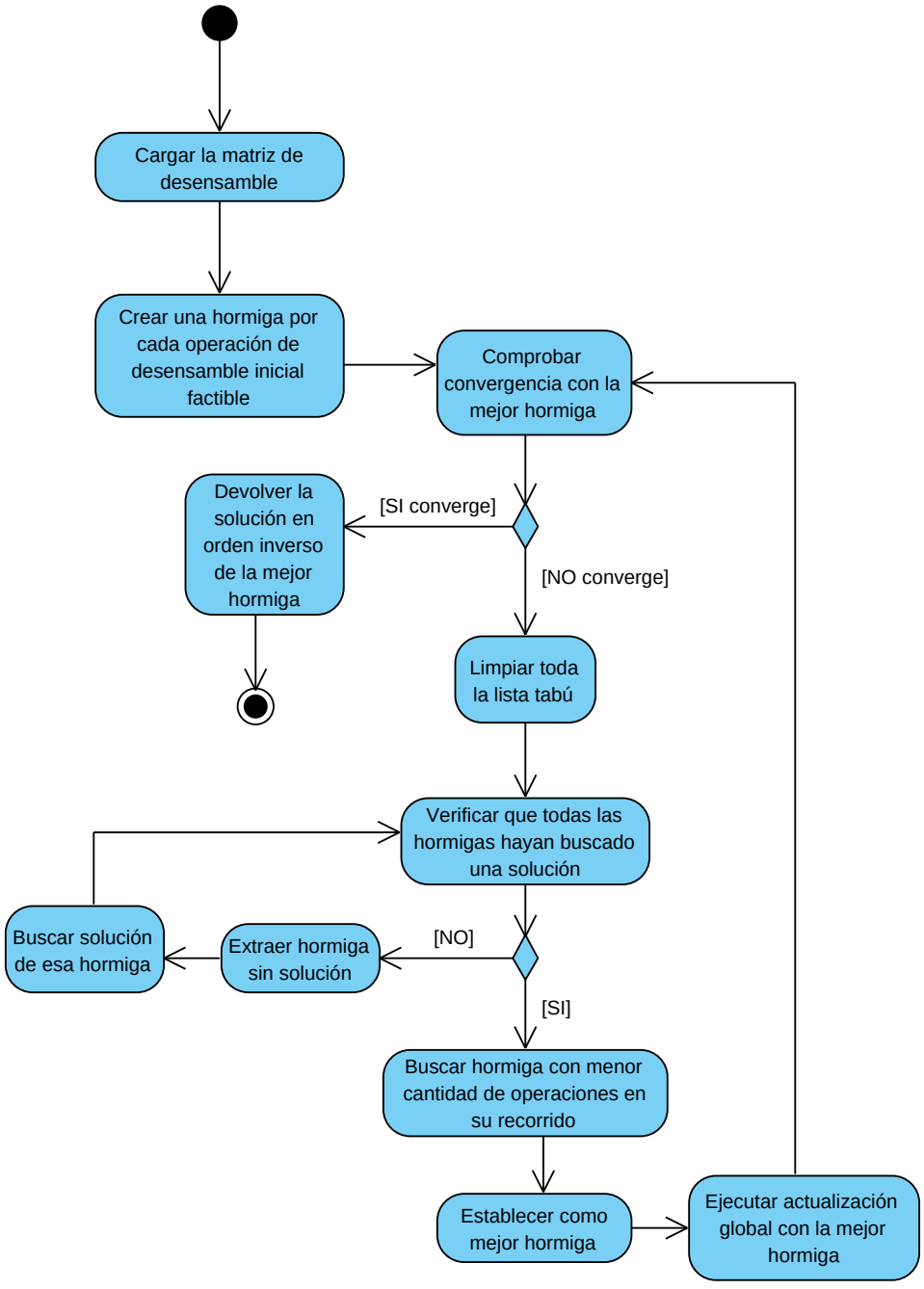


Figura 2.3: Diagrama de actividades del algoritmo

CAPÍTULO 3

Resultados y Discusión

3.1. Introducción

En este capítulo se realiza un estudio de los resultados obtenidos con la implementación propuesta. Para comprobar los resultados se construyó un entorno de prueba que permite elegir indistintamente las dos restricciones de optimización abordadas en la investigación. Además, permite visualizar la secuencia de ensamble y desensamble propuesta por el algoritmo como solución. La combinación de teclas Ctrl + d visualiza la secuencia de desensamble. Si se desea retornar a la secuencia de ensamble se debe utilizar la combinación Ctrl + a. Como valor agregado el entorno genera un reporte. Este posibilita que un operario contraste la salida propuesta con las secuencias de ensamble encontradas por el resto de las hormigas involucradas en el caso de estudio. Además, el entorno de prueba permite guardar cada una de las corridas. Así se reducen los análisis para un mismo producto. Como resultado se puede crear una base de conocimiento. Desde un punto de vista algorítmico vale recordar que el proceso de actualización de globalBestAntSoFar en el Algoritmo 1.12, ocurre únicamente si se cumple una comparación estricta. Si se encuentra otra hormiga con igual cantidad de reorientaciones y cambios de herramientas, globalBestAntSoFar no se actualiza. De esta forma se obtiene una secuencia adicional para la toma de decisiones. La hormiga globalBestAntSoFar constituye una copia de la mejor hormiga global hasta la iteración en curso. Esta última pudiera evolucionar a una secuencia diferente pero con igual número de reorientaciones y cambios de herramientas. En este caso, globalBestAntSoFar conservaría una secuencia igual de competitiva que la mejor hormiga global al finalizar el algoritmo. Conservar esta secuencia es de vital importancia pues posee generalmente el mejor sentido práctico. En el reporte se incluye dicha salida adicional (secuencia de salida premiada), de existir para el caso de estudio.

Para validar la implementación se seleccionaron casos de estudio recogidos en artículos publicados en revistas referenciadas del tema. Adicionalmente, se verificó el buen desempeño del algoritmo con casos de estudio relativos a un taladro de perforación de petróleo de 2000 HP, cortesía del proyecto CDSEM.

En la investigación se constata la efectividad del algoritmo $\mathcal{MM}$AS para resolver los problemas ASP. Además, se demuestra la utilidad de incluir los cambios de herramientas en la regla probabilística, lo cual se puede comprobar con los resultados mostrados en este capítulo. Vale destacar que la inclusión de los cambios de herramientas en la regla probabilística conlleva a modificar la interpretación de la variable L^{best} en la ecuación (2.4.5). En esta investigación L^{best} se define como el total de reorientaciones y cambios de herramientas.

Todas las pruebas realizadas se ejecutaron en un ordenador ASUS con procesador Intel Core i5-2430M de segunda generación a 2.4 GHz, con una memoria RAM de 4GB. Los valores de las constantes utilizadas en las corridas del algoritmo para los distintos juegos de datos están recogidos en la Tabla 3.1. La selección de estos valores fue realizada a partir de los resultados de Stützle y Hoos (Stützle & Hoos, 2000).

Tabla 3.1: Valores de las constantes utilizadas en la corrida del algoritmo

Parámetro	Símbolo	Valor
Importancia de la feromona	α	1
Importancia de la heurística de reorientación	β	2
Importancia de la heurística de cambios de herramientas	γ	2
Información heurística de reorientaciones sin cambio de dirección	η_1	1
Información heurística de reorientaciones con cambio de dirección	η_2	0,4
Información heurística de cambios de herramientas sin cambio	φ_1	1
Información heurística de cambios de herramientas con cambio	φ_2	0,2
Factor de evaporación de la feromona	ρ	0,02
Probabilidad con la que se encuentra la mejor solución	P_{best}	0,05

3.2. Casos de estudio tomados de artículos en revistas referenciadas

3.2.1. Ensamble industrial

La Figura 3.1 muestra el caso de estudio Ensamble industrial, anteriormente discutido en la literatura (Wang & Liu, 2010). El modelo cuenta con un total de 11 componentes en dos ejes. En este ensamble los tornillos no son considerados. La matriz de desensamble de la Figura 3.1 se muestra en la Sección de fichero 1.

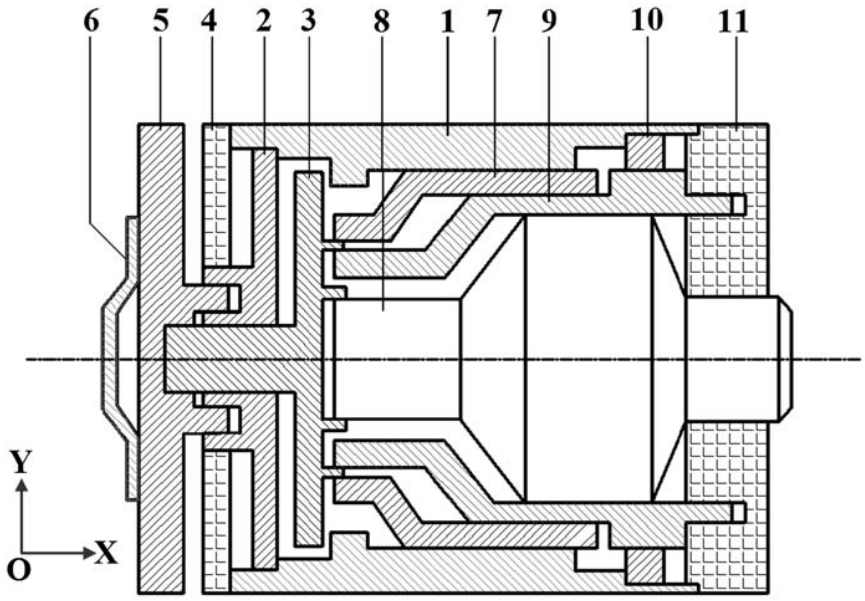


Figura 3.1: Ensamble industrial

Sección de fichero 1: Ensamble industrial

DISASSEMBLY_MATRIX_SECTION

00	01	01	00	00	00	11	01	11	11	11
11	00	11	01	01	00	10	10	10	10	10
11	01	00	01	01	00	11	11	11	00	10
10	11	11	00	01	00	10	10	10	10	10
10	11	11	11	00	00	10	10	10	10	10
00	10	10	10	10	00	10	10	10	00	10
01	00	01	00	00	00	00	11	11	00	10
01	00	01	00	00	00	01	00	01	01	11
01	00	01	00	00	00	01	11	00	01	11
01	00	00	00	00	00	00	01	11	00	10
01	00	00	00	00	00	00	01	01	00	00

La Figura 3.2 muestra la salida del algoritmo sin cambios de herramientas. Esta salida presenta cero reorientaciones. Este comportamiento es semejante al obtenido por Wang y Liu (2010), ilustrado en la Tabla 3.7.

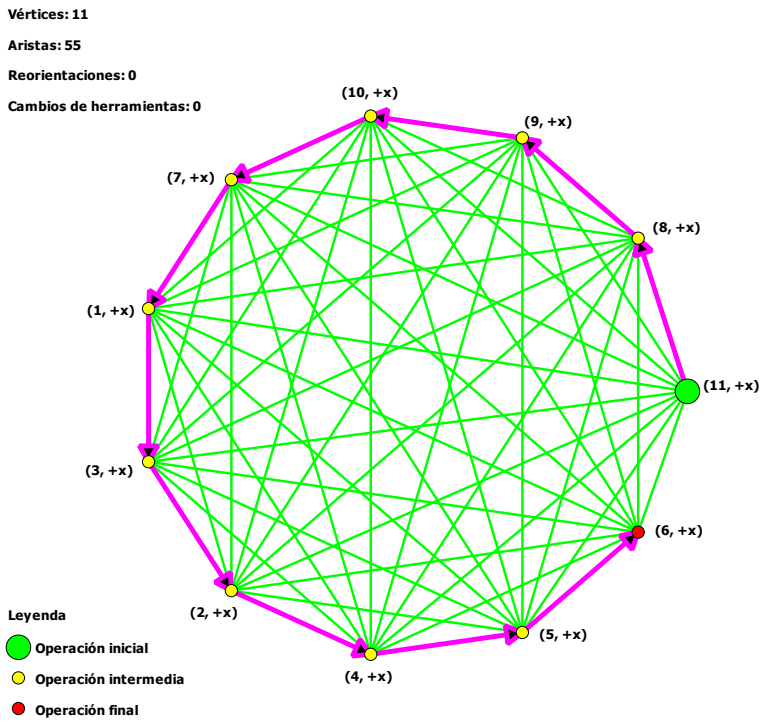


Figura 3.2: Salida del algoritmo para el Ensamble industrial

En el reporte Ensamble industrial para este caso de estudio se reflejaron dos salidas con igual cantidad de reorientaciones que la salida principal propuesta por el algoritmo. Vale destacar que la hormiga 1 brinda una secuencia elegible, mientras que la hormiga 4 carece de sentido práctico. La salida mostrada por el algoritmo pertenece a la secuencia premiada del caso de estudio.

3.2.2. Controlador industrial

La Figura 3.3 muestra el caso de estudio Controlador industrial, analizado anteriormente en (Boothroyd, 1994). El modelo cuenta con un total de 16 componentes a lo largo de los tres ejes principales. La matriz de desensamble de la Figura 3.3 se muestra en la Sección de fichero 2.

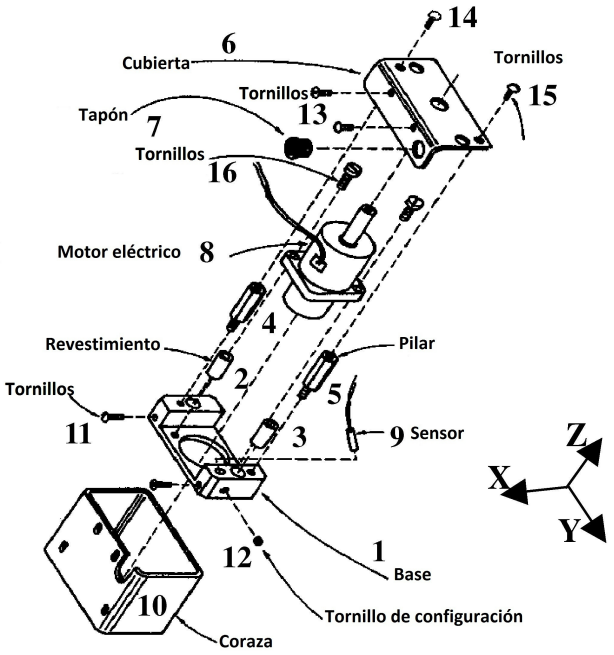


Figura 3.3: Controlador industrial

Sección de fichero 2: Controlador industrial

DISASSEMBLY_MATRIX_SECTION

000	111	111	111	111	001	001	111	111	110	111	111	001	001	001	111
110	000	010	100	000	001	000	010	000	110	100	000	000	000	000	000
110	000	000	000	000	001	000	000	100	110	100	100	000	000	000	000
110	000	000	000	000	101	010	010	010	110	100	010	010	111	000	010
110	000	100	000	000	101	100	000	100	110	100	100	000	000	111	000
000	000	000	010	010	000	111	110	000	000	000	000	111	111	111	000
000	000	000	000	000	011	000	000	000	010	000	000	000	000	000	000
110	000	010	100	010	111	110	000	010	110	000	010	111	000	010	111
110	000	000	000	000	001	001	000	000	110	110	110	101	000	000	000
111	111	111	111	111	101	111	111	111	000	111	111	111	111	111	111
011	000	000	001	000	001	001	010	001	011	000	010	000	001	000	010
101	000	000	000	000	101	001	000	101	110	100	000	001	000	000	000
000	000	000	000	000	011	010	010	010	011	000	000	000	000	000	000
000	000	000	110	000	110	010	010	010	110	000	000	010	000	000	000
000	000	000	000	110	110	100	000	100	110	000	000	000	000	000	000
110	000	000	000	010	101	100	110	010	110	010	010	001	000	010	000

La Figura 3.4 muestra la salida del algoritmo sin cambios de herramientas. Esta salida presenta tres reorientaciones. Este comportamiento es comparable con el obtenido por Boothroyd (1994), ilustrado en la Tabla 3.7. Vale destacar que aunque ambas salidas presentan la misma cantidad de reorientaciones, la secuencia obtenida en esta investigación es distinta, pero aun así válida.

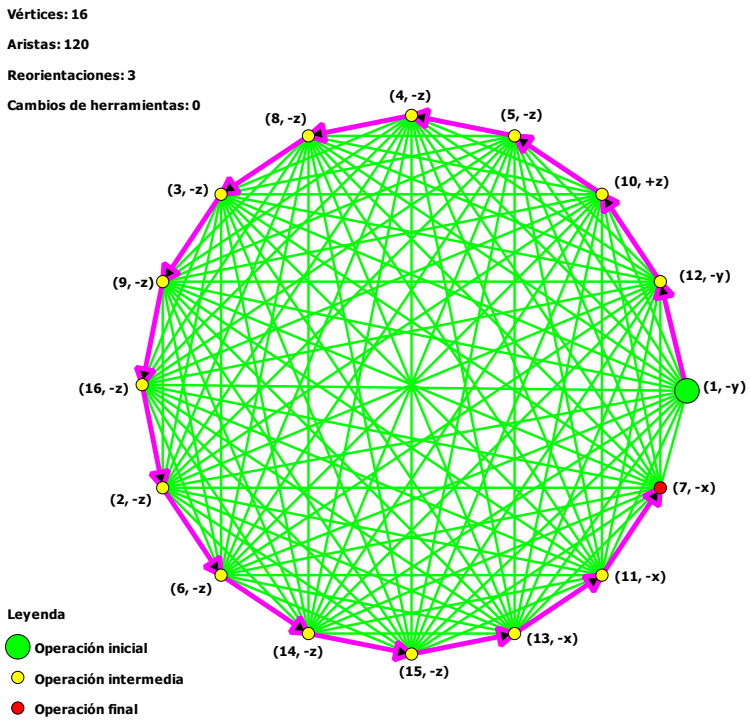


Figura 3.4: Salida del algoritmo para el ensamble del Controlador industrial

En el reporte Controlador industrial para este caso de estudio sin considerar los cambios de herramientas se reflejaron dos salidas con igual cantidad de reorientaciones que la salida principal propuesta por el algoritmo. Vale destacar que ambas secuencias obtenidas en el reporte carecen de sentido práctico. La salida mostrada por el algoritmo pertenece a la secuencia premiada del caso de estudio.

La Figura 3.5 muestra la salida del algoritmo considerando los cambios de herramientas. Esta salida presenta cuatro reorientaciones y dos cambios de herramientas. Es válido destacar que al contemplar los cambios de herramientas en el modelo, la salida toma mayor sentido en la práctica pues se ensambla completamente el núcleo del controlador y después se introduce en la coraza.

En el reporte para este caso de estudio considerando los cambios de herramientas se reflejó una salida con igual cantidad de reorientaciones y cambios de herramientas que la salida principal propuesta por el algoritmo. La Tabla 3.2 muestra dicho resultado. Vale destacar que la hormiga 1 brinda una secuencia elegible. La salida mostrada por el algoritmo pertenece a la secuencia premiada del caso de estudio. Por otro lado las herramientas utilizadas para el ensamble del Controlador industrial se muestran en la Tabla 3.3.

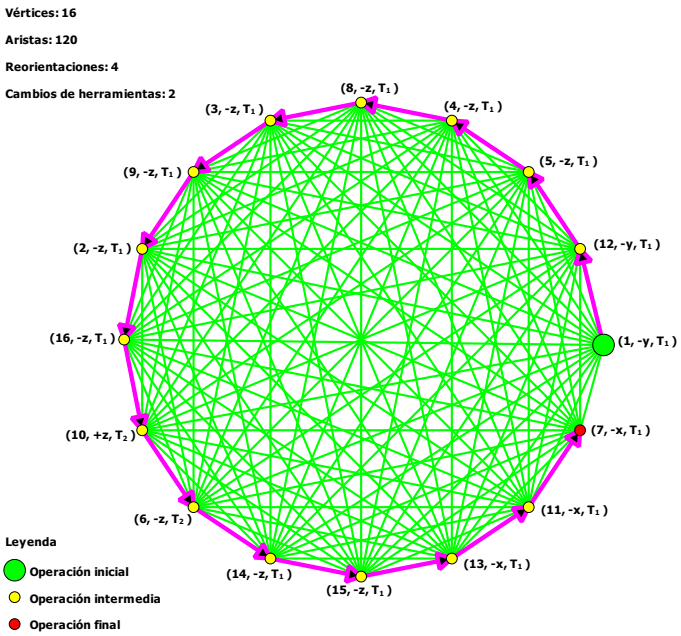


Figura 3.5: Salida del algoritmo para el ensamble del Controlador industrial con cambios de herramientas

Tabla 3.2: Secuencias del Controlador industrial con igual cantidad de reorientaciones y cambios de herramientas

Hormiga	Reorientaciones	Cambios de herramientas	Salida
1	4	2	(12, +y, T_1)-(1, +y, T_1)-(4, -z, T_1)-(3, -z, T_1)-(5, -z, T_1)-(8, -z, T_1)-(2, -z, T_1)-(9, -z, T_1)-(16, -z, T_1)-(10, +z, T_2)-(6, -z, T_2)-(15, -z, T_1)-(14, -z, T_1)-(13, -x, T_1)-(11, -x, T_1)-(7, -x, T_1)

Tabla 3.3: Herramientas para el ensamble de cada componente del Controlador industrial

Número	Nombre del componente	Herramienta
1	Base	T_1
2	Revestimiento	T_1
3	Revestimiento	T_1
4	Pilar	T_1
5	Pilar	T_1
6	Cubierta	T_2
7	Tapón	T_1
8	Motor eléctrico	T_1
9	Sensor	T_1
10	Coraza	T_2
11	Tornillos	T_1
12	Tornillo de configuración	T_1
13	Tornillos	T_1
14	Tornillos	T_1
15	Tornillos	T_1
16	Tornillos	T_1

3.2.3. Sistema de poleas

La Figura 3.6 muestra el caso de estudio Sistema de poleas, analizado anteriormente en (Zhang, Sun & He, 2010). El modelo cuenta con un total de 16 componentes a lo largo de los tres ejes principales. La matriz de desensamble de la Figura 3.6 se muestra en la Sección de fichero 3.

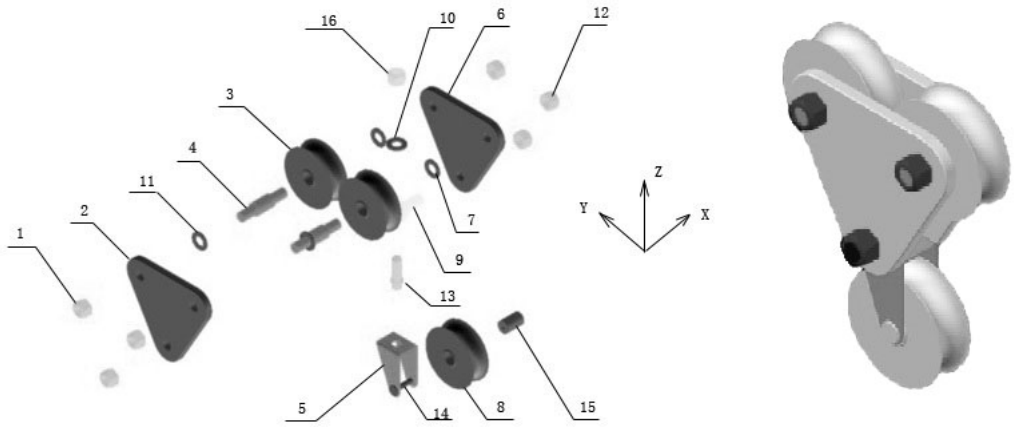


Figura 3.6: Sistema de poleas

Sección de fichero 3: Sistema de poleas

DISASSEMBLY_MATRIX_SECTION

000	100	100	111	000	100	100	000	111	100	100	100	100	000	000	100
000	000	100	111	100	100	100	000	111	100	100	100	100	000	000	100
000	000	000	011	000	100	100	000	000	000	000	100	000	000	000	000
011	011	111	000	000	111	000	000	000	000	011	111	000	000	000	000
000	000	001	001	000	100	100	100	001	001	000	100	111	011	100	001
000	000	000	011	000	000	000	000	011	000	000	100	000	000	000	000
000	000	000	011	000	100	000	000	011	000	000	100	000	000	000	000
000	000	001	001	101	000	001	000	001	001	001	000	001	011	011	001
011	011	001	000	000	111	111	000	000	000	000	111	110	000	000	111
000	000	000	011	000	100	000	000	111	000	000	100	110	000	000	001
000	000	100	111	000	100	100	000	000	000	000	100	000	000	000	000
000	000	000	011	000	000	000	000	011	000	000	000	000	000	000	000
000	000	001	000	110	100	100	000	111	111	000	100	000	000	000	111
000	001	100	000	111	001	000	111	001	001	000	000	001	000	111	001
000	001	001	000	101	001	000	011	001	001	000	000	001	011	000	001
000	000	101	000	000	000	000	000	000	000	000	100	110	000	000	000

La Figura 3.7 muestra la salida del algoritmo sin cambios de herramientas. Este resultado presenta tres reorientaciones. Vale notar que el algoritmo propuesto mejora el número de operaciones de ensamble en un 70 % en relación a la salida recogida en (Zhang et al., 2010). Este resultado se puede constatar en la Tabla 3.7.

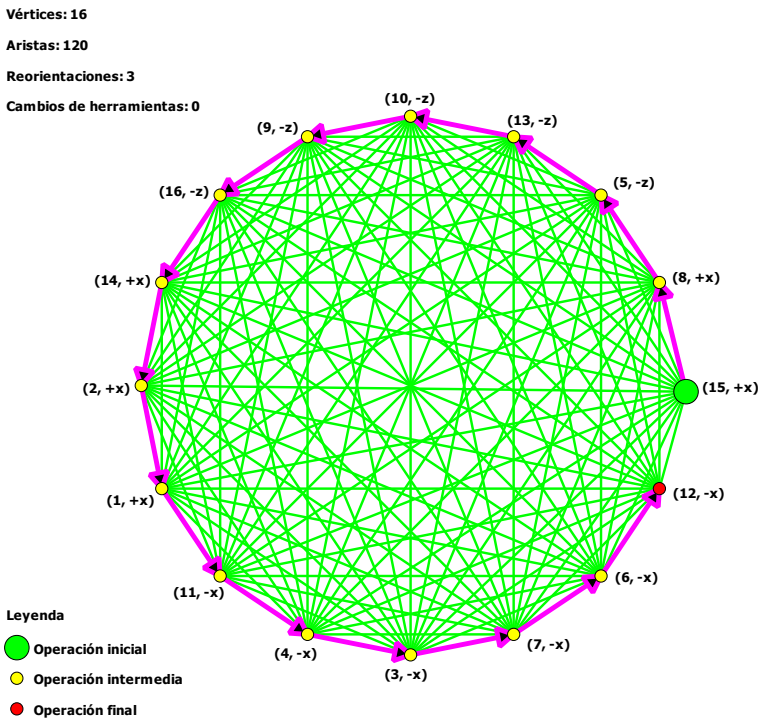


Figura 3.7: Salida del algoritmo para el ensamble del Sistema de poleas

En el reporte Sistema de poleas sin considerar los cambios de herramientas se reflejaron dos salidas con igual cantidad de reorientaciones que la salida principal propuesta por el algoritmo. Vale destacar que ambas secuencias presentan menor sentido práctico en relación a la salida mostrada por el algoritmo. Esta última pertenece a la secuencia adicional del caso de estudio.

La Figura 3.8 muestra la salida considerando los cambios de herramientas. Este resultado presenta cinco reorientaciones y doce cambios de herramientas. Vale notar que el algoritmo propuesto mejora el número de operaciones de ensamble en un 29.2 % en relación a la salida recogida en la literatura. Este resultado se puede constatar en la Tabla 3.7.

En el reporte Sistema de poleas considerando los cambios de herramientas se reflejó una salida con igual cantidad de reorientaciones y cambios de herramientas que la salida principal propuesta por el algoritmo. Vale destacar que la hormiga 2 brinda una secuencia con un mayor sentido práctico que la salida mostrada por el algoritmo. Esta última pertenece a la secuencia premiada del caso de estudio. Por otro lado las herramientas utilizadas para el ensamble del Sistema de poleas se muestran en la Tabla 3.4.

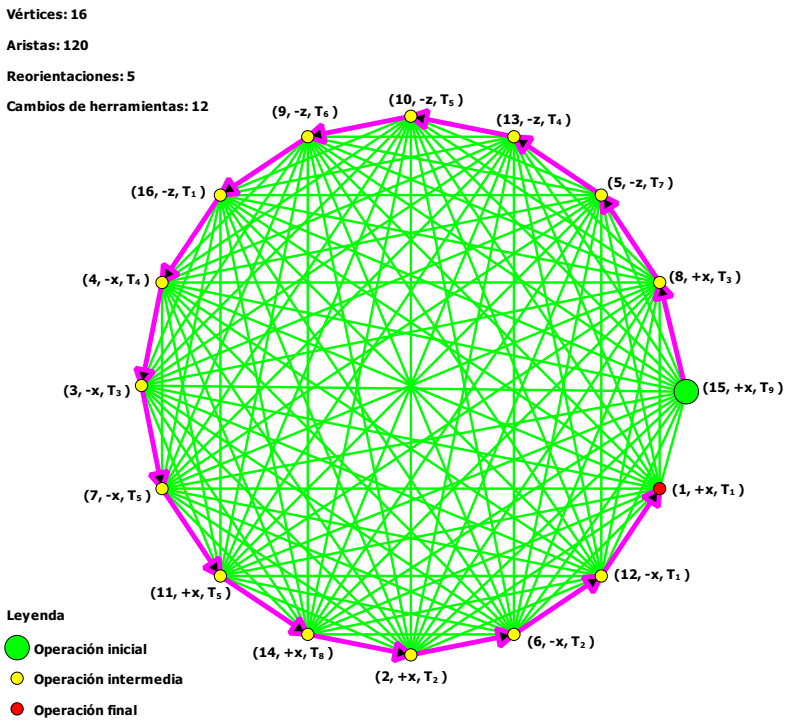


Figura 3.8: Salida del algoritmo para el ensamble del Sistema de poleas con cambios de herramientas

Tabla 3.4: Herramientas para el ensamble de cada componente del Sistema de poleas

Número	Nombre del componente	Herramienta
1	Tuerca	T_1
2	Placa fija	T_2
3	Rueda	T_3
4	Eje	T_4
5	Armadura de la rueda	T_7
6	Placa fija	T_2
7	Junta	T_5
8	Rueda	T_3
9	Eje central	T_6
10	Junta	T_5
11	Junta	T_5
12	Tuerca	T_1
13	Eje	T_4
14	Pasador	T_8
15	Fijador de rueda	T_9
16	Tuerca	T_1

3.2.4. Generador

La Figura 3.9 muestra el ensamble de un generador discutido anteriormente en (Wang & Liu, 2010). El modelo cuenta con un total de 15 componentes en los tres ejes principales. La matriz de desensamble de la Figura 3.9 se muestra en la Sección de fichero 4.

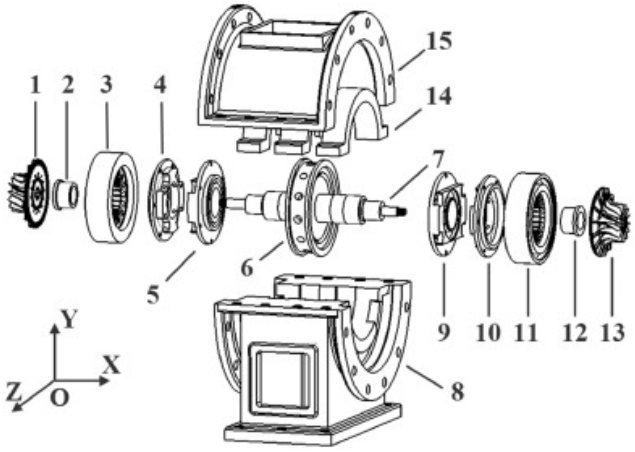


Figura 3.9: Ensamble de un generador

Sección de fichero 4: Generador

DISASSEMBLY_MATRIX_SECTION

000	110	110	100	100	100	111	101	100	100	100	100	100	100	110
011	000	111	100	100	100	111	101	100	100	100	100	100	110	110
011	011	000	111	100	100	111	101	100	100	100	100	100	110	110
000	000	011	000	111	100	111	101	100	100	100	100	100	110	110
000	000	000	011	000	111	111	101	100	100	100	100	100	110	110
000	000	000	011	001	000	011	101	111	100	100	100	100	110	110
011	011	011	011	011	011	000	001	111	111	111	111	111	010	010
010	010	110	010	010	110	010	000	110	110	110	010	010	110	110
000	000	000	000	000	011	011	101	000	111	100	100	100	110	010
000	000	000	000	000	000	011	101	011	000	111	100	100	110	010
000	000	000	000	000	000	011	101	000	011	000	111	100	110	010
000	000	000	000	000	000	011	001	000	000	011	000	100	010	010
000	000	000	000	000	000	011	001	000	000	000	011	000	010	010
101	101	101	101	101	101	101	101	101	101	101	101	101	000	110
101	101	101	101	101	101	101	101	101	101	101	101	101	101	000

La Figura 3.10 muestra la salida del algoritmo sin cambios de herramientas. Esta salida presenta dos reorientaciones. Es válido destacar que la subsecuencia 1-2-3-4-5-7 carece de sentido práctico, pues se está ensamblando un componente grande, 7, en otros más pequeños 1-2-3-4-5 cuando lo natural sería realizar el proceso inverso. Estas cuestiones se mitigan cuando se le añade al modelo los cambios de herramientas, cuya salida se muestra en la Figura 3.11.

En el reporte Generador sin considerar los cambios de herramientas se reflejaron dos salidas con igual cantidad de reorientaciones que la salida principal propuesta por el algoritmo. Vale destacar que ambas secuencias son elegibles. La salida mostrada por el algoritmo pertenece a la secuencia adicional del caso de estudio. Esta salida presenta tres reorientaciones y dos cambios de herramientas. Vale destacar que al considerar los cambios de herramientas en el modelo, la salida de la secuencia adquiere mayor sentido práctico.

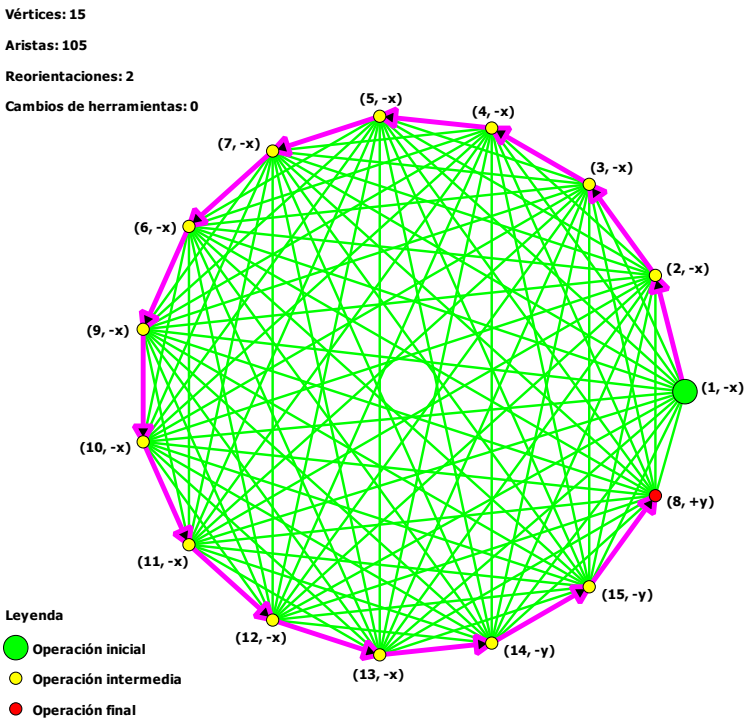


Figura 3.10: Salida del algoritmo para el Generador

En el reporte para este caso de estudio considerando los cambios de herramientas se reflejaron tres salidas con igual cantidad total de reorientaciones y cambios de herramientas que la salida principal propuesta por el algoritmo. La Tabla 3.5 muestra dichos resultados. Vale destacar que la hormiga 1 brinda una secuencia perfectamente elegible, mientras que las salidas de las hormigas 2 y 5 carecen de sentido práctico. La salida mostrada por el algoritmo pertenece a la secuencia adicional del caso de estudio. Por otro lado las herramientas utilizadas para el ensamble del generador se muestran en la Tabla 3.6.

Tabla 3.5: Secuencias del Generador con igual cantidad total de reorientaciones y cambios de herramientas

Hormiga	Reorientaciones	Cambios de herramientas	Salida
1	3	2	(7, -x, T_2)-(9, -x, T_1)-(10, -x, T_1)-(11, -x, T_1)-(12, -x, T_1)-(13, -x, T_1)-(6, +x, T_1)-(5, +x, T_1)-(4, +x, T_1)-(3, +x, T_1)-(2, +x, T_1)-(1, +x, T_1)-(14, +z, T_3)-(15, +z, T_3)-(8, +y, T_3)
2	2	3	(1, -x, T_1)-(2, -x, T_1)-(3, -x, T_1)-(4, -x, T_1)-(5, -x, T_1)-(6, -x, T_1)-(7, -x, T_2)-(9, -x, T_1)-(10, -x, T_1)-(11, -x, T_1)-(12, -x, T_1)-(13, -x, T_1)-(14, -y, T_3)-(15, -y, T_3)-(8, -z, T_3)
3	3	2	(1, -x, T_1)-(2, -x, T_1)-(3, -x, T_1)-(4, -x, T_1)-(5, -x, T_1)-(6, -x, T_1)-(7, -x, T_2)-(9, -x, T_1)-(10, -x, T_1)-(11, -x, T_1)-(12, -x, T_1)-(13, -x, T_1)-(8, +y, T_3)-(14, -y, T_3)-(15, -y, T_3)

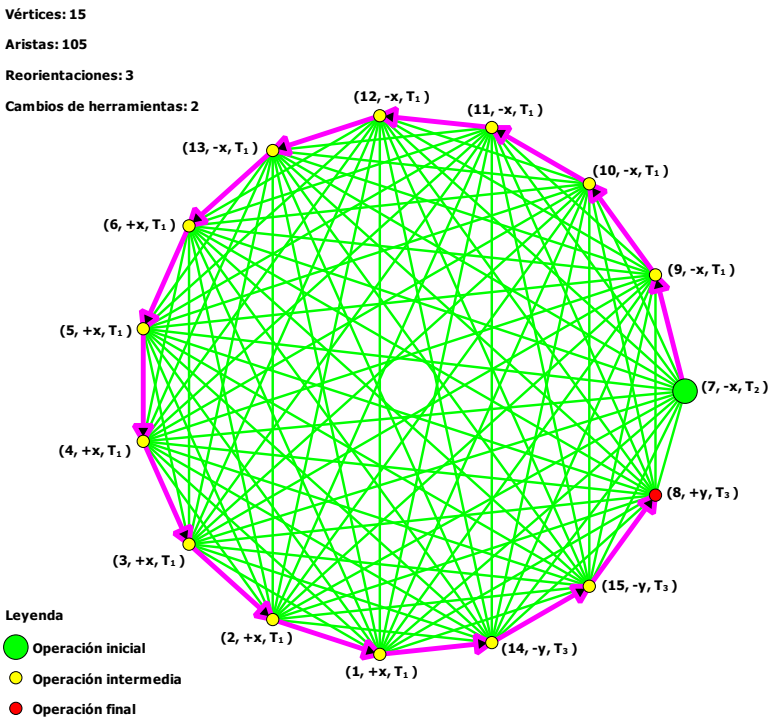


Figura 3.11: Salida del algoritmo para el Generador considerando los cambios de herramientas

Tabla 3.6: Herramientas para el ensamble de cada componente del Generador

Número del componente	Herramienta
1	T_1
2	T_1
3	T_1
4	T_1
5	T_1
6	T_1
7	T_2
8	T_3
9	T_1
10	T_1
11	T_1
12	T_1
13	T_1
14	T_3
15	T_3

Tabla 3.7: Comparación de las salidas

Caso de estudio	Salida en la literatura	Reorientaciones	Cambios de herramientas	Salida del algoritmo	Reorientaciones	Cambios de herramientas
Ensamble industrial	11-8-9-7-10-1-3-2-4-5-6	0	0	11-8-9-10-7-1-3-2-4-5-6	0	0
Controlador industrial	(1,-z)-(8,-z)-(16,-z)-(3,-z)-(5,-z)-(4,-z)-(9,-z)-(2,-z)-(6,-z)-(14,-z)-(15,-z)-(12,-y)-(10,+z)-(7,-x)-(11,-x)-(13,-x)	3	0	(1,-y)-(12,-y)-(10,+z)-(5,-z)-(4,-z)-(8,-z)-(3,-z)-(9,-z)-(16,-z)-(2,-z)-(6,-z)-(14,-z)-(15,-z)-(13,-x)-(11,-x)-(7,-x)	3	0
Sistema de poleas	(5,+z)-(8,+z)-(15,-x)-(14,+x)-(13,-x)-(10,-z)-(9,-x)-(16,-z)-(2,+x)-(11,+x)-(4,-x)-(1,x)-(3,-x)-(7,-x)-(6,-x)-(12,-x)	10	0	(15,+x)-(8,+x)-(5,-z)-(13,-z)-(10,-z)-(9,-z)-(16,-z)-(14,+x)-(2,+x)-(1,+x)-(11,-x)-(4,-x)-(3,-x)-(7,-x)-(6,-x)-(12,-x)	3	0
Sistema de poleas con herramientas	(5,+z,T_7)-(8,+z,T_3)-(15,-x,T_9)-(14,+x,T_8)-(13,-x,T_4)-(10,-z,T_5)-(9,-x,T_6)-(16,-z,T_1)-(2,+x,T_2)- (11,+x,T_7)-(4,-x,T_4)-(1,+x,T_1)-(3,-x,T_3)-(7,-x,T_5)-(6,-x,T_2)-(12,-x,T_1)	10	14	(15,+x,T_9)-(8,+x,T_3)-(5,-z,T_7)-(13,-z,T_4)-(10,-z,T_5)-(9,-z,T_6)-(16,-z,T_1)-(4,-x,T_4)-(3,-x,T_3)-(7,-x,T_5) -(11,+x,T_5)-(14,+x,T_8)-(2,+x,T_2)-(6,-x,T_2)-(12,-x,T_1)-(1,+x,T_1)	5	12
Generador con herramientas	(7,-x,T_1)-(9,-x,T_1)-(10,-x,T_1)-(11,-x,T_2)-(6,+x,T_2)-(5,+x,T_2)-(4,+x,T_2)-(3,+x,T_2)-(2,+x,T_4)- (1,+x,T_4)-(12,-x,T_4)-(13,-x,T_4)-(15,-x,T_4)-(14,-x,T_4)-(8,+y,T_4)	3	2	(7,-x,T_2)-(9,-x,T_1)-(10,-x,T_1)-(11,-x,T_1)-(12,-x,T_1)-(13,-x,T_1)-(6,+x,T_1)-(5,+x,T_1)-(4,+x,T_1)- (3,+x,T_1)-(2,+x,T_1)-(1,+x,T_1)-(14,-y,T_3)-(15,-y,T_3)-(8,+y,T_3)	3	2

3.3. Casos de estudio de un taladro de perforación de petróleo de 2000 HP

3.3.1. Sección 1 del cuerpo principal de la cabria

La cabria es uno de los componentes principales del sistema de izaje de un taladro de perforación de petróleo. Ella tiene la función de orientar y sostener los tubos de perforación y de revestimiento al introducirlos o extraerlos del pozo durante el proceso de perforación. La cabria se divide estructuralmente en los subconjuntos siguientes:

- cuerpo principal
- marco tipo A
- componentes auxiliares

La Figura 3.12 muestra la sección 1 del cuerpo principal de la cabria. El modelo cuenta con un total de 18 componentes en los 3 ejes principales. La matriz de desensamble de la Figura 3.12 se

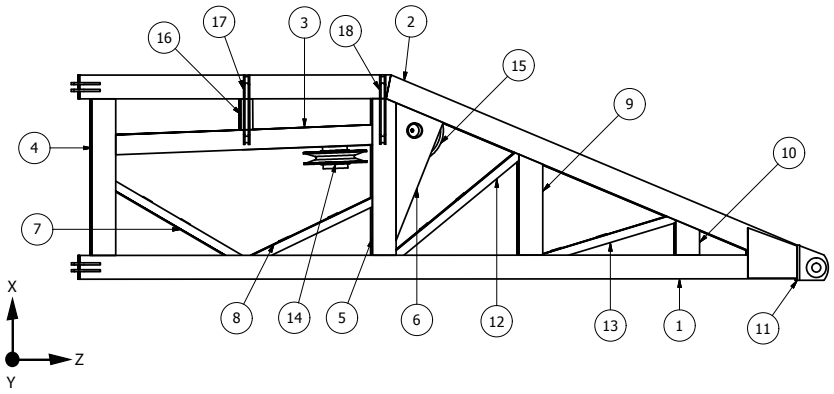


Figura 3.12: Sección 1 del cuerpo principal de la cabria

muestra en la Sección de fichero 5.

Sección de fichero 5: Sección 1 del cuerpo principal de la cabria

DISASSEMBLY_MATRIX_SECTION

000	100	100	110	110	100	110	110	110	110	011	110	110	100	100	100	100	101
000	000	000	010	010	010	000	000	010	010	011	010	010	000	000	010	111	111
000	101	000	010	011	001	000	000	001	000	000	001	000	011	001	110	111	110
010	111	011	000	001	001	011	001	001	001	001	001	001	001	001	001	001	001
010	111	010	000	000	001	000	010	001	001	001	011	001	000	001	000	000	010
000	111	000	000	000	000	000	000	001	001	001	001	001	000	101	000	000	010
010	101	100	010	001	001	000	001	001	001	001	001	001	001	001	100	001	000
010	101	100	000	011	001	000	000	001	001	001	001	001	100	100	100	000	000
010	111	000	000	000	000	000	000	000	001	001	001	011	000	000	000	000	000
010	111	000	000	000	000	000	000	000	000	001	000	010	000	000	000	000	000
010	010	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000

010	111	000	000	010	100	000	000	011	001	001	000	001	000	000	000	000	000
010	111	000	000	000	000	000	000	010	011	001	000	000	000	000	000	000	000
000	101	111	000	001	001	000	000	001	001	001	001	000	000	001	000	000	001
000	111	010	000	010	111	000	000	001	000	000	001	000	000	000	000	000	010
000	111	010	000	010	010	000	000	001	000	000	001	000	001	001	000	001	000
000	000	000	000	000	001	000	000	001	000	000	001	000	000	000	000	000	001
000	000	000	000	000	001	000	000	100	000	000	000	000	000	000	000	000	000

La salida se muestra en la Figura 3.13, la cual presenta dos reorientaciones. En el reporte para este caso de estudio sin considerar los cambios de herramientas se reflejaron dos salidas con igual cantidad de reorientaciones. La Tabla 3.8 muestra dichos resultados. Vale destacar que ambas secuencias son elegibles. La salida mostrada por el algoritmo pertenece a la salida de la hormiga 9. En el reporte de este caso de estudio no existe secuencia adicional.

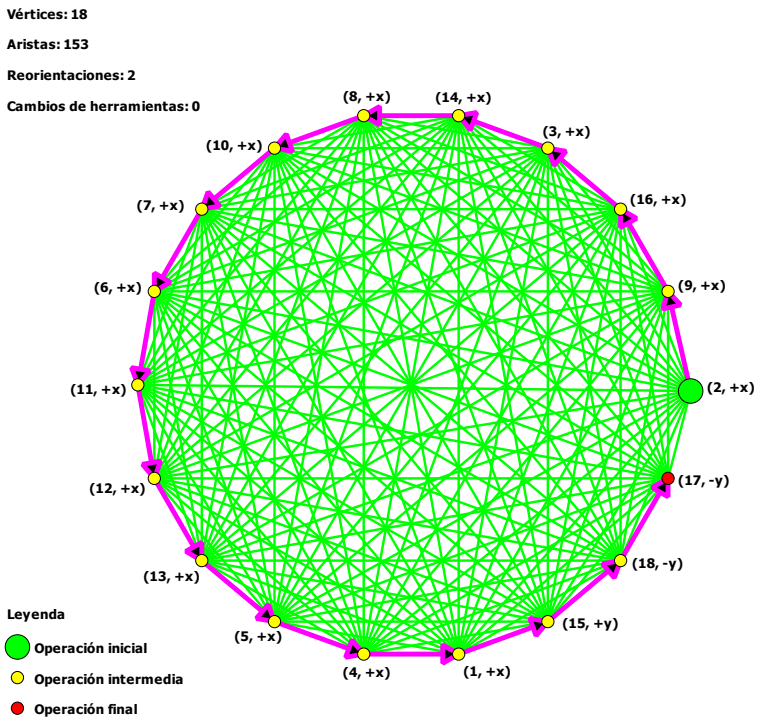


Figura 3.13: Salida del algoritmo para la Sección 1 del cuerpo principal de la cabria

Tabla 3.8: Secuencias de la Sección 1 con igual cantidad de reorientaciones

Hormiga	Salida
9	(2, +x)-(9, +x)-(16, +x)-(3, +x)-(14, +x)-(8, +x)-(10, +x)-(7, +x)-(6, +x)-(11, +x)-(12, +x)-(13, +x)-(5, +x)-(4, +x)-(1, +x)-(15, +y)-(18, -y)-(17, -y)
10	(17, +x)-(2, +x)-(9, +x)-(16, +x)-(3, +x)-(14, +x)-(8, +x)-(10, +x)-(7, +x)-(6, +x)-(11, +x)-(12, +x)-(13, +x)-(5, +x)-(4, +x)-(1, +x)-(15, +y)-(18, -y)

La Figura 3.14 muestra la salida del algoritmo considerando los cambios de herramientas. Esta salida presenta tres reorientaciones y un cambio de herramienta. En el reporte para este caso de estudio considerando los cambios de herramientas se reflejó una salida con igual cantidad total de reorientaciones y cambios de herramientas que la salida principal propuesta por el algoritmo. La Tabla 3.9 muestra dicho resultado. Vale destacar que la hormiga 1 brinda una secuencia perfectamente elegible. La salida mostrada por el algoritmo pertenece a la secuencia adicional del caso de estudio.

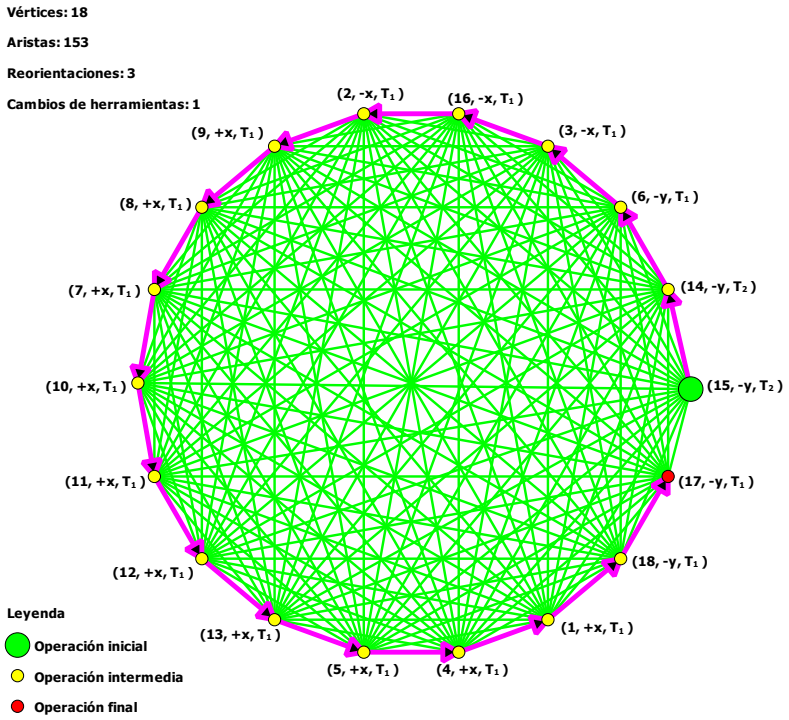


Figura 3.14: Salida del algoritmo para la Sección 1 del cuerpo principal de la cabria con herramientas

Tabla 3.9: Secuencias de la Sección 1 con igual cantidad total de reorientaciones y cambios de herramientas

Hormiga	Reorientaciones	Cambios de herramientas	Salida
1	3	1	(15, -y, T_2)-(14, -y, T_2)-(6, -y, T_1)-(3, -x, T_1)-(9, -x, T_1)-(16, -x, T_1)-(2, -x, T_1)-(17, -y, T_1)-(18, -y, T_1)-(11, +x, T_1)-(12, +x, T_1)-(13, +x, T_1)-(10, +x, T_1)-(8, +x, T_1)-(7, +x, T_1)-(5, +x, T_1)-(4, +x, T_1)-(1, +x, T_1)

Por otro lado las herramientas utilizadas para el ensamble de la Sección 1 del cuerpo principal de la cabria se muestran en la Tabla 3.10.

Tabla 3.10: Herramientas para el ensamble de cada componente de la Sección 1

Número	Nombre del componente	Herramienta
1	Viga 1	T_1
2	Viga 2	T_1
3	Viga 3	T_1
4	Viga 4	T_1
5	Viga 5	T_1
6	Viga 6	T_1
7	Viga L	T_1
8	Viga L	T_1
9	Viga 7	T_1
10	Viga 8	T_1
11	Soporte de rotación	T_1
12	Viga L	T_1
13	Viga L	T_1
14	Polea	T_2
15	Polea con soporte	T_2
16	Viga 9	T_1
17	Oreja	T_1
18	Oreja	T_1

3.3.2. Marco tipo A

La Figura 3.15 muestra el marco tipo A de un taladro de perforación de petróleo de 2000 HP. Este marco cumple la función de propiciar la elevación de la cabria de conjunto con los elementos del dispositivo de izamiento. Además, sirve de sostén y asegura la correcta ubicación vertical de la cabria. El modelo cuenta con un total de 19 componentes en los 3 ejes principales. La matriz de desensamble de la Figura 3.15 se muestra en la Sección de fichero 6.

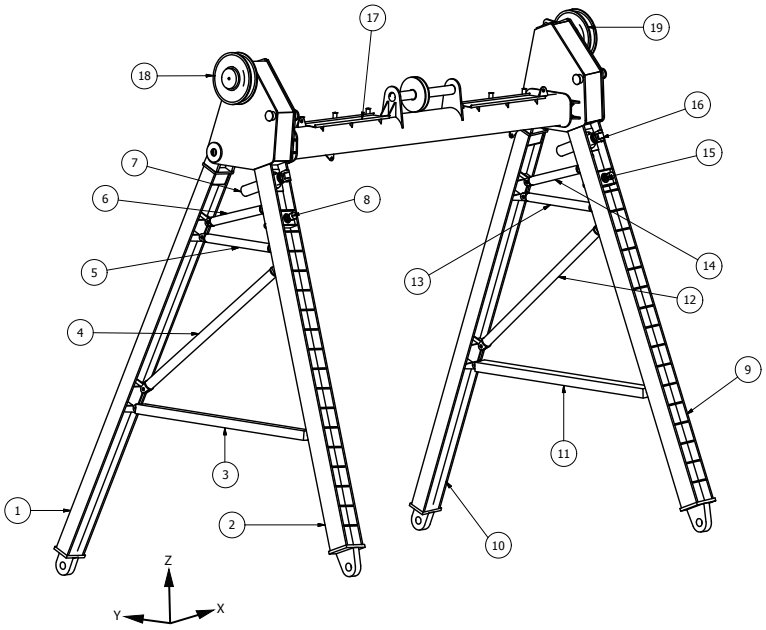


Figura 3.15: Marco tipo A de un taladro de perforación de petróleo de 2000 HP

Sección de fichero 6: Marco tipo A

DISASSEMBLY_MATRIX_SECTION

000	101	100	100	100	100	000	000	000	000	000	000	000	000	000	000	000	001	000
110	000	110	110	110	110	101	101	100	100	100	100	100	100	000	000	101	011	100
111	101	000	001	001	001	001	001	100	100	100	000	000	000	000	000	000	001	000
111	101	000	001	001	001	001	001	100	100	000	100	000	000	000	000	000	001	000
111	101	000	000	000	001	001	001	100	100	000	000	100	000	000	000	000	001	000
111	101	000	000	000	000	000	000	100	100	000	000	000	100	000	000	000	001	000
010	111	000	000	000	000	000	000	100	000	000	000	000	000	000	100	000	001	000
010	111	000	000	000	011	000	000	100	000	000	000	000	100	100	000	111	000	001
000	000	000	000	000	000	000	000	000	110	110	110	110	110	101	101	101	000	111
000	000	000	000	000	000	000	000	101	000	100	100	100	100	000	000	000	000	001
000	000	000	000	000	000	000	000	101	111	000	001	001	001	001	001	000	000	001
000	000	000	000	000	000	000	000	101	111	000	000	001	001	001	001	000	000	001
000	000	000	000	000	000	000	000	101	111	000	000	000	001	001	001	000	000	001
000	000	000	000	000	000	000	000	101	111	000	000	000	000	000	001	000	000	001
000	000	000	000	000	000	000	000	111	010	000	000	000	001	000	001	000	000	000
000	000	000	000	000	000	000	000	111	010	000	000	000	000	000	000	000	000	001
000	001	000	000	000	000	000	000	101	000	000	000	000	000	000	000	000	000	000
000	111	000	000	000	000	000	000	100	000	000	000	000	000	000	000	000	000	100
000	000	000	000	000	000	000	000	011	000	000	000	000	000	000	000	000	000	000

La Figura 3.16 muestra la salida del algoritmo sin cambios de herramientas. Esta salida presenta dos reorientaciones.

En el reporte para este caso de estudio sin considerar los cambios de herramientas se reflejaron dos salidas con igual cantidad de reorientaciones que la salida principal propuesta por el algoritmo. La Tabla 3.11 muestra dichos resultados. Vale destacar que ambas secuencias carecen de sentido práctico. La salida mostrada por el algoritmo pertenece a la secuencia adicional del caso de estudio.

Tabla 3.11: Secuencias con igual cantidad de reorientaciones para el Marco tipo A

Hormiga	Salida
1	(19, +x)-(9, +x)-(2, +x)-(18, +x)-(16, +y)-(15, +y)-(8, +y)-(7, +y)-(12, -y)-(13, -y)-(14, -y)-(11, -y)-(17, -y)-(6, -y)-(5, -y)-(4, -y)-(10, -y)-(3, -y)-(1, -y)
9	(19, +x)-(9, +x)-(2, +x)-(18, +x)-(16, +y)-(15, +y)-(8, +y)-(7, +y)-(12, -y)-(13, -y)-(14, -y)-(11, -y)-(6, -y)-(5, -y)-(4, -y)-(10, -y)-(3, -y)-(1, -y)-(17, -y)

La Figura 3.17 muestra la salida del algoritmo considerando los cambios de herramientas. Esta salida presenta dos reorientaciones y cuatro cambios de herramientas. El reporte Marco tipo A reflejó una salida con igual cantidad total de reorientaciones y cambios de herramientas que la salida principal propuesta por el algoritmo. La salida mostrada por el algoritmo pertenece a la secuencia adicional para el caso de estudio. Las herramientas utilizadas para el ensamble del marco tipo A se muestran en la Tabla 3.12.

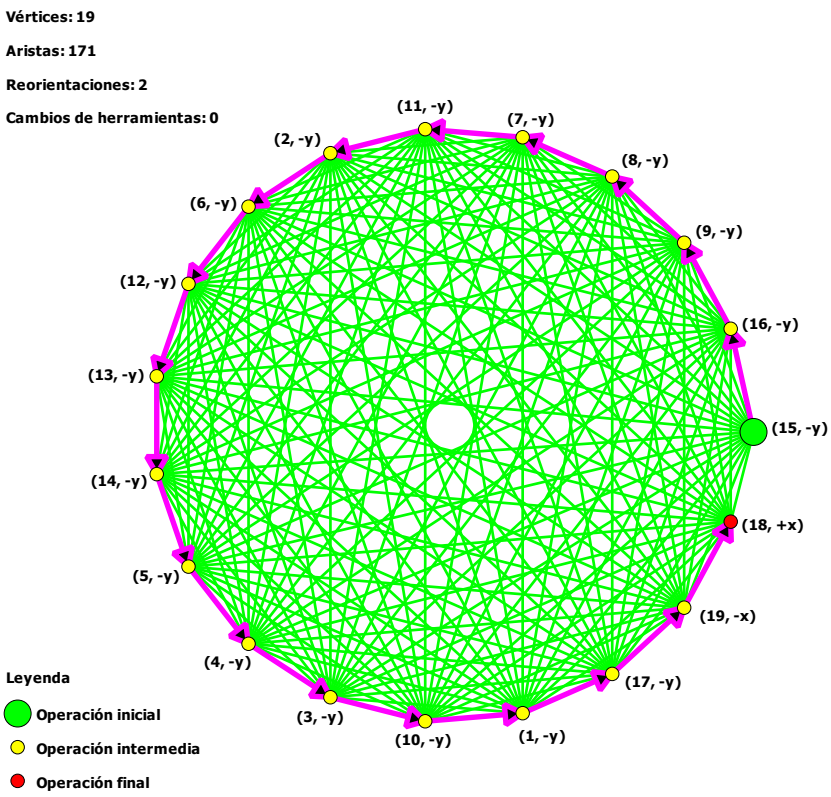


Figura 3.16: Salida del algoritmo para el Marco tipo A

Tabla 3.12: Herramientas para el ensamble de los componentes del Marco tipo A

Número	Nombre del componente	Herramienta
1	Pata trasera izquierda	T_1
2	Pata delantera izquierda	T_1
3	Tirante	T_2
4	Tirante	T_2
5	Tirante	T_2
6	Tirante	T_2
7	Gato izquierdo	T_3
8	Tornillo	T_3
9	Pata delantera derecha	T_1
10	Pata trasera derecha	T_1
11	Tirante	T_2
12	Tirante	T_2
13	Tirante	T_2
14	Tirante	T_2
15	Tornillo	T_3
16	Gato derecho	T_3
17	Tirante principal	T_1
18	Polea izquierda	T_4
19	Polea derecha	T_4

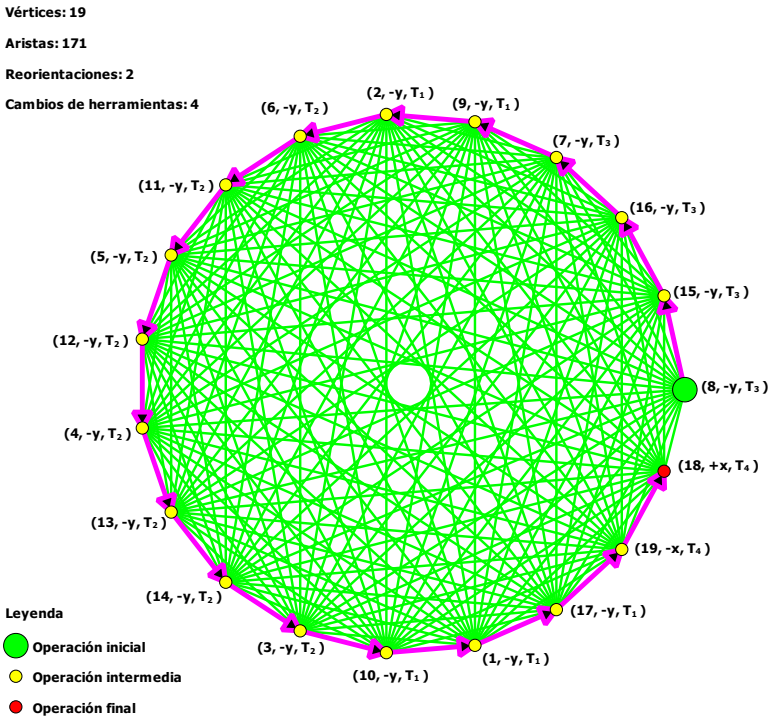
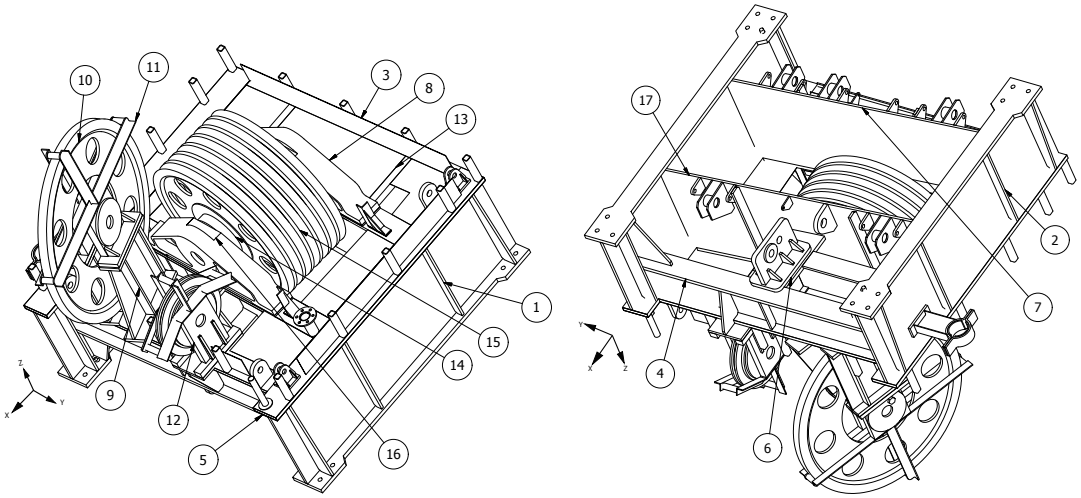


Figura 3.17: Salida del algoritmo para el Marco tipo A considerando los cambios de herramientas

3.3.3. Bloque corona

La Figura 3.18 muestra el Bloque corona de un taladro de perforación de petróleo de 2000 HP.



(a) Vista frontal del Bloque corona (b) Vista posterior del Bloque corona

Figura 3.18: Bloque corona de un taladro de perforación de petróleo de 2000 HP

El Bloque corona esta ubicado en la parte superior de la torre y está constituido por una serie de poleas. El cable de perforación pasa a través de estas poleas y llega al Bloque viajero. El Bloque corona es quien sostiene y da movilidad al Bloque viajero. Su función es la de proporcionar los medios de soporte para suspender las herramientas (García, 2008). El modelo cuenta con un total de 17 componentes en los 3 ejes principales. La matriz de desensamble de la Figura 3.18 se muestra en la Sección de fichero 7.

Sección de fichero 7: Bloque corona

DISASSEMBLY_MATRIX_SECTION

000	000	000	101	001	000	101	000	000	000	000	000	000	000	000	000	101
010	000	010	111	001	111	111	000	000	000	000	000	010	010	010	000	111
010	000	000	100	110	100	000	100	100	100	000	100	100	100	100	100	000
110	100	000	000	011	000	000	000	001	001	001	001	000	000	000	000	000
110	100	010	000	000	110	000	001	001	001	001	001	110	111	110	001	000
010	100	000	101	001	000	000	000	001	001	001	000	110	000	000	000	010
110	100	000	100	001	100	000	001	000	000	000	000	000	000	100	000	100
000	000	000	000	010	000	000	000	100	100	000	100	000	111	100	100	000
000	000	000	000	010	110	000	000	000	010	111	010	000	000	000	000	000
000	000	000	000	010	000	000	000	010	000	111	010	000	000	000	000	000
000	000	000	000	000	000	000	000	010	110	000	000	000	000	000	000	000
000	000	000	000	010	000	000	000	000	000	000	000	000	000	000	000	000
000	000	000	000	111	110	000	000	000	000	000	000	000	000	000	000	000
010	000	000	100	110	100	000	011	100	100	000	100	010	000	011	111	000
010	000	000	100	110	100	000	000	100	100	100	100	110	011	000	100	100
000	000	000	000	010	000	000	000	100	101	001	100	000	011	000	000	000
110	100	000	100	000	100	000	000	000	000	000	000	000	000	000	001	000

La Figura 3.19 muestra la salida del algoritmo sin cambios de herramientas. Esta salida presenta dos reorientaciones. En el reporte Bloque corona para este caso de estudio sin considerar los cambios de herramientas se reflejaron siete salidas con igual cantidad de reorientaciones que la salida principal propuesta por el algoritmo. Vale destacar que las secuencias brindadas por las hormigas 10, 8 y 7 reportan secuencias con mayor sentido práctico que la secuencia propuesta por el algoritmo. Las secuencias brindadas por las hormigas 1, 4, 6 y 11 carecen de sentido práctico. La salida mostrada por el algoritmo pertenece a la secuencia premiada del caso de estudio.

La Figura 3.20 muestra la salida del algoritmo considerando los cambios de herramientas. Esta salida presenta tres reorientaciones y cuatro cambios de herramientas. En el reporte para este caso de estudio considerando los cambios de herramientas se reflejó una salida con igual cantidad total de reorientaciones y cambios de herramientas que la salida principal propuesta por el algoritmo. La Tabla 3.13 muestra dicho resultado. Vale destacar que la hormiga 8 brinda una secuencia con menor sentido práctico que el presentado en la salida del algoritmo. Esta última pertenece a la secuencia adicional del caso de estudio. Por otro lado las herramientas utilizadas para el ensamble del Bloque corona se muestran en la Tabla 3.14.

Para realizar el recambio de partes defectuosas de un producto se toma la planificación de secuencia de ensamble de este y se desensamblan todos los componentes que se encuentran después de la parte a cambiar. Vale destacar que el orden de los componentes a desensamblar es el inverso de la secuencia de ensamble planificada para ese producto. Ver Figura 3.2 y Figura 3.21.

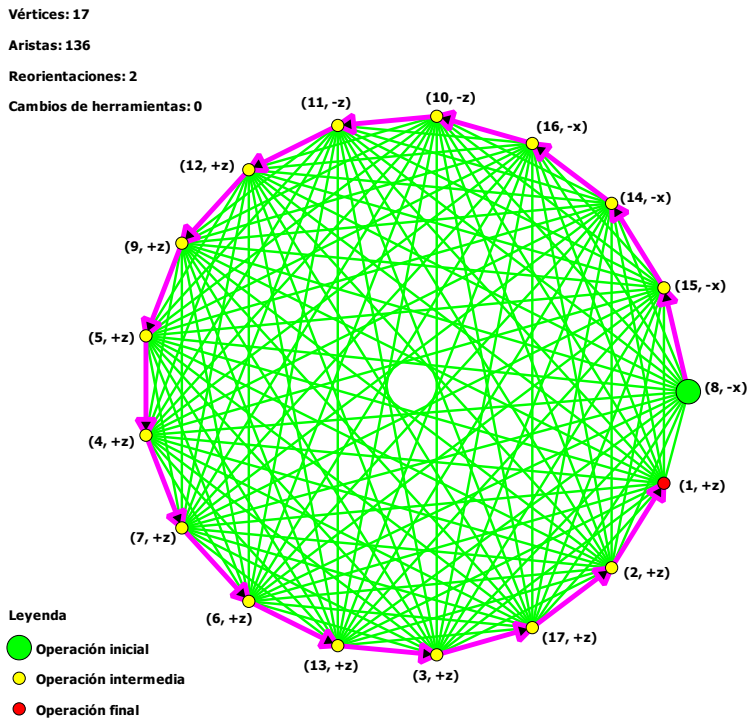


Figura 3.19: Salida del algoritmo para el Bloque corona de un taladro de perforación de petróleo de 2000 HP

Tabla 3.13: Secuencias con igual cantidad total de reorientaciones y cambios de herramientas para el Bloque corona

Hormiga	Reorientaciones	Cambios de herramientas	Salida
8	3	4	(15, -x, T_4)-(14, -x, T_4)-(16, -x, T_2)-(8, +x, T_2)-(5, +z, T_3)-(13, +z, T_3)-(17, +z, T_1)-(7, +z, T_1)-(4, +z, T_1)-(6, +z, T_1)-(2, +z, T_1)-(1, +z, T_1)-(9, -z, T_1)-(3, -z, T_1)-(12, -z, T_5)-(10, -z, T_5)-(11, -z, T_5)

La Tabla 3.15 muestra la reducción de la memoria necesaria para representar los casos de estudio tratados en este trabajo. Esta reducción es posible mediante la inferencia de conocimiento de la MD descrita en la Sección §2.3.

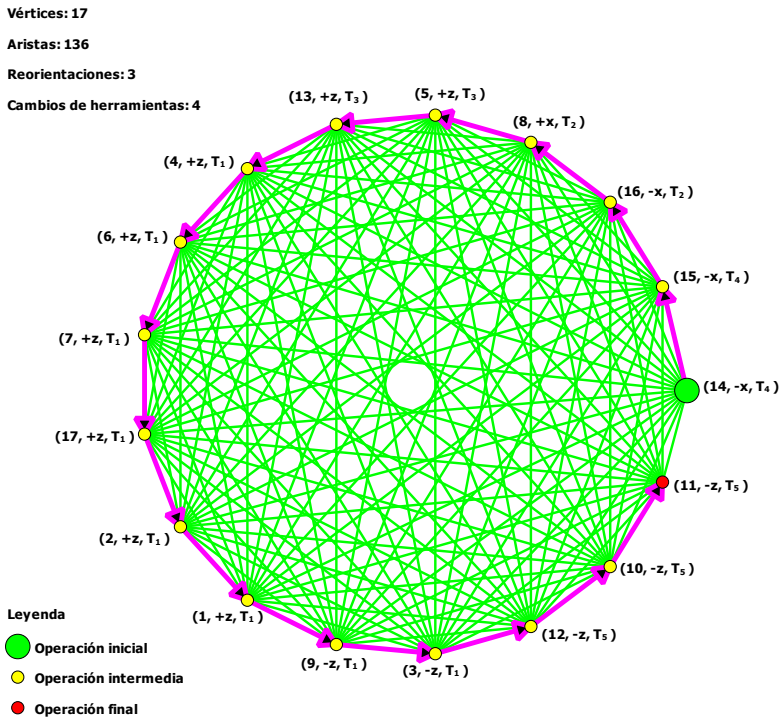


Figura 3.20: Salida del algoritmo para el Bloque corona considerando los cambios de herramientas

Tabla 3.14: Herramientas para el ensamble de los componentes del Bloque corona

Número	Nombre del componente	Herramienta
1	Viga izquierda	T_1
2	Viga derecha	T_1
3	Cierre extremo 1	T_1
4	Cierre extremo 2	T_1
5	Viga tranque 1	T_3
6	Viga tranque 2	T_1
7	Viga tranque 3	T_1
8	Chumacera	T_2
9	Pedestal polea grande	T_1
10	Polea grande	T_5
11	Límite polea grande	T_5
12	Calzo polea guía tubo	T_5
13	Piso	T_3
14	Eje de poleas	T_4
15	Poleas polipasto	T_4
16	Chumacera	T_2
17	Viga tranque 4	T_1

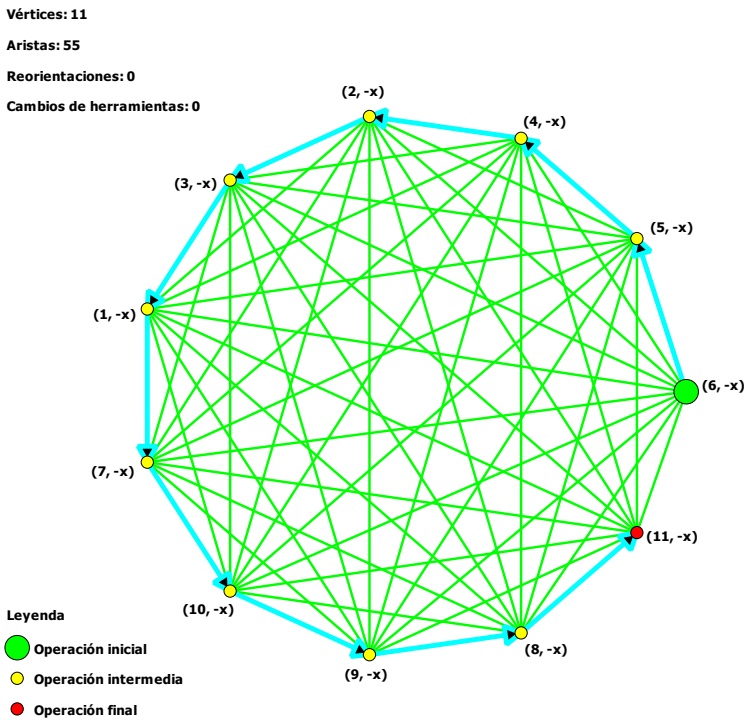


Figura 3.21: Secuencia de desensamble para el caso de estudio Ensamble industrial

Tabla 3.15: Reducción de memoria para representar los casos de estudio

			Inicial	Final	
Casos de estudio	No. componentes	No. ejes	No. nodos	No. nodos	No. aristas
Ensamble industrial	11	2	44	11	55
Controlador industrial	16	3	96	16	120
Sistema de poleas	16	3	96	16	120
Generador	15	3	90	15	105
Sección 1 de la cabria	18	3	108	18	153
Marco tipo A	19	3	114	19	171
Bloque corona	17	3	102	17	136

Tabla 3.16: Iteraciones y tiempo de corrida de los casos de estudio

Casos de estudio	Cantidad de iteraciones	Tiempo de ejecución (ms)
Ensamble industrial	325	1264
Controlador industrial	362	4196
Sistema de poleas	524	5148
Sistema de poleas con herramientas	361	3494
Generador	356	3198
Generador con herramientas	356	3120
Sección 1 del cuerpo principal de la cabria	546	13822
Marco tipo A	404	10608
Bloque corona	369	7144

Conclusiones

Con la realización de la presente investigación se arribaron a las siguientes conclusiones:

- El algoritmo $\mathcal{MM}$AS resulta factible para la solución del problema ASP.
- Con el algoritmo $\mathcal{MM}$AS es posible considerar los cambios de herramientas para la resolución del problema ASP. Estos cambios deben considerarse como se muestra en la regla probabilística 2.4.2.
- La interpretación dada a la variable L^{best}, para resolver el problema ASP, resulta factible pues permite contemplar de manera conjunta las reorientaciones y los cambios de herramientas.
- El reporte que genera el módulo de planificación de secuencias de ensamble reduce los análisis para un mismo producto por lo que se puede crear una base de conocimientos.
- El sentido práctico de las secuencias obtenidas con el algoritmo propuesto puede mejorar cuando se tienen en cuenta los cambios de herramientas. Esto se puede constatar en los resultados de los casos de estudio: Sistema de poleas, Generador y Bloque corona.
- Se obtuvo un módulo que minimiza la cantidad de reorientaciones y los cambios de herramientas durante el proceso de ensamble.

Con el objetivo de obtener un proceso de planificación de secuencia más eficiente se recomienda:

- Integrar el módulo de planificación de secuencias de ensamble al sistema Galba-CAD del proyecto CDSEM.
- Generar la matriz de desensamble a partir de las restricciones presentes en un modelo de ensamble CAD.
- Contemplar en el algoritmo propuesto las interferencias de las herramientas y optimizar los puestos de trabajo involucrados.

Acrónimos

ACO Optimización basada en Colonia de Hormigas (por sus siglas en inglés, Ant Colony Optimization)

ACS Sistema de Colonia de Hormigas (por sus siglas en inglés, Ant Colony System)

AG Algoritmos Genéticos

ASP Planificación de Secuencias de Ensamble (por sus siglas en inglés, Assembly Sequence Planning)

ASrank Sistema de Hormigas basado en el rango (por sus siglas en inglés, Rank-Based Ant System)

AS Sistema de Hormigas (por sus siglas en inglés, Ant System)

CAD Diseño Asistido por Computadoras (por sus siglas en inglés, Computer Aided Design)

CAE Ingeniería Asistida por Computadoras (por sus siglas en inglés, Computer Aided Engineering)

CDSEM Centro de Diseño y Simulación de Estructuras Mecánicas

EAS Sistema de Hormigas Elitista (por sus siglas en inglés, System Ant Elitist)

GID Gerencia de Ingeniería y Diseño

ICVT Industria China Venezolana de Taladros, S.A

MD Matriz de Desensamble

$\mathcal{MM}$**AS** Sistema de Hormigas $\mathcal{MAX}$-$\mathcal{MIN}$ (por sus siglas en inglés, $\mathcal{MAX}$-$\mathcal{MIN}$ Ant System)

OD Operación de desensamble

PSO Optimización por Enjambre de Partículas (por sus siglas en inglés, Particle Swarm Optimisation)

TSP Problema del Agente Viajero (por sus siglas en inglés, Travelling Salesman Problem)

UCI Universidad de las Ciencias Informáticas

Referencias bibliográficas

Ahmed, Z. H. (2013). An experimental study of a hybrid genetic algorithm for the maximum traveling salesman problem. *Mathematical Sciences*, *7*(1), 10.

Alonso, S., Cordón, O., Fernández, I & Herrera, F. (2004). La metaheurística de optimización basada en colonias de hormigas: modelos y nuevos enfoques. *Optimización inteligente: técnicas de inteligencia computacional para optimización*, 261-314.

Autodesk, I. (2013). Autodesk inventor. http://usa.autodesk.com/autodesk-inventor. [Consulta: 22.06.13].

Ben-Arieh, D & Kramer, B. (1994). Computer-aided process planning for assembly: generation of assembly operations sequence. *The international journal of production research*, *32*(3), 643-656.

Boothroyd, G. (1994). Product design for manufacture and assembly. *Computer-Aided Design*, *26*(7), 505-520.

Ch, S. & Mathur, S. (2010). Modeling uncertainty analysis in flow and solute transport model using adaptive neuro fuzzy inference system and particle swarm optimization. *Civil Engineering*, *14*, 941-951.

Chang, C. C., Tseng, H. E. & Meng, L. P. (2009). Artificial immune systems for assembly sequence planning exploration. *Engineering Applications of Artificial Intelligence*, *22*(8), 1218-1232.

Chen, S. F. & Liu, Y. J. (2001). An adaptive genetic assembly-sequence planner. *International Journal of Computer Integrated Manufacturing*, *14*(5), 489-500.

Chen, W. C., Hsu, Y. Y., Hsieh, L. F. & Tai, P. H. (2010). A systematic optimization approach for assembly sequence planning using taguchi method, doe, and bpnn. *Expert Systems with Applications*, *37*(1), 716-726.

De Lit, P., Latinne, P., Rekiek, B. & Delchambre, A. (2001). Assembly planning with an ordering genetic algorithm. *International Journal of Production Research*, *39*(16), 3623-3640.

Dini, G & Santochi, M. (1992). Automated sequencing and subassembly detection in assembly planning. *CIRP Annals-Manufacturing Technology*, *41*(1), 1-4.

Dorigo, M. & Blum, C. (2005). Ant colony optimization theory: a survey. *Theoretical computer science*, *344*(2), 243-278.

Dorigo, M. & Stützle, T. (2004). *Ant colony optimization*. The MIT Press.

Dorigo, M., Birattari, M. & Stutzle, T. (2006). Ant colony and optimization and artificial ants and as a computational and intelligence technique. (Vol. 1, *4*, pp. 28-39). IEEE.

Dorzán, M, Gagliardi, E, Leguizamón, M, Taranilla, M & Hernández-Penalver, G. (2009). Algoritmos aco aplicados a problemas geométricos de optimización. En *Xiii encuentros de geometría computacional* (pp. 183-190).

Gálvez, D. L. (1998). *Algoritmos genéticos*.

García, A. (2008). *El taladro y sus componentes*. Centro Internacional de Educación y Desarrollo.

Homem de Mello, L. & Sanderson, A. (1991). A correct and complete algorithm for the generation of mechanical assembly sequences. *Robotics and Automation, IEEE Transactions on*, *7*(2), 228-240.

Hsu, Y. Y., Tai, P. H., Wang, M. W. & Chen, W. C. (2011). A knowledge-based engineering system for assembly sequence planning. *The International Journal of Advanced Manufacturing Technology*, *55*, 763-782.

Huang, Y. & Lee, C. (1991). A framework of knowledge-based assembly planning. En *Robotics and automation, 1991. proceedings., 1991 ieee international conference on* (pp. 599-604). IEEE.

Hui, C., Yuan, L. & Kai-fu, Z. (2009). Efficient method of assembly sequence planning based on gaaa and optimizing by assembly path feedback for complex product. *The International Journal of Advanced Manufacturing Technology*, *42*(11), 1187-1204.

Jiménez, P. (2013). Survey on assembly sequencing: a combinatorial and geometrical perspective. *Journal of Intelligent Manufacturing*, *24*(2), 235-250.

Kennedy, J. & Eberhart, R. (1995). Particle swarm optimization. En *Neural networks, 1995. proceedings., ieee international conference on* (Vol. 4, pp. 1942-1948). IEEE.

Ko, H. & Lee, K. (1987). Automatic assembling procedure generation from mating conditions. *Computer-Aided Design*, *19*(1), 3-10.

Kumar, V. M., Murthy, A. & Chandrashekara, K. (2012). A hybrid algorithm optimization approach for machine loading problem in flexible manufacturing system. *Journal of Industrial Engineering International*, *8*(1), 1-10.

Lendak, I., Erdeljan, A., Čapko, D. & Vukmirovic, S. (2010). Algorithms in electric power system one-line diagram creation. En *Systems man and cybernetics (smc), 2010 ieee international conference on* (pp. 2867-2873).

Li, J., Khoo, L. & Tor, S. (2003). A tabu-enhanced genetic algorithm approach for assembly process planning. *Journal of Intelligent Manufacturing*, *14*(2), 197-208.

Lin, Y., Che, Z., Chiang, T.-A., Che, Z.-G. & Chiang, C. (2009). A bi-objective model for concurrent planning of supplier selection and assembly sequence planning. En *Global perspective for competitive enterprise, economy and ecology* (pp. 573-580). Springer.

Lv, H. G., Lu, C. & Zha, J. (2010). A hybrid dpso-sa approach to assembly sequence planning. En *Mechatronics and automation (icma), 2010 international conference on* (pp. 1998-2003). IEEE.

Lv, H. & Lu, C. (2010). An assembly sequence planning approach with a discrete particle swarm optimization algorithm. *The International Journal of Advanced Manufacturing Technology*, *50*(5), 761-770.

Maniezzo, V., Gambardella, L. & De Luigi, F. (2004). Ant colony optimization. En G. Onwubolu & B. Babu (Eds.), *New optimization techniques in engineering* (Cap. 5, pp. 101-117). Springer-Verlag Berlin Heidelberg.

Marian, R., Luong, L. & Abhary, K. (2003). *Optimisation of assembly sequences using genetic algorithms*. (Tesis doctoral, University of South Australia).

Matai, R., Singh, S & Mittal, M. (2010). Traveling salesman problem: an overview of applications, formulations, and solution approaches. En D. Davendra (Ed.), *Traveling salesman problem, theory and applications* (Cap. 1, pp. 1-25). InTech.

Meier, R., Brandt, F., Pisabarro, T. M., Baldauf, C. & Sippl, W. (2008). A new approach for flexible protein-ligand docking based on particle swarm optimisation. *Chemistry Central Journal*, *2*, 1-1.

Moon, I., Logendran, R. & Lee, J. (2009). Integrated assembly line balancing with resource restrictions. *International Journal of Production Research*, *47*(19), 5525-5541.

Ou, L. M. & Xu, X. (2013). Relationship matrix based automatic assembly sequence generation from a cad model. *Computer-Aided Design*, *45*(7), 1053-1067.

Pan, C. (2005). *Integrating cad files and automatic assembly sequence planning*. Iowa State University. Recuperado desde http://books.google.com.cu/books?id=EyWDNwAACAAJ

PTC, I. (2013). Creo parametric. http://www.ptc.com/product/creo/parametric. [Consulta: 22.06.13].

Qin, L., Pan, Y., Chen, L. & Chen, Y. (2006). An improved ant colony algorithm with diversified solutions based on the immune strategy. *BMC bioinformatics*, *7*(Suppl 4), S3.

Rashid, M., Hutabarat, W. & Tiwari, A. (2012). A review on assembly sequence planning and assembly line balancing optimisation using soft computing approaches. *The International Journal of Advanced Manufacturing Technology*, *59*(1), 335-349.

Shan, H., Li, S., Gong, D. & Lou, P. (2006). Genetic simulated annealing algorithm-based assembly sequence planning. En *Technology and innovation conference, 2006. itic 2006. international*.

Shan, H., Zhou, S. & Sun, Z. (2009). Research on assembly sequence planning based on genetic simulated annealing algorithm and ant colony optimization algorithm. *Assembly Automation*, *29*, 249-256.

Shu-xia, L. & Hong-bo, S. (2008). Gssa and aco for assembly sequence planning: a comparative study. En *Automation and logistics, 2008. ical 2008. ieee international conference on*.

Sivanandam, S. & Deepa, S. (2008). *Introduction to genetic algorithms*. Springer-Verlag Berlin Heidelberg.

Socha, K., Knowles, J. & Sampels, M. (2002). A max-min ant system for the university course timetabling problem. *Ant algorithms*, *2463*, 1-13.

Sorin, I. & Costin, B. (2013). Multi-agent approach to distributed ant colony optimization. *Science of Computer Programming*, *78*(6), 762 -774.

SPLMS, I. (2013a). Nx. http://www.plm.automation.siemens.com/en_us/products/nx. [Consulta: 22.06.13].

SPLMS, I. (2013b). Solid edge. http://www.plm.automation.siemens.com/en_us/products/velocity/solidedge. [Consulta: 22.06.13].

Stützle, T. & Hoos, H. (2000). Max-min ant system. *Future generation computer systems*, *16*(8), 889-914.

Systèmes, D. (2013a). Catia. http://www.3ds.com/products/catia. [Consulta: 22.06.13].

Systèmes, D. (2013b). Solidworks. http://www.solidworks.com/sw/products/3d-cad/packages.htm. [Consulta: 22.06.13].

Tomás García, L. (2009). Planificación y optimización asistida por computadora de secuencias de ensamble mecánico. *Ingeniería Mecánica*, *12*(1), 59-68.

Wang, J., Liu, J. & Zhong, Y. (2005). A novel ant colony algorithm for assembly sequence planning. *The International Journal of Advanced Manufacturing Technology*, *25*, 1137-1143.

Wang, Y & Liu, J. (2010). Chaotic particle swarm optimization for assembly sequence planning. *Robotics and Computer-Integrated Manufacturing*, *26*(2), 212-222.

Wu, M. T., Hong, T. P. & Lee, C. N. (2012). A continuous ant colony system framework for fuzzy data mining. *Soft Computing-A Fusion of Foundations, Methodologies and Applications*, *16*, 1-12.

Yousefikhoshbakht, M., Khorram, E. & Hamedan, I. (2012). Solving the vehicle routing problem by a hybrid meta-heuristic algorithm. *Journal of Industrial Engineering International*, *8*, 11.

Yu, J. & Yin, Y. (2010). Assembly line balancing based on an adaptive genetic algorithm. *The International Journal of Advanced Manufacturing Technology*, *48*(1), 347-354.

Yu, J. & Wang, C. (2013). A max–min ant colony system for assembly sequence planning. *The International Journal of Advanced Manufacturing Technology*, *10.1007/s00170-012-4695-x*, 1-17. doi:10.1007/s00170-012-4695-x

Zhang, J., Sun, J. & He, Q. (2010). An approach to assembly sequence planning using ant colony optimization. En *Intelligent control and information processing (icicip), 2010 international conference on* (pp. 230-233). IEEE.

Zhou, W., Zheng, J.-r., Yan, J.-j. & Wang, J.-f. (2011). A novel hybrid algorithm for assembly sequence planning combining bacterial chemotaxis with genetic algorithm. *The International Journal of Advanced Manufacturing Technology*, *52*(5-8), 715-724.

Generado con LaTeX: 22 de noviembre de 2013

Apéndices

REPORTE DEL PLANIFICADOR DE SECUENCIA DE ENSAMBLE

Ing. Maybel Díaz Capote
Universidad de las Ciencias Informáticas, La Habana, Cuba
Fecha: sábado 5 de octubre de 2013

DETALLES DEL REPORTE:

Caso de estudio: Ensamble industrial
Reorientaciones: sí
Cambios de herramientas: no
Vértices: 11
Aristas: 55
Número total de hormigas: 4

Secuencias de ensamble:

Hormiga: 1
Reorientaciones: 0

Salida:
(11, +x)-(8, +x)-(9, +x)-(7, +x)-(10, +x)-(1, +x)-(3, +x)-(2, +x)-(4, +x)-(5, +x)-(6, +x)

Hormiga: 2
Reorientaciones: 1

Salida:
(5, -x)-(4, -x)-(2, -x)-(3, -x)-(1, -x)-(7, -x)-(10, -x)-(9, -x)-(8, -x)-(11, -x)-(6, +y)

Hormiga: 3
Reorientaciones: 1

Salida:
(5, -x)-(4, -x)-(2, -x)-(3, -x)-(1, -x)-(7, -x)-(10, -x)-(9, -x)-(8, -x)-(11, -x)-(6, -y)

Hormiga: 4
Reorientaciones: 0

Salida:
(6, -x)-(5, -x)-(4, -x)-(2, -x)-(3, -x)-(1, -x)-(7, -x)-(10, -x)-(9, -x)-(8, -x)-(11, -x)

Secuencia de salida premiada
Reorientaciones: 0

Salida:
(11, +x)-(8, +x)-(9, +x)-(10, +x)-(7, +x)-(1, +x)-(3, +x)-(2, +x)-(4, +x)-(5, +x)-(6, +x)

REPORTE DEL PLANIFICADOR DE SECUENCIA DE ENSAMBLE

Ing. Maybel Díaz Capote
Universidad de las Ciencias Informáticas, La Habana, Cuba
Fecha: sábado 5 de octubre de 2013

DETALLES DEL REPORTE:

Caso de estudio: Controlador industrial
Reorientaciones: sí
Cambios de herramientas: no
Vértices: 16
Aristas: 120
Número total de hormigas: 5

Secuencias de ensamble:

Hormiga: 1
Reorientaciones: 3

Salida:
(14, +z)-(15, +z)-(6, +z)-(16, +z)-(8, +z)-(5, +z)-(9, +z)-(4, +z)-(3, +z)-(2, +z)-(1, +z)-(12, -y)-(10, +z)-(13, -x)-(11, -x)-(7, -x)

Hormiga: 2
Reorientaciones: 5

Salida:
(14, +z)-(15, +z)-(6, +z)-(7, -x)-(16, +z)-(8, +z)-(5, +z)-(9, +z)-(4, +z)-(3, +z)-(2, +z)-(1, +z)-(12, -y)-(10, +z)-(13, -x)-(11, -x)

Hormiga: 3
Reorientaciones: 3

Salida:
(14, +z)-(15, +z)-(6, +z)-(16, +z)-(8, +z)-(5, +z)-(9, +z)-(4, +z)-(3, +z)-(2, +z)-(1, +z)-(12, -y)-(10, +z)-(11, -x)-(7, -x)-(13, -x)

Hormiga: 4
Reorientaciones: 4

Salida:
(6, +z)-(16, +z)-(8, +z)-(5, +z)-(9, +z)-(4, +z)-(3, +z)-(2, +z)-(1, +z)-(12, -y)-(10, +z)-(13, -x)-(11, -x)-(7, -x)-(15, -z)-(14, -z)

Hormiga: 5
Reorientaciones: 4

Salida:
(6, +z)-(16, +z)-(8, +z)-(5, +z)-(9, +z)-(4, +z)-(3, +z)-(2, +z)-(1, +z)-(12, -y)-(10, +z)-(13, -x)-(11, -x)-(7, -x)-(14, -z)-(15, -z)

Secuencia de salida premiada
Reorientaciones: 3

Salida:
(1, -y)-(12, -y)-(10, +z)-(5, -z)-(4, -z)-(8, -z)-(3, -z)-(9, -z)-(16, -z)-(2, -z)-(6, -z)-(14, -z)-(15, -z)-(13, -x)-(11, -x)-(7, -x)

REPORTE DEL PLANIFICADOR DE SECUENCIA DE ENSAMBLE

Ing. Maybel Díaz Capote
Universidad de las Ciencias Informáticas, La Habana, Cuba
Fecha: sábado 5 de octubre de 2013

DETALLES DEL REPORTE:

Caso de estudio: Sistema de poleas
Reorientaciones: sí
Cambios de herramientas: no
Vértices: 16
Aristas: 120
Número total de hormigas: 3

Secuencias de ensamble:

Hormiga: 1
Reorientaciones: 3

Salida:
(15, +x)-(8, +x)-(5, -z)-(13, -z)-(10, -z)-(9, -z)-(16, -z)-(3, -x)-(7, -x)-(6, -x)-(12, -x)-(4, +x)-(11, +x)-(14, +x)-(2, +x)-(1, +x)

Hormiga: 2
Reorientaciones: 4

Salida:
(15, +x)-(8, +x)-(5, -z)-(13, -z)-(10, -z)-(9, -z)-(4, +x)-(11, +x)-(2, +x)-(1, +x)-(14, +x)-(16, -z)-(3, -x)-(7, -x)-(6, -x)-(12, -x)

Hormiga: 3
Reorientaciones: 3

Salida:
(15, +x)-(8, +x)-(5, -z)-(13, -z)-(10, -z)-(9, -z)-(16, -z)-(3, -x)-(7, -x)-(6, -x)-(12, -x)-(4, +x)-(11, +x)-(2, +x)-(1, +x)-(14, +x)

Secuencia de salida premiada
Reorientaciones: 3

Salida:
(15, +x)-(8, +x)-(5, -z)-(13, -z)-(10, -z)-(9, -z)-(16, -z)-(14, +x)-(2, +x)-(1, +x)-(11, -x)-(4, -x)-(3, -x)-(7, -x)-(6, -x)-(12, -x)

REPORTE DEL PLANIFICADOR DE SECUENCIA DE ENSAMBLE

Ing. Maybel Díaz Capote
Universidad de las Ciencias Informáticas, La Habana, Cuba
Fecha: sábado 5 de octubre de 2013

DETALLES DEL REPORTE:

Caso de estudio: Sistema de poleas
Reorientaciones: sí
Cambios de herramientas: sí
Vértices: 16
Aristas: 120
Número total de hormigas: 3

Secuencias de ensamble:

Hormiga: 1
Reorientaciones: 4
Cambios de herramientas: 14

Salida:
(15, +x, T_9)-(8, +x, T_3)-(5, -z, T_7)-(13, -z, T_4)-(10, -z, T_5)-(9, -z, T_6)-(7, -x, T_5)-(6, -x, T_2)-(12, -x, T_1)-(16, -z, T_1)-(3, -z, T_3)-(14, +x, T_8)-(4, +x, T_4)-(11, +x, T_5)-(2, +x, T_2)-(1, +x, T_1)

Hormiga: 2
Reorientaciones: 5
Cambios de herramientas: 12

Salida:
(15, +x, T_9)-(8, +x, T_3)-(5, -z, T_7)-(13, -z, T_4)-(10, -z, T_5)-(9, -z, T_6)-(16, -z, T_1)-(3, -z, T_3)-(14, +x, T_8)-(4, +x, T_4)-(11, +x, T_5)-(7, -x, T_5)-(6, -x, T_2)-(2, +x, T_2)-(1, +x, T_1)-(12, -x, T_1)

Hormiga: 3
Reorientaciones: 5
Cambios de herramientas: 14

Salida:
(15, +x, T_9)-(8, +x, T_3)-(5, -z, T_7)-(13, -z, T_4)-(10, -z, T_5)-(9, -z, T_6)-(7, -x, T_5)-(6, -x, T_2)-(12, -x, T_1)-(16, -z, T_1)-(3, +y, T_3)-(4, +x, T_4)-(11, +x, T_5)-(2, +x, T_2)-(1, +x, T_1)-(14, +x, T_8)

Secuencia de salida premiada
Reorientaciones: 5
Cambios de herramientas: 12

Salida:
(15, +x, T_9)-(8, +x, T_3)-(5, -z, T_7)-(13, -z, T_4)-(10, -z, T_5)-(9, -z, T_6)-(16, -z, T_1)-(4, -x, T_4)-(3, -x, T_3)-(7, -x, T_5)-(11, +x, T_5)-(14, +x, T_8)-(2, +x, T_2)-(6, -x, T_2)-(12, -x, T_1)-(1, +x, T_1)

REPORTE DEL PLANIFICADOR DE SECUENCIA DE ENSAMBLE

Ing. Maybel Díaz Capote
Universidad de las Ciencias Informáticas, La Habana, Cuba
Fecha: sábado 5 de octubre de 2013

DETALLES DEL REPORTE:

Caso de estudio: Generador
Reorientaciones: sí
Cambios de herramientas: no
Vértices: 15
Aristas: 105
Número total de hormigas: 5

Secuencias de ensamble:

Hormiga: 1
Reorientaciones: 2

Salida:
(13, +x)-(12, +x)-(11, +x)-(10, +x)-(9, +x)-(7, +x)-(6, +x)-(5, +x)-(4, +x)-(3, +x)-(2, +x)-(1, +x)-(14, -y)-(15, -y)-(8, +y)

Hormiga: 2
Reorientaciones: 2

Salida:
(13, +x)-(12, +x)-(11, +x)-(10, +x)-(9, +x)-(7, +x)-(6, +x)-(5, +x)-(4, +x)-(3, +x)-(2, +x)-(1, +x)-(14, -y)-(15, -y)-(8, -z)

Hormiga: 3
Reorientaciones: 3

Salida:
(11, +x)-(10, +x)-(9, +x)-(7, +x)-(6, +x)-(5, +x)-(4, +x)-(3, +x)-(2, +x)-(1, +x)-(14, -y)-(15, -y)-(8, +y)-(12, -x)-(13, -x)

Hormiga: 4
Reorientaciones: 4

Salida:
(11, +x)-(10, +x)-(9, +x)-(7, +x)-(6, +x)-(5, +x)-(4, +x)-(3, +x)-(8, +y)-(12, -x)-(13, -x)-(2, +x)-(1, +x)-(14, +z)-(15, +z)

Hormiga: 5
Reorientaciones: 4

Salida:
(11, +x)-(10, +x)-(9, +x)-(7, +x)-(6, +x)-(5, +x)-(4, +x)-(3, +x)-(8, +y)-(12, -x)-(13, -x)-(2, +x)-(1, +x)-(14, -y)-(15, -y)

Secuencia de salida premiada
Reorientaciones: 2

Salida:
(1, -x)-(2, -x)-(3, -x)-(4, -x)-(5, -x)-(7, -x)-(6, -x)-(9, -x)-(10, -x)-(11, -x)-(12, -x)-(13, -x)-(14, -y)-(15, -y)-(8, +y)

REPORTE DEL PLANIFICADOR DE SECUENCIA DE ENSAMBLE

Ing. Maybel Díaz Capote
Universidad de las Ciencias Informáticas, La Habana, Cuba
Fecha: sábado 5 de octubre de 2013

DETALLES DEL REPORTE:

Caso de estudio: Marco tipo A
Reorientaciones: sí
Cambios de herramientas: sí
Vértices: 19
Aristas: 171
Número total de hormigas: 11

Secuencias de ensamble:

Hormiga: 1
Reorientaciones: 5
Cambios de herramientas: 6

Salida:
(2, -x, T_1)-(9, -x, T_1)-(19, -x, T_4)-(18, +x, T_4)-(14, -y, T_2)-(10, -y, T_1)-(17, +y, T_1)-(16, +y, T_3)-(15, +y, T_3)-(8, +y, T_3)-(7, +y, T_3)-(13, +z, T_2)-(12, +z, T_2)-(11, +z, T_2)-(6, -y, T_2)-(5, -y, T_2)-(4, -y, T_2)-(3, -y, T_2)-(1, -y, T_1)

Hormiga: 2
Reorientaciones: 4
Cambios de herramientas: 7

Salida:
(18, -x, T_4)-(2, -x, T_1)-(7, +y, T_3)-(8, +y, T_3)-(15, -y, T_3)-(16, -y, T_3)-(9, -y, T_1)-(14, -y, T_2)-(6, -y, T_2)-(5, -y, T_2)-(4, -y, T_2)-(1, -y, T_1)-(10, -y, T_1)-(17, -y, T_1)-(19, -x, T_4)-(13, +z, T_2)-(12, +z, T_2)-(11, +z, T_2)-(3, +z, T_2)

Hormiga: 3
Reorientaciones: 3
Cambios de herramientas: 4

Salida:
(2, -x, T_1)-(9, -x, T_1)-(19, -x, T_4)-(18, +x, T_4)-(13, -y, T_2)-(12, -y, T_2)-(14, -y, T_2)-(11, -y, T_2)-(6, -y, T_2)-(5, -y, T_2)-(4, -y, T_2)-(3, -y, T_2)-(1, -y, T_1)-(10, -y, T_1)-(17, +y, T_1)-(15, +y, T_3)-(16, +y, T_3)-(8, +y, T_3)-(7, +y, T_3)

Hormiga: 4
Reorientaciones: 3
Cambios de herramientas: 4

Salida:
(2, -x, T_1)-(9, -x, T_1)-(19, -x, T_4)-(18, +x, T_4)-(13, -y, T_2)-(12, -y, T_2)-(14, -y, T_2)-(11, -y, T_2)-(6, -y, T_2)-(5, -y, T_2)-(4, -y, T_2)-(3, -y, T_2)-(1, -y, T_1)-(10, -y, T_1)-(17, +y, T_1)-(7, +y, T_3)-(15, +y, T_3)-(16, +y, T_3)-(8, +y, T_3)

Hormiga: 5
Reorientaciones: 3
Cambios de herramientas: 5

Salida:
(2, -x, T_1)-(9, -x, T_1)-(19, -x, T_4)-(18, +x, T_4)-(15, +y, T_3)-(8, +y, T_3)-(16, +y, T_3)-(7, +y, T_3)-(17, -y, T_1)-(13, -y, T_2)-(12, -y, T_2)-(14, -y, T_2)-(11, -y, T_2)-(6, -y, T_2)-(5, -y, T_2)-(4, -y, T_2)-(3, -y, T_2)-(1, -y, T_1)-(10, -y, T_1)

Hormiga: 6
Reorientaciones: 4

Cambios de herramientas: 7

Salida:
(18, -x, T_4)-(2, -x, T_1)-(7, +y, T_3)-(8, +y, T_3)-(15, -y, T_3)-(16, -y, T_3)-(9, -y, T_1)-(14, -y, T_2)-(6, -y, T_2)-(5, -y, T_2)-(4, -y, T_2)-(1, -y, T_1)-(10, -y, T_1)-(17, -y, T_1)-(19, -x, T_4)-(3, +z, T_2)-(13, +z, T_2)-(12, +z, T_2)-(11, +z, T_2)

Hormiga: 7
Reorientaciones: 3
Cambios de herramientas: 4

Salida:
(2, -x, T_1)-(9, -x, T_1)-(19, -x, T_4)-(18, +x, T_4)-(13, -y, T_2)-(12, -y, T_2)-(14, -y, T_2)-(11, -y, T_2)-(6, -y, T_2)-(5, -y, T_2)-(4, -y, T_2)-(3, -y, T_2)-(1, -y, T_1)-(10, -y, T_1)-(17, +y, T_1)-(8, +y, T_3)-(16, +y, T_3)-(7, +y, T_3)-(15, +y, T_3)

Hormiga: 8
Reorientaciones: 3
Cambios de herramientas: 4

Salida:
(2, -x, T_1)-(9, -x, T_1)-(19, -x, T_4)-(18, +x, T_4)-(13, -y, T_2)-(12, -y, T_2)-(14, -y, T_2)-(11, -y, T_2)-(6, -y, T_2)-(5, -y, T_2)-(4, -y, T_2)-(3, -y, T_2)-(1, -y, T_1)-(10, -y, T_1)-(17, +y, T_1)-(8, +y, T_3)-(15, +y, T_3)-(7, +y, T_3)-(16, +y, T_3)

Hormiga: 9
Reorientaciones: 3
Cambios de herramientas: 4

Salida:
(2, -x, T_1)-(9, -x, T_1)-(19, -x, T_4)-(18, +x, T_4)-(15, +y, T_3)-(8, +y, T_3)-(16, +y, T_3)-(7, +y, T_3)-(13, -y, T_2)-(12, -y, T_2)-(14, -y, T_2)-(11, -y, T_2)-(6, -y, T_2)-(5, -y, T_2)-(4, -y, T_2)-(3, -y, T_2)-(1, -y, T_1)-(10, -y, T_1)-(17, -y, T_1)

Hormiga: 10
Reorientaciones: 2
Cambios de herramientas: 4

Salida:
(7, -y, T_3)-(8, -y, T_3)-(15, -y, T_3)-(16, -y, T_3)-(9, -y, T_1)-(2, -y, T_1)-(13, -y, T_2)-(12, -y, T_2)-(14, -y, T_2)-(11, -y, T_2)-(6, -y, T_2)-(5, -y, T_2)-(4, -y, T_2)-(3, -y, T_2)-(1, -y, T_1)-(10, -y, T_1)-(17, -y, T_1)-(19, -x, T_4)-(18, +x, T_4)

Hormiga: 11
Reorientaciones: 3
Cambios de herramientas: 6

Salida:
(18, -x, T_4)-(2, -x, T_1)-(7, +y, T_3)-(8, +y, T_3)-(15, -y, T_3)-(16, -y, T_3)-(9, -y, T_1)-(13, -y, T_2)-(12, -y, T_2)-(14, -y, T_2)-(11, -y, T_2)-(6, -y, T_2)-(5, -y, T_2)-(4, -y, T_2)-(3, -y, T_2)-(1, -y, T_1)-(10, -y, T_1)-(17, -y, T_1)-(19, -x, T_4)

Secuencia de salida premiada
Reorientaciones: 2
Cambios de herramientas: 4

Salida:
(8, -y, T_3)-(15, -y, T_3)-(16, -y, T_3)-(7, -y, T_3)-(9, -y, T_1)-(2, -y, T_1)-(6, -y, T_2)-(11, -y, T_2)-(5, -y, T_2)-(12, -y, T_2)-(4, -y, T_2)-(13, -y, T_2)-(14, -y, T_2)-(3, -y, T_2)-(10, -y, T_1)-(1, -y, T_1)-(17, -y, T_1)-(19, -x, T_4)-(18, +x, T_4)

REPORTE DEL PLANIFICADOR DE SECUENCIA DE ENSAMBLE

Ing. Maybel Díaz Capote
Universidad de las Ciencias Informáticas, La Habana, Cuba
Fecha: sábado 5 de octubre de 2013

DETALLES DEL REPORTE:

Caso de estudio: Bloque Corona
Reorientaciones: sí
Cambios de herramientas: no
Vértices: 17
Aristas: 136
Número total de hormigas: 11

Secuencias de ensamble:

Hormiga: 1
Reorientaciones: 2

Salida:
(11, +z)-(10, +z)-(16, +x)-(15, +x)-(14, +x)-(8, +x)-(12, +z)-(9, +z)-(5, +z)-(7, +z)-(17, +z)-(4, +z)-(6, +z)-(3, +z)-(2, +z)-(13, +z)-(1, +z)

Hormiga: 2
Reorientaciones: 3

Salida:
(11, +z)-(10, +z)-(16, +x)-(15, +x)-(14, +x)-(8, +x)-(12, +z)-(9, +z)-(5, +z)-(4, +z)-(6, +z)-(3, +z)-(2, +z)-(13, +z)-(7, -y)-(17, -y)-(1, -y)

Hormiga: 3
Reorientaciones: 3

Salida:
(11, +z)-(10, +z)-(16, +x)-(15, +x)-(14, +x)-(8, +x)-(12, +z)-(9, +z)-(5, +z)-(3, +z)-(6, +z)-(17, +z)-(1, +z)-(13, +z)-(7, +y)-(4, +y)-(2, +y)

Hormiga: 4
Reorientaciones: 2

Salida:
(11, +z)-(10, +z)-(16, +x)-(15, +x)-(14, +x)-(8, +x)-(12, +z)-(9, +z)-(5, +z)-(7, +z)-(17, +z)-(4, +z)-(13, +z)-(1, +z)-(6, +z)-(3, +z)-(2, +z)

Hormiga: 5
Reorientaciones: 3

Salida:
(11, +z)-(10, +z)-(16, +x)-(14, +x)-(15, +x)-(12, +z)-(9, +z)-(5, +z)-(4, +z)-(6, +z)-(7, +z)-(17, +z)-(1, +z)-(2, +z)-(13, +z)-(8, +x)-(3, +x)

Hormiga: 6
Reorientaciones: 2

Salida:
(11, +z)-(10, +z)-(16, +x)-(15, +x)-(14, +x)-(8, +x)-(12, +z)-(9, +z)-(5, +z)-(7, +z)-(17, +z)-(4, +z)-(1, +z)-(6, +z)-(2, +z)-(13, +z)-(3, +z)

Hormiga: 7
Reorientaciones: 2

Salida:
(8, -x)-(14, -x)-(15, -x)-(16, -x)-(5, +z)-(4, +z)-(6, +z)-(7, +z)-(17, +z)-(2, +z)-(13, +z)-(1, +z)-(9, -z)-(10, -z)-(11, -z)-(12, -z)-(3, -z)

Hormiga: 8
Reorientaciones: 2

Salida:
(8, -x)-(14, -x)-(15, -x)-(16, -x)-(5, +z)-(4, +z)-(6, +z)-(7, +z)-(17, +z)-(2, +z)-(13, +z)-(1, +z)-(3, -z)-(9, -z)-(10, -z)-(12, -z)-(11, -z)

Hormiga: 9
Reorientaciones: 3

Salida:
(8, -x)-(14, -x)-(15, -x)-(16, -x)-(10, -z)-(11, -z)-(9, +z)-(5, +z)-(4, +z)-(7, +z)-(17, +z)-(1, +z)-(6, +z)-(3, +z)-(2, +z)-(13, +z)-(12, -x)

Hormiga: 10
Reorientaciones: 2

Salida:
(8, -x)-(14, -x)-(15, -x)-(16, -x)-(5, +z)-(4, +z)-(6, +z)-(7, +z)-(17, +z)-(2, +z)-(13, +z)-(1, +z)-(3, -z)-(9, -z)-(10, -z)-(11, -z)-(12, -z)

Hormiga: 11
Reorientaciones: 2

Salida:
(11, +z)-(10, +z)-(16, +x)-(15, +x)-(14, +x)-(8, +x)-(12, +z)-(9, +z)-(5, +z)-(7, +z)-(17, +z)-(4, +z)-(1, +z)-(6, +z)-(3, +z)-(2, +z)-(13, +z)

Secuencia de salida premiada
Reorientaciones: 2

Salida:
(8, -x)-(15, -x)-(14, -x)-(16, -x)-(10, -z)-(11, -z)-(12, +z)-(9, +z)-(5, +z)-(4, +z)-(7, +z)-(6, +z)-(13, +z)-(3, +z)-(17, +z)-(2, +z)-(1, +z)