Programación de las Actividades en las Grúas de Construcción mediante un método Metaheurístico para minimizar los Costos de Operación.

Yudy Eliana Castro Castro

Lizeth Paola Hernández Ortiz

Trabajo de Grado para Optar el Titulo de Ingeniería Industrial

Director:

Carlos Eduardo Díaz Bohórquez

Magister en Ingeniería Industrial

Universidad Industrial de Santander

Facultad de Ingenierías Físico - Mecánicas

Escuela de Estudios Industriales y Empresariales

Bucaramanga

2018

DEDICATORIA

A Dios, por cumplir mi sueño de pertenecer a una de las mejores universidades de Colombia, por ser mi guía y darme la fortaleza, la sabiduría y la capacidad para superar cada obstáculo que se me presentó en el transcurso de esta etapa universitaria. Dedico este proyecto a mi madre la cual siempre me apoyo e impulso para permitir que este sueño se hiciera realidad.

AGRADECIMIENTOS

Agradezco a mi familia por el apoyo recibido durante mi carrera, en especial a mi madre Vilma Castro Pinto, por escucharme siempre y brindarme los mejores consejos cuando más los necesito.

A mi padre Luis Adelmo Castro Pulido que está en el cielo y lo llevo en mi corazón.

A mi tío Saúl Castro Pinto, por estar siempre pendiente de cada cosa que he necesitado, por brindarme su apoyo incondicional.

A cada uno de mis amigos por compartir momentos gratos, donde maduramos y crecimos para ser mejores personas.

Al profesor Carlos Eduardo Díaz Bohórquez, por brindarme el apoyo y orientación en este trabajo de grado.

A mi compañera de proyecto, por haberme acompañado en el desarrollo del proyecto de grado.

Al grupo de investigación OPALO en especial a Fabián Alexander Torres Cárdenas por ser ese apoyo incondicional, por ser mi guía y orientador en el desarrollo del proyecto de grado.

Finalmente, a la Universidad Industrial de Santander por darme la oportunidad de pertenecer a la Escuela de Estudios Industriales y Empresariales de la cual me siento orgullosa y me llevo muchos recuerdos memorables.

Yudy Eliana Castro Castro

DEDICATORIA

A Dios doy infinitas gracias, por brindarme la oportunidad de desarrollar este proyecto, el cual dedico a mis padres por su esfuerzo y apoyo incondicional durante toda mi carrera y por permitirme vivir experiencias enriquecedoras a lo largo de este ciclo, que me ayudaron a fortalecer mi formación profesional y personal.
Con sus consejos y sus esfuerzos, fueron el motor que me impulso a superar cada una de las etapas de mi carrera profesional; y culminar con éxito el fruto de este sueño que hoy es una realidad. Continuaré comprometida, en aplicar estos conocimientos y ayudar a las personas que necesiten de una mano amiga e incondicional durante el tiempo que Dios me lo permita.

AGRADECIMIENTOS

Agradezco a mi familia por el apoyo recibido durante mi carrera, en especial a mis padres Maria Elena Ortiz y Luis Ernesto Hernández quienes fueron mi modelo a seguir y un gran ejemplo de que la perseverancia del trabajo en equipo tiene sus recompensas.

Agradezco a mis abuelos la alegría de tenerlos hoy con vida y sus consejos sabios para llevar a cabo los planes de Dios.

Agradezco a cada uno de mis amigos y compañeros que estuvieron a lo largo de mi carrera apoyándome y motivándome para dar lo mejor de mí y no desfallecer en las adversidades que se me presentaron a lo largo del camino.

Agradezco a todos mis profesores, que estuvieron presentes en cada una de las etapas de mi formación profesional y personal.

Agradezco al profesor Carlos Eduardo Díaz Bohórquez, por darme la confianza y oportunidad de trabajar como su auxiliar y ser el Director de mi proyecto de grado, para terminar con dedicación y esfuerzo, bajo su acompañamiento y orientación este trabajo.

Agradezco a mi compañera de proyecto y al grupo de investigación OPALO por haberme acompañado y asesorado en este proceso, para finalizar esta última etapa de mi proyecto.

Finalmente quiero agradecer a la Universidad Industrial de Santander por haberme abierto las puertas y darme la oportunidad de pertenecer a la Escuela de Estudios Industriales y Empresariales de la cual hoy me siento inmensamente orgullosa y me llevo muchos recuerdos memorables de cada una de las experiencias vividas junto a personas maravillosas que conocí durante mi instancia y que hoy somos grandes amigos con los cuales espero seguir a lo largo del camino y crecer juntos profesionalmente.

Lizeth Paola Hernández Ortiz

Tabla de Contenido

Lista de Figuras

Lista de Tablas

Lista de Apéndices

"Los apéndices de este trabajo se adjuntan en medio digital a través de un CD"

Resumen

TÍTULO: PROGRAMACIÓN DE LAS ACTIVIDADES EN LAS GRÚAS DE CONSTRUCCIÓN MEDIANTE UN MÉTODO METAHEURÍSTICO PARA MINIMIZAR LOS COSTOS DE OPERACIÓN. *

AUTOR: CASTRO CASTRO, Yudy Eliana.

HERNÁNDEZ ORTIZ, Lizeth Paola. **

PALABRAS CLAVE: Algoritmo de Recocido Simulado, Costos de Operación, Eficiencia, Heurísticas, Obras de Construcción, Torre grúa.

DESCRIPCIÓN: En las obras de construcción el manejo de la torre grúa es una de las principales actividades a lo largo del ciclo del proyecto. Por lo tanto, es importante secuenciar y asignar de manera correcta cada una de las tareas que debe realizar para el mayor aprovechamiento de los recursos. En la siguiente investigación se planteó la programación de tareas para mejorar la eficiencia de las operaciones de la torre grúa, debido a que en las obras de construcción no se establece un cronograma a seguir de los tiempos de entrega para el cumplimiento de estas tareas, ocasionando retrasos en los tiempos de ejecución del proyecto. Esta investigación tiene como objetivo minimizar los costos de operación a través del Algoritmo de Recocido Simulado (S.A), este método se emplea para desarrollar la planificación de las tareas en un horizonte de tiempo dado y proporcionar un cronograma confiable minimizando dichos costos. Para validar esta investigación, se emplearon tres heurísticas: Primeras en entrar, Primeras en salir (FIFO), Fecha de vencimiento más temprana (EDD) y Tiempo de procesamiento más corto (SPT). Estos modelos permitieron validar cual es el método más eficiente a la hora de realizar la programación de las actividades en la torre grúa. Se concluyó que el modelo que tuvo mejor desempeño fue el Algoritmo de Recocido Simulado a partir de 4 tamaños de 10, 20, 50 y 100 tareas con 5 instancias cada tamaño de tareas.

---

* Proyecto de grado

** Facultad de Ingeniería Físico-mecánicas. Escuela de Estudios Industriales y Empresariales. Programa de Ingeniería Industrial. Director: M.Sc. Carlos Eduardo Díaz Bohórquez.

Abstract

TITLE: PROGRAMMING ACTIVITIES IN CONSTRUCTION CRANES USING A METAHEURISTIC METHOD TO MINIMIZE OPERATING COSTS. * [3]

AUTHOR: CASTRO CASTRO, Yudy Eliana.

HERNÁNDEZ ORTIZ, Lizeth Paola [4] **

KEY WORDS: Simulated Annealing Algorithm, Operating Costs, Efficiency, Heuristics, Construction Works, Tower Crane.

DESCRIPTION: In the constructions works, the tower crane is one of the main activities throughout the project cycle. Therefore, it is necessary to manage and assign each of the task that must be performed for the greatest use of the available resources. In the following research work there were programmed the necessary task to improve the efficiency of the crane tower operations due to in the constructions works there is no established a schedule to follow the delivery times for the fulfillment of these tasks. This may cause delays in the execution time of the project. The purpose of this research work is to minimize the operation costs by using the Simulated Annealing Algorithm (S.A.). This algorithm is used to develop the planning of tasks in a given time deadline and provide a reliable schedule to minimize these costs. Three heuristics methods were used to validate this research work: First In, First Out (FIFO), Earliest due date (EDD) and Shortest processing time (SPT). These models allowed to validate which is the most efficient method when programming the tower crane activities. It was concluded that the model that had the best performance was the Simulated Annealing Algorithm from 4 sizes of 10, 20, 50 and 100 tasks with 5 instances each task size.

* Bachelor thesis

** Facultad de Ingeniería Físico-mecánicas. Escuela de Estudios Industriales y Empresariales. Programa de Ingeniería Industrial. Director: M.Sc. Carlos Eduardo Díaz Bohórquez.

Introducción

Actualmente, la industria de la construcción ha incrementado su participación en la estructura económica del país. La presidente Ejecutiva de Camacol, Sandra Forero Ramírez, presentó las propuestas del Gremio como lo son: “Fortalecer el acceso a la vivienda formal, impulsar la promoción de proyectos de construcción con seguridad jurídica, contribuir con la construcción de ciudades de calidad e incrementar la productividad de toda la cadena de valor, son los pilares de las propuestas que el Gremio estructuró para garantizar el desarrollo de la actividad edificadora en los próximos años”. Forero Ramírez (2018)

En Colombia se identifica el negocio del sector privado de la construcción como “la edificación”, el cual requiere de un eficiente manejo de Gerencia de Proyectos que busca ante todo una rentabilidad financiera. Es de anotar que prima el interés público de preservar en las nuevas construcciones, las premisas de bienestar común y aprovechamiento máximo de los limitados recursos de tiempo y dinero. Por lo tanto, el reto no es el tipo de edificación que se va a construir, ni el sistema constructivo en sí mismo, sino el poder dirigir, encausar o gestionar los procesos de todos estos conocimientos de diferentes profesiones y disciplinas para concretar un ciclo de proyectos que logren que la construcción cumpla su cometido. Este cometido es ante todo y lo más importante cumplir con que el proyecto sea rentable para los inversionistas y con las expectativas de todos los interesados. Vargas (2014)

Según Shapira, Lucko & Schexnayder (2007). Los proyectos de construcción de edificios actuales están altamente mecanizados y cada día van innovando más. Con la creciente industrialización de la construcción, se produce un cambio a la prefabricación de elementos estructurales y de acabado fuera del sitio que luego se instalan o ensamblan en lugar de producirse en el sitio. En consecuencia, el equipo de transporte está reemplazando el equipo de producción en el sitio de construcción donde el manejo de materiales y los equipos de elevación ahora dominan los sitios de construcción de edificios más que nunca y constituyen el elemento crítico para lograr la productividad.

Como lo expresa Shapira et al. (2007). Las grúas han llegado a simbolizar la construcción de edificios en sí, ya que se encargan de realizar el transporte de materiales y componentes del punto de suministro al punto de demanda. Las torres grúas están ganando popularidad en todo el mundo frente a las grúas móviles debido al aumento de los desarrollos inmobiliarios, ideal para entornos urbanos densos y con un tamaño reducido; están disponibles en diversidad de tamaños y configuraciones cada vez mayor, lo cual las posiciona en el eje central de los proyectos de construcción de altura y gran altura.

A medida que se tiene una planificación de diseño de las instalaciones dentro de los sitios de construcción, las torres grúas deben estar ubicadas en un buen lugar para lograr que el transporte de materiales pesados se realice en un menor tiempo con el fin de reducir los costos operativos y mejorar la eficiencia general. Una vez se ubica la grúa, aparece un problema para la programación de actividades de una grúa desde un punto de suministro a un punto de demanda el cuál es un puesto de trabajo, donde el tiempo de viaje está asociado con cada tarea, determinado por la

distancia entre el punto de suministro y el punto de demanda. De acuerdo con este enfoque, el gancho de la grúa debe cumplir las tareas teniendo en cuenta un tiempo de desplazamiento desde su punto de origen hasta el punto de suministro donde va a realizar un servicio que comprende un tiempo de carga de material, un tiempo de desplazamiento hacia el nodo de demanda y un tiempo de descarga del material.

En la revisión de la literatura aparecen enfoques para la solución de este problema de secuenciación de actividades. Zavichi, Madani, Xanthopoulos & Oloufa (2014), muestra como el Problema de Secuencia de Servicio de Grúa (Crane Service Sequencing Problem CSSP), se puede convertir a un Problema del Agente Viajero (Traveling Salesman Problem, TSP), suponiendo que cada tarea (comenzando desde un punto de suministro y terminando en un punto de demanda) es una ciudad y el tiempo de viaje asociado con cada tarea es la distancia entre dos ciudades. De acuerdo con este enfoque, el gancho de la grúa debe cumplir las tareas visitando los puntos de demanda y regresando a su posición inicial una vez que no quede ninguna otra tarea pendiente. Esto convierte el Problema de Secuencia de Servicio de Grúa (CSSP) en un Problema del Agente Viajero (TSP) asimétrico.

De acuerdo con Zavichi et al. (2014), el Problema de Secuencia de Servicio de Grúa (Crane Service Sequencing Problem CSSP), es similar al Problema del Agente Viajero (Travelling Salesman Problem, TSP), un conocido problema de optimización combinatoria NP-Hard. Al ser un problema NP–Hard aparecen metaheurísticas para la solución como: Búsqueda Tabú (Tabú Search, TS), Algoritmo Genético (Genetic Algorithm, GA), Optimización por enjambre de partícula (Particle Swarm Optimization, PSO) y Recocido Simulado (Simulated Annealing, SA).

En este trabajo de investigación se abordará el Problema de Secuencia de Servicio de Grúa (CSSP) de la siguiente manera, se plantearán restricciones que no se encuentran en la literatura pero que se presentan en la cotidianidad de los proyectos de construcción, se utilizará la Metaheurística de Recocido Simulado (S.A) para minimizar los costos de operación y se validará este método de solución con tres metaheurísticas: Primeras en entrar, Primeras en salir (FIFO), Fecha de vencimiento más temprana (EDD) y Tiempo de procesamiento más corto (SPT). Estos modelos permitirán validar cuál es el método más eficiente a la hora de realizar la programación de las actividades en la torre grúa.

La presente propuesta está organizada de la siguiente manera. En el capítulo 4 se encuentra la revisión de literatura relevante sobre el Problema de Secuencia de Servicio de Grúa (CSSP). Luego en el capítulo 5 se presenta el Marco Teórico. En el capítulo 6 está la descripción del Recocido Simulado. Luego en el capítulo 7 se presenta el problema a tratar en este trabajo, el cual es la programación de las actividades mediante un método metaheurístico para minimizar los costos de operación. En el capítulo 8 se describe el algoritmo propuesto para resolver el Problema de Secuencia de Servicio de Grúa (CSSP). Luego en el capítulo 9 se muestra la validación del algoritmo propuesto y en el capítulo 10 se muestran los resultados computacionales. Por último, en el capítulo 11 y capítulo 12 se muestran las conclusiones del presente proyecto y las recomendaciones para futuras investigaciones.

Tabla 1.

Cumplimiento de objetivos del proyecto

Objetivo	Cumplimiento
Realizar una revisión de literatura sobre los métodos empleados en la Programación de actividades de grúas en construcción, así como los enfoques propuestos por distintos autores.	Capítulo 4
Formular un modelo matemático que represente el problema de Programación de actividades de grúas en construcción.	Capítulo 7
Plantear una metaheurística para evaluar la solución del problema a partir de la revisión de literatura.	Capítulo 8
Diseñar un código en MATLAB que sirva para dar solución al problema de Programación de actividades de grúas en construcción.	Capítulo 8
Validar los resultados del algoritmo propuesto con diferentes heurísticas encontradas en la literatura.	Capítulo 9
Elaborar un artículo de carácter publicable en base a los resultados obtenidos del trabajo realizado.	Apéndice N

1. Planteamiento del Problema

Según Zavichi et al. (2014) el Problema de Secuencia de Servicio de Grúa (CSSP) es un TSP asimétrico, el cual se puede representar mediante un grafo dirigido G= (N, A) donde N es un conjunto de elementos llamados vértices o nodos y A es un conjunto cuyos elementos llamaremos arcos o aristas. Cada arco o arista representa una conexión directa entre dos nodos o elementos de N (p.71). En las obras de construcción se cuenta con una disposición del sitio de grúa, nodos de demanda y nodos de suministro, en el que el material se debe entregar desde los puntos de suministro a los puntos de demanda en función de sus tareas utilizando una torre grúa. Cada nodo de demanda debe enviar sus tareas al operador de la grúa para recibir ciertos materiales. La persona encargada de la programación de las actividades debe decidir el orden de cumplimiento de la tarea, tratando de minimizar el tiempo de operación total como objetivo, respecto a diferentes restricciones como el tiempo de ejecución y la precedencia de las tareas.

La solución al problema involucra determinar qué tareas se asignan a la torre grúa y su secuencia de procesamiento con el objetivo de minimizar el costo de utilización de la grúa, el cual está relacionado directamente con el Makespan, es decir, el máximo tiempo de operación de las tareas (Makespan). Por consiguiente, la programación de actividades en la industria de la construcción es un tema que ha sido estudiado en la última década y ha ido evolucionando en los últimos años con el fin de mejorar la eficiencia de las operaciones en los proyectos de construcción. En el área de Investigación de Operaciones se presentan problemas representativos de optimización combinatoria que presentan soluciones factibles a la temática en desarrollo que

permiten reducir los costos de operación y cumplir con los tiempos de entrega establecidos al inicio del proyecto. (Zavichi et al., 2014)

Las restricciones planteadas para resolver el Problema de Secuencia de Servicio de Grúa (CSSP) son: Los tiempos de entrega de las tareas, se pueden adelantar o retrasar dependiendo de la secuencia de actividades; el adelanto de las tareas generará un tiempo ocioso de la grúa y el retraso de las tareas hará que los recursos de la actividad permanezcan ociosos, por este motivo se incurrirá en costos adicionales como el costo de atraso, además si la actividad la cual se retrasa se encuentra en la ruta crítica del proyecto, incurrirá en un porcentaje de penalización al costo de retraso.

En este proyecto se abordará el Problema de Secuencia de Servicio de Grúa (CSSP) de la siguiente manera, se utilizará la Metaheurística de Recocido Simulado (S.A) para minimizar los costos de operación.

El CSSP es un problema considerado difícil de resolver, denominándose en lenguaje computacional NP-Complete, es decir, es un problema para el que no podemos garantizar que se encontrará la mejor solución en un tiempo de cómputo razonable. Para dar solución se emplean diferentes métodos, entre los cuales, los principales se denominan heurísticas cuyo objetivo es generar soluciones de buena calidad en tiempos de cómputo mucho más pequeños (soluciones óptimas tiempo – respuesta). (Fuentes Penna, 2014).

Según investigaciones realizadas por Zavichi et al. (2014), ‘‘el Problema de Secuencia de Servicio de Grúa (CSSP) es similar al problema del Agente Viajero (TSP), un conocido problema de optimización combinatoria’’ (p.72). Además, el planteamiento de este problema sirve como punto de partida para mejorar las operaciones puntuales de los proyectos de construcción, es esencial el uso eficiente del equipo de construcción para la finalización exitosa de los proyectos.

Dado que en la literatura no se encuentran instancias para el Problema de Secuencia de Servicio de Grúa (CSSP), en el presente proyecto se proponen y evalúan las instancias de forma aleatoria dadas a partir del conocimiento de la cotidianidad de las operaciones de los proyectos de construcción; esto permitirá la evaluación y validación de futuros métodos de solución planteados.

2. Justificación

De acuerdo con lo expresado por Forero Ramírez (2018), la Cámara Colombiana de la construcción (CAMACOL) para el 2018 argumentó que la proyección de alcanzar los niveles de generación de valor agregado del sector promedio de la última década, tendrá un crecimiento en un 4,6%, lo cual indica que el sector de la construcción se encuentra en constante crecimiento.

En el entorno de la construcción, un aspecto importante que se presenta con frecuencia en las obras, es el incumplimiento de los plazos establecidos para la entrega del proyecto. Debido a que este se convierte en un factor crítico con el cual se mide la coordinación y planificación de las empresas, es fundamental lograr la máxima eficiencia de las operaciones que se van a realizar en el proyecto. A partir de esta problemática, en los últimos años se hace notable la importancia de la utilización de herramientas disponibles actuales como la maquinaria y otros instrumentos de apoyo.

Hoy en día, con la necesidad de operaciones puntuales, dentro del presupuesto y de alta calidad en proyectos de construcción, solo el equipo de construcción supone una gran carga financiera para los proyectos y puede causar pérdidas económicas si no se utiliza de manera eficiente. Las grúas se encuentran entre los costosos equipos de construcción, las cuales desempeñan un papel importante en los sitios de trabajo, especialmente en proyectos de construcción de gran altura. Las actividades que dependen de grúas generalmente se encuentran en la ruta crítica del proyecto, por

lo tanto, mejorar las operaciones de la grúa puede mejorar significativamente el rendimiento del proyecto. (Zavichi et al., 2014, p.69).

Las operaciones de grúa en un sitio de construcción son críticas ya que presentan demoras en el momento de carga y descarga de materiales o componentes en los puntos de almacenamiento y en los puntos donde se encuentre el equipo de trabajo a la espera del material para iniciar o dar continuación a las tareas asignadas de forma anticipada por el encargado de la obra; el retraso de las operaciones puede aumentar los costos del proyecto e impedir el progreso de las actividades de construcción posteriores y a su vez reprogramar la finalización del proyecto de manera que se acumulen costos adicionales de alquiler de las grúas. El alto costo de las actividades de construcción vencidas o con tiempos ociosos dificulta el progreso de las actividades en curso y causa cuellos de botella imprevistos en el progreso diario de la construcción. (Monghasemi, Nikoo & Adamowskiet, 2016).

La programación de actividades en la industria de la construcción es un tema que ha sido estudiado en la última década y ha ido evolucionando en los últimos años con el fin de mejorar la eficiencia de las operaciones en los proyectos de construcción. En el área de Investigación de Operaciones se presentan problemas representativos de optimización combinatoria que presentan soluciones factibles a la temática en desarrollo que permiten reducir los costos de operación y cumplir con los tiempos de entrega establecidos al inicio del proyecto. (Zavichi et al., 2014).

En la mayoría de los casos de estudio analizados, el operador de la grúa es el encargado de seleccionar la secuencia con la que se deben cumplir las tareas en función de su experiencia

personal y criterio, o utiliza reglas de programación heurística, por ejemplo, Primeros en entrar, Primeros en salir (First In, First Out FIFO), esta regla consiste en que las tareas se deben cumplir en el orden en que las recibe el operador sin tener en cuenta su prioridad, incurriendo en costos adicionales de mano de obra o de maquinaria por no tener una debida planificación de actividades a desarrollar. (Monghasemi et al., 2016)

Con base en lo anterior, surge la necesidad de estudiar varios métodos de optimización aplicados en investigaciones previas acerca de la utilización de grúas como herramientas para la programación de actividades en los proyectos de construcción y lograr un desarrollo más rápido y eficiente de sus operaciones en las construcciones.

Este estudio contribuye a proporcionar una herramienta de planificación y optimización de operaciones en función de que el resultado se de en el menor tiempo posible y así se pueda utilizar para realizar la asignación de las tareas de manera eficiente y mejorar las operaciones de la torre grúa. Por lo tanto, el objetivo principal de este documento es diseñar un método de solución para la programación de las actividades en las grúas de construcción a través del algoritmo de Recocido Simulado (SA) que permita minimizar los costos de operación.

3. Objetivos

3.1 Objetivo General

Diseñar un método de solución para la Programación de las actividades en las grúas de construcción a través de una metaheurística que permita minimizar los costos de operación.

3.2 Objetivos Específicos

- Realizar una revisión de literatura sobre los métodos empleados en la Programación de actividades de grúas en construcción, así como los enfoques propuestos por distintos autores.
- Formular un modelo matemático que represente el problema de Programación de actividades de grúas en construcción.
- Plantear una metaheurística para evaluar la solución del problema a partir de la revisión de literatura.
- Diseñar un código en MATLAB que sirva para dar solución al problema de Programación de actividades de grúas en construcción.
- Validar los resultados del algoritmo propuesto con diferentes heurísticas encontradas en la literatura.
- Elaborar un artículo de carácter publicable en base a los resultados obtenidos del trabajo realizado.

4. Revisión de la literatura.

El estudio de la investigación se inició con una búsqueda avanzada de artículos científicos el cual fue realizado por medio de la plataforma virtual de la base de datos WEB OF SCIENCE, la cual pertenece a los recursos electrónicos disponibles de la Universidad Industrial de Santander. El resultado de la búsqueda inicial arrojó un resultado de 179 artículos en inglés de los últimos 9 años en los que se hicieron publicaciones de investigaciones relacionadas con la optimización de ubicaciones de torres grúas, sitios de suministro de materiales y sitios de demanda como las operaciones de grúas en la construcción utilizando diferentes técnicas de optimización y secuenciación de actividades de las torres grúas.

Luego mediante un proceso de refinado que permite la plataforma WEB OF SCIENCE se redujo a 15 artículos relacionados con el tema de investigación a desarrollar (apéndice A).

En la revisión de literatura se encuentran varios temas de investigación relacionados con la programación de las grúas y diferentes métodos empleados para dar solución a esta programación, algunos de ellos son: Algoritmos exactos, Heurísticas, Metaheurísticas e Híbridos.

Takagi & Nishimura (2003) centran sus investigaciones tanto en el control del balanceo de la carga como la vibración de la torre mediante la capacidad inherente de la torre grúa. Realizaron un análisis para comparar la estabilidad robusta de las variaciones de los parámetros entre el sistema de control centralizado y el descentralizado. El experimento mostró que era difícil mejorar

el rendimiento con un sistema de control centralizado debido a que la teoría tuvo dificultades al ajustar el control arbitrariamente. Por lo tanto, el sistema de control descentralizado fue suficientemente efectivo en el control de la torre grúa.

Vaughan, Smith, Kang & Singhose (2011) investigan sobre cómo reducir el desfase temporal en la entrega de solicitudes ya que no hay una buena razón para descuidar el retraso de tiempo; el cual no puede ser eliminado. Los autores dan a conocer un método de control que ayuda al operador humano al mostrar gráficamente una predicción de dónde se debe detener la grúa.

Huang, Wong & Tam (2011) en su investigación estudiaron los efectos de la ubicación de una torre grúa sobre la capacidad requerida y, por lo tanto, sobre sus costos de alquiler y operación, así como la consiguiente necesidad de considerar este efecto para optimizar la ubicación de la torre grúa en una obra. El problema de asignación cuadrática (Quadratic Assignment Problem, QAP), de naturaleza no lineal, se ha desarrollado para simular el procedimiento de transporte de material. Al aplicar conjuntos de restricciones lineales, el problema cuadrático se puede linealizar y el problema se puede formular en un problema de programación lineal entera mixta (Mixed Integer Linear Programming, MILP). Los resultados numéricos muestran que los resultados de MILP superan a los optimizados por los algoritmos genéticos (GA), por lo tanto, el enfoque MILP permite una mayor flexibilidad para expandir la formulación del problema agregando restricciones lineales para considerar condiciones del sitio más complejas y realistas.

Lien & Cheng (2014) propusieron una Optimización por enjambre de partícula (PSO) un algoritmo de enjambre híbrido que integra las ventajas respectivas de las abejas melíferas y los

enjambres de aves, este algoritmo se utiliza para mejorar la capacidad de búsqueda local (BA), la capacidad de búsqueda global del (PSO) y buscar registros de experiencias pasadas durante el proceso de búsqueda de optimización. El algoritmo de partícula de abeja (PBA) se utilizó para optimizar la ubicación de la torre grúa y también para optimizar la distancia de operación y la frecuencia entre los puntos de demanda y suministro en términos de costos operativos totales en función de los requisitos de materiales en los puntos de demanda y suministro. Los resultados que arrojó la investigación determinan que, si bien el (PBA) tiene un buen rendimiento en la optimización de la ubicación de la torre grúa, el algoritmo no puede minimizar los costos de operación para la demanda y la capacidad del punto de suministro para problemas de gran dimensión.

Zavichi et al. (2014) investigaron el impacto de priorizar el Problema de Secuencia de Servicio de Grúa (CSSP) en el tiempo total de viaje, usando un innovador método de optimización que modifica específicamente el modelo de optimización del problema del agente viajero (TSP) para maximizar la eficiencia de las operaciones de la torre grúa de construcción. El CSSP se puede formular como un TSP, suponiendo que cada solicitud (comenzando desde un punto de suministro y terminando en un punto de demanda) es una ciudad y el tiempo de viaje asociado con cada enlace es la distancia entre dos ciudades conectadas por el enlace. El CSSP se convierte en un TSP asimétrico lo cual es muy complejo de resolver. Una forma de resolver un TSP asimétrico es duplicar el tamaño de la matriz de distancia reemplazando cada nodo en el gráfico con dos nodos, teniendo en cuenta que los nodos añadidos representan ciudades ficticias. Este procedimiento transforma la matriz asimétrica en una matriz simétrica permitiendo así encontrar el costo mínimo de transporte y la ruta de viaje asociada. Para este propósito, se proponen un método de

optimización combinatoria exacta para superar las limitaciones existentes del modelo general de (TSP) que lo hacen inaplicable a la Secuencia de Servicios (CSSP) de grúas de construcción. El modelo desarrollado optimiza el tiempo de viaje, que es una parte importante de las operaciones del ciclo de la grúa, especialmente en construcciones de gran altura donde los tiempos de carga y descarga constituyen una pequeña porción del ciclo de movimiento de la grúa.

Abdelmegid, Shawki & Abdel-Khaked (2015) plantean una optimización de algoritmos genéticos (GA) para resolver el problema de ubicación de la torre grúa en sitios de construcción con el objetivo de minimizar el tiempo total de transporte. El modelo propuesto se desarrolla en base a la mejora de los modelos anteriores teniendo en cuenta muchos factores que se han descuidado, como la variación en la velocidad vertical de la extensión del gancho de la torre grúa. Otra parte importante y mejorada en los cálculos es el método para calcular el número de ciclos necesarios para cada tarea en función de la capacidad de la torre grúa para cada ubicación, que se ha supuesto fija en estudios previos. Además, el sistema de base de la torre grúa también está representado para tratar a la grúa como un área, no como un punto para aumentar la confiabilidad del modelo.

Wang et al. (2015) desarrollan un enfoque integrado que combine con el Modelado de información de construcción (Building Information Modeling**,** BIM) y con el Algoritmo de luciérnaga (Firefly Algorithm, FA) para generar automáticamente un plan de distribución de torre grúa óptima. En primer lugar, BIM se utiliza para proporcionar entradas para el modelo matemático y después el FA se usa para determinar las ubicaciones óptimas de la torre grúa y los puntos de suministro. Finalmente, el esquema óptimo de distribución de la torre grúa se visualizará

y evaluará a través de simulación basada en BIM. Los hallazgos de este estudio permiten a los gerentes de sitio generar, visualizar y simular automáticamente el esquema de diseño de la torre grúa. Puede ayudar significativamente a mejorar la productividad en el sitio porque la entrada manual consume mucho tiempo. Por lo tanto, este estudio contribuye al conjunto de conocimientos relacionados con la planificación del diseño del sitio de construcción.

Marzouk & Abubakr (2016) presentan un marco para la selección de tipos de torre grúa y ubicaciones en sitios de construcción. El marco considera tres modelos principales: 1) modelo de toma de decisiones para seleccionar el tipo de torre grúa por medio del uso de BIM y algoritmos genéticos, 2) modelo de optimización para la selección del número ideal y ubicación de torre grúa y 3) modelo de simulación 4D para simular operaciones de torre grúa. El modelo de toma de decisiones se desarrolló para seleccionar el tipo de torre grúa que coincida con los requisitos del proyecto y utiliza una técnica de proceso de análisis jerárquico (Analytic Hierarchy Process, AHP) para ayudar en la selección de la torre grúa más adecuadas. Se realizaron análisis de sensibilidad según los criterios del AHP para revelar los criterios más críticos y la medida más crítica del desempeño que podría afectar el proceso de toma de decisiones. Los resultados revelaron que la torre grúa de cabeza de martillo es la más sensible ya que está asociada con los coeficientes de sensibilidad más altos.

Tubaileh (2016) presenta una nueva técnica para encontrar el diseño óptimo de la grúa y los puntos de suministro. En la formulación realizada, tiene como objetivo optimizar el costo total de transporte de material, los aspectos dinámicos y cinemáticos de la grúa los cuales se modelan con el fin de desarrollar un diseño que cumpla con las limitaciones de rendimiento de la grúa, por

consiguiente, se desarrolla un algoritmo para estimar la capacidad de carga y la velocidad de elevación máxima de acuerdo con las tablas de rendimiento de la grúa real seleccionada. La caja de herramientas de optimización del grupo de algoritmos numéricos (Numerical Algorithms Group, NAG) para Matlab y el algoritmo de recocido simulado (SA), se utilizaron para encontrar el costo de viaje óptimo para todas las operaciones de la grúa en un período de tiempo determinado. Al realizar una comparación entre los resultados obtenidos y los estudios anteriores, se demuestra la efectividad del método propuesto para producir el diseño factible y aplicable de la grúa y las instalaciones del sitio de construcción.

Monghasemi et al. (2016) Proporcionan un modelo de optimización de solicitud de servicio capaz de priorizar las solicitudes de servicio de grúa de tal manera que se minimicen los tiempos pendientes de múltiples solicitudes de servicio incumplidas. Para esto, se utilizó el método de la menor desviación para priorizar las posibles alternativas de secuencia en términos de tiempos pendientes de solicitud. Aunque la secuencia de solicitudes de tiempo de viaje resulto en un menor tiempo de viaje de la grúa en comparación con las horas equitativas pendientes, las solicitudes optimizadas de tiempo de viaje no fueron capaces de distribuir uniformemente los tiempos pendientes. Por lo tanto, el equipo de construcción distante se colocará último en cola para recibir su servicio.

Sadat, Nadoushani, Hammad & Akbarnezhad (2017) Centran su inventigación en resolver el problema de ubicación de la torre grúa (Tower Crane Location Problem, TCLP), considerando los costos de operación y alquiler, desarrollarón un modelo de programación entera mixta (Mixed Integer Programming, MIP) el cual toma en cuenta los efectos de la ubicación de la grúa en su

capacidad requerida, además del tiempo de operación, cuando se minimizan los costos totales de las operaciones de elevación. Los resultados de la ubicación de la grúa indicaron diferencias considerables cuando se descuida la capacidad de la torre grúa. Los beneficios del modelo propuesto se destacaron por una reducción del 31.9% en los costos totales logrados al aplicar el modelo propuesto en comparación con los costos asociados con la solución óptima obtenida utilizando un modelo que minimiza los costos de operación solamente.

Al Hattab, Zankoul & Hamzeh (2017) centran su trabajo en optimizar las operaciones de la grúa en edificios de gran altura considerando el espacio de superposición entre las grúas. La planificación anticipada (Look-Ahead Planning, LAP) y la definición de tarea de la grúa permiten a los usuarios preparar un cronograma diario confiable de las tareas que deben realizar las grúas. La duración de estas tareas se calcula a partir de un conjunto de ecuaciones derivadas matemáticamente y un sistema de coordenadas basado en un Modelado de Información de Construcción (Building Information Modeling, BIM). Este cronograma se usa luego como entrada para un modelo de simulación – optimización (Simulation Optimization, SO) que aplica metaheurísticas para buscar secuencias óptimas de asignaciones de grúas y detectar cualquier colisión. Los resultados del estudio de caso indican que la duración del cronograma de la grúa y las tasas de utilización del LAP adecuado se pueden mejorar aún más mediante la optimización. Por lo tanto, se puede obtener una secuencia mejorada de la asignación de torre grúa si el LAP inicial se realiza correctamente, seguido de la optimización. Con una buena entrada, el modelo de optimización puede producir mejores resultados, por lo tanto, el impacto combinado de LAP y la optimización proporciona mejores resultados.

Al Hattab, Zankoul, Barakat & Hamzeh (2018) Exploraron el impacto de las grúas superpuestas, utilizadas en edificios de gran altura, en la flexibilidad operativa, que es el equilibrio entre la duración del cronograma, la utilización de la grúa y la seguridad. Desarrollaron y aplicaron un modelo de simulación (Building Information Modeling, BIM) en un proyecto real para analizar y comparar los impactos de diferentes tamaños de superposición. También desarrollaron un modelo de simulación de eventos discretos (Discrete Event Simulation, DES) que toma como entrada la demanda de carga de trabajo y la capacidad de la grúa en función del espacio de superposición. Los resultados muestran que el aumento del tamaño de superposición de la grúa puede lograr desviaciones de programación más cortas, así como tasas de utilización de la grúa más altas y más equilibradas, incluso con un aumento en el número de incidencias de proximidad. La planificación adecuada de las asignaciones de las tareas a la grúa y la maniobra de los movimientos de la grúa son factores importantes a considerar para obtener los beneficios deseados del aprovechamiento de superposiciones de grúas.

Existe una gran cantidad de publicaciones que describen diferentes aspectos de diseño de ubicación de las grúas y solo se encuentran estos autores Zavichi et al.(2014), Monghasemi et al. (2016) y Al Hattab et al. (2018) haciendo referencia a la programación de tareas en las grúas, que buscan minimizar los costos teniendo en cuenta los tiempos de operación y en la literatura no se encuentra ningún autor que considere el modelo de costos que se va a utilizar en este proyecto. Algunos investigadores aplicaron diferentes métodos metaheurísticos para resolver el problema y dar soluciones factibles, pero no óptima.

Por ejemplo Zavichi et al.(2014) plantea un modelo de optimización de enteros para un diseño de sitio predefinido en el que determina la secuencia de costo mínimo (tiempo de viaje) que satisface cada solicitud una sola vez. Tal secuencia se conoce como un *recorrido* o *secuencia hamiltoniano* (o *ciclo de recorrido hamiltoniano*). A su vez destaca la utilidad del modelo de optimización sugerido y compara su rendimiento con tres métodos de planificación heurísticos convencionales: Primeras en entrar, primeras en salir (First In First Out, FIFO), tiempo de procesamiento más corto (Shortest Processing Time, SPT) y fecha de vencimiento más temprana (Earliest Due Date, EDD).

5. Marco Teórico

5.1 Secuenciación de las tareas

Según Chase et al. (2005). El proceso de determinar qué tarea se inicia primero en alguna máquina o centro de trabajo se conoce como secuenciación o secuenciamiento de prioridades. Las normas prioritarias son aquellas utilizadas en la obtención de una secuencia de tareas. Pueden ser muy sencillas y requieren sólo que las tareas sean secuenciadas de acuerdo al orden en el cual cada una de estas tareas debe ser procesada, como el tiempo de procesamiento, la fecha de vencimiento o el orden de llagada. (p. 683).

Para evaluar las normas prioritarias se utilizan las siguientes medidas estándar de la ejecución del programa:

1. Ajustarse a las fechas de vencimiento de los clientes o de las operaciones mediatas.
2. Minimizar el tiempo del flujo (el tiempo que requiere una tarea en el proceso).
3. Minimizar el inventario del trabajo en proceso.
4. Minimizar el tiempo de inactividad de las máquinas o los trabajadores.

Castillo & Fandiño (2005) afirman que ‘‘El objetivo de la secuenciación, es la asignación eficiente de máquinas y otros recursos a los trabajos, o a las operaciones contenidas en estos, y la determinación del orden en el cual cada uno de estos trabajos debe ser procesado’’ (p.58).

Según Conway, Maxwell & Miller (1967) ‘‘Las siguientes hipótesis aparecen frecuentemente en la literatura sobre secuenciación de trabajos en máquinas’’ (p.36):

- Las máquinas están siempre disponibles y nunca dejan de funcionar.
- Cada máquina puede procesar a lo sumo un trabajo a la vez.
- Cualquier trabajo puede ser procesado únicamente en una máquina a la vez.
- Los tiempos de preparación de todos los trabajos son cero, por ejemplo, todos los trabajos están disponibles al comienzo del proceso.
- No se permiten interrupciones, es decir, cuando una operación ha comenzado debe terminarse antes de empezar otra en la misma máquina.
- Los tiempos de cambio son independientes de los programas y están incluidos en los tiempos de procesado.

- Los tiempos de cambio y las restricciones tecnológicas son deterministas y se conocen de antemano, y similarmente ocurre con las fechas de entrega.

Con las anteriores hipótesis, se refleja que la teoría se aleja de la realidad dado que los entornos donde se realiza la producción son dinámicos y sujetos a una serie de variaciones de carácter estocástico. Estas perturbaciones pueden modificar el estado del sistema productivo y afectar el rendimiento del mismo, entre estas perturbaciones se encuentran eventos relacionados con los recursos como los fallos en máquinas, bajas de los operarios, no disponibilidad de materiales o herramientas, etc.

Según Liu & Maccarthy (2015) ‘‘En muchos problemas industriales reales estas hipótesis no son válidas y es por eso que en los últimos años han aparecido modelos y procedimientos de solución que relajan una o varias de las hipótesis anteriores’’ (p.3305).

Las variantes más importantes del problema de secuenciación son:

- **Programación de tipo taller de trabajo (Job Shop Scheduling, JSP):** Bard & Feo (1989) ‘‘Consiste en un conjunto finito de trabajos, en donde cada trabajo está dividido en operaciones o actividades, estas operaciones serán procesadas o ejecutadas en un determinado número de recursos o máquinas, cada trabajo tiene su propia ruta de actividades a seguir’’ (p.249).
- **Programación de tareas (Task Scheduling, TS):** Tupia Anticona (2005) ‘‘Se tiene un conjunto de tareas, cada una de ellas está asociada a una duración determinada de tiempo, el objetivo es programar las tareas en unas máquinas procurando el orden más apropiado,

atendiendo que para que se ejecute una tarea, sus predecesoras necesariamente tienen que ser ejecutadas'' (p.1).

- **Programación de tienda de flujo (Flow Shop Scheduling, FS):** Pinedo (2005) ''Consiste en un número de trabajos que son procesados en un número de máquinas, cada trabajo debe ser procesado por todas las máquinas en el mismo orden'' (p.28).

5.2 Optimización

Según Duarte Muñoz, Pantrigo Fernández & Gallego Carillo (2007) ''La optimización se concibe como el proceso de intentar encontrar la mejor solución posible a un problema, generalmente en un tiempo limitado. Se considera un problema de optimización como un problema en el que hay varias posibles soluciones y alguna forma clara de comparación entre ellas'' (p.1).

Desde el punto de vista matemático, según Martí (2003) ciertos problemas pueden ser matemáticamente modelados representando lo más cercano a la vida real, para los cuales existen modelos bien definidos. Un problema de optimización P se puede formular como una lista de 3 elementos P= (f, SS, F) definida como sigue

$$P = \begin{cases} Opt: f(x), & Función\ Objetivo \\ & s.a \\ x \in F \subset SS, & Restriciones \end{cases}$$

Donde f es la función a optimizar (maximizar o minimizar), F es el conjunto de soluciones factibles y SS es el espacio de soluciones (p.2).

5.2.1 Optimización Combinatoria. La combinatoria, con base a la definición planteada por Lawer (1996), es la rama de la matemática que trata de ordenar objetos usualmente finitos en número y sujetos a varias restricciones. Por otro lado, Pastor Moreno (1999) lo define como un problema de optimización que busca la maximización o minimización de la función objetivo dentro de la región de soluciones factibles.

Ahora, si el espacio de solución junto con el conjunto de soluciones factibles del problema es discreto, es decir, si son conjuntos de un número finito de elementos, se denomina problema de optimización combinatoria (p.15).

Martí (2001) afirma que en estos problemas el objetivo es encontrar el máximo (o el mínimo) de una determinada función sobre un conjunto finito de soluciones. No se exige ninguna condición o propiedad sobre la función objetivo o la definición del conjunto S. Es importante notar que, dada la finitud de S, las variables han de ser discretas, restringiendo su dominio a una serie finita de valores. Habitualmente, el número de elementos de S es muy elevado, haciendo impracticable la evaluación de todas sus soluciones para determinar el óptimo (p.60).

A partir de esto, Blum & Roli (2008) establecen las siguientes condiciones para denominar a un problema, problema de optimización combinatoria.

- Existencia de unas variables de decisión $X= [X_1, X_2, ..., X_n]$
- Se debe determinar el dominio de cada una de las variables $D_1, D_2, D_3, ..., Dn$ y así, poder determinar el tipo de optimización combinatoria.

- Deben existir las restricciones entre variables que ayudan a discriminar las soluciones existentes.
- Debe existir una función objetivo *f* tal que el objetivo del problema sea minimizar o maximizar dicha función.

La solución óptima que se obtiene de este problema debe ser $S_1 \epsilon S$ tal que $f(S_1) \leq f(S_n)$ si se desea minimizar la función objetivo, si se desea maximizar la función objetivo entonces $f(S_1) > f(S_n)$, $\forall$ n

Donde S_n representa todo el espacio de soluciones factibles (p.268).

Martí (2001) ‘‘Los problemas de optimización son muy comunes en la vida real, se presentan tanto en distribución como en la organización de la producción, en el diseño de redes de telecomunicación y en problemas de ingeniería y reingeniería de software’’ (p.60). Estos tipos de problemas suelen clasificarse según la dificultad computacional que presentan.

Algunos ejemplos de problemas de optimización clásicos son:

- Problema binario de la mochila (Knapsack Problem).
- Problema del agente viajero (TSP).
- Problema de asignación cuadrática (QAP).
- Programación entera (integer programming).

5.3 Complejidad Computacional

Gallego Rendon & Escobar Zuluaga (2006) afirman que siempre existe una forma de encontrar la solución óptima de un problema combinatorio y la más simple consiste en realizar una *búsqueda exhaustiva*. En este caso se debe encontrar la función objetivo de cada solución factible (configuración factible) y la solución óptima será aquella solución factible que tiene la mejor función objetivo, a pesar de ello, esta estrategia es impracticable porque se observa que el esfuerzo computacional crece de manera exponencial con el tamaño del problema.

Sin embargo, no todos los problemas combinatorios son tan complejos de resolver. Existen algunos problemas para los cuales existen algoritmos que los resuelven con un esfuerzo computacional que crece de manera polinomial con la dimensión del problema.

Los problemas combinatorios pueden ser clasificados en 2 grandes grupos de acuerdo con el esfuerzo computacional requerido.

Clase P: Problemas tipo *P* para los cuales existen algoritmos con esfuerzo computacional de tipo polinomial para encontrar su solución óptima.

Clase NP: Problemas tipo *NP* (non polinomial) para los cuales no se conocen algoritmos con esfuerzo computacional de tipo polinomial para encontrar su solución óptima (p.4).

Algunos problemas *NP* son especialmente difíciles de resolver y son denominados ***NP-COMPLETE*** y ***NP-HARD***.

Zabala (2014). El problema ***NP-COMPLETE*** tiene como características que es un problema *NP* y que todos los demás problemas de *NP* se pueden reducir a problema *NP-COMPLETE* añadiéndole un costo en el tiempo polinomial. Es importante resaltar que si se demuestra que existe un problema ***NP-COMPLETE*** que pertenece a los problemas ***P*** entonces todos los problemas ***NP-COMPLETE*** pertenecen a los problemas ***P.*** (p.12).

El problema ***NP-HARD*** no tiene ningún algoritmo polinómico que lo resuelva, es así cómo se utilizan algoritmos que ofrezcan una respuesta aproximada a la óptima.

Zavichi et al. (2014) define el *problema del Agente Viajero (TSP)*, como un problema combinatorio desde el punto de vista de la complejidad computacional pertenece a la clase ***NP Hard.***

En cuanto a la optimización combinatoria Zabala (2014) afirma que algunos de estos problemas son fáciles de resolver porque son problemas lineales y por tanto se solucionan mediante métodos exactos; pero la mayoría de los problemas de optimización combinatoria son de difícil solución, considerándose problemas ***NP-HARD*** por lo que se hace necesaria la aplicación de métodos aproximados.

5.4 Programación Lineal Entera y Entera Mixta

Cornejo (como se citó en Hillier & Lieberman, 2010) define un modelo de programación lineal entera (ILP) como aquel donde las variables son números enteros no negativos. En las situaciones reales, el analista se enfrenta a "decisiones sí o no", las que pueden representarse con variable denominadas binarias en caso de que la programación sea binaria; es un caso particular de programación lineal entera. Cuando sólo es necesario que algunas de las variables sean enteras y el resto continuas, el modelo recibe el nombre de problema de Programación Lineal Entera Mixta (MILP). Esta clasificación incluye modelos que además de tener variables enteras no negativas y variables continuas, tienen también variables binarias

"Los problemas de programación con enteros se formulan de la misma manera que los problemas de programación lineal, pero agregando la condición de que al menos alguna de las variables de decisión debe tomar valores enteros" (Bermúdez Colina, 2007, p.89).

5.5 Traveling Salesman Problem (TSP)

Fuentes Penna (2014) afirma que el problema del Agente Viajero (TSP) se encuentra clasificado como Problema de optimización Combinatoria, es decir, es un problema donde intervienen cierto número de variables donde cada variable puede tener N diferentes valores y cuyo número de combinaciones es de carácter exponencial, lo que da lugar a múltiples soluciones óptimas factibles (soluciones que se calculan en un tiempo finito) para una instancia.

TSP es un problema considerado difícil de resolver, denominándose en lenguaje computacional *NP-Completo,* es decir, es un problema para el que no podemos garantizar que se encontrará la mejor solución en un tiempo de cómputo razonable. Para dar solución se emplean diferentes métodos, entre los cuales, los principales se denominan heurísticas cuyo objetivo es generar soluciones de buena calidad en tiempos de cómputo mucho más pequeños (soluciones óptimas tiempo – respuesta) (p.2).

Según Zavichi et al. (2014) autores del artículo ‘‘*Operaciones de grúas mejoradas en la construcción utilizando la optimización de solicitud del servicio*’’, presentan el conocido método del vendedor viajero (TSP) como un problema *NP-HARD* que se ha vuelto representativo de los difíciles problemas de optimización combinatoria y es utilizado como referencia para secuenciar solicitudes de servicio de grúas con la capacidad de considerar un número relativamente grande de solicitudes para minimizar el tiempo total de viaje de la grúa.

5.6 Crane Service Sequencing Problem (CSSP)

Zavichi et al. (2014) define el Problema de Secuencia de Servicio de Grúa (CSSP) como un TSP asimétrico el cual puede describirse mejor usando un grafo es decir, la colección de vértices y bordes de conexión, asociados con los tiempos de viaje. La Figura 1 muestra un diseño del sitio utilizando un grafo que consta de nodos de demanda (C) y suministro (M) en un sitio de construcción en el que el material se entrega desde *m* puntos de almacenamiento a *n* equipos de trabajo en función de sus tareas utilizando una grúa central. Cada nodo de demanda envía sus tareas al operador de la grúa para recibir ciertos materiales. El operador debe entonces decidir el

orden de cumplimiento de la tarea, tratando de minimizar el tiempo de operación total como un objetivo con respecto a diferentes restricciones como el tiempo debido y la precedencia de las tareas.

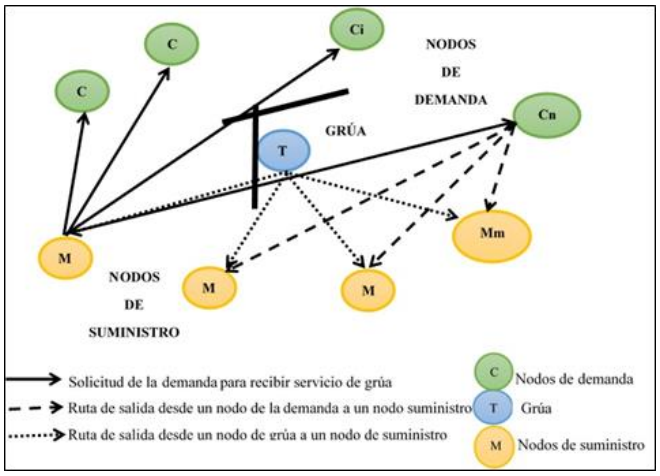

Figura 1. Ilustración gráfica de un diseño de sitio de construcción. Adaptado de Amir Zavichi, Kaveh Madani, Petros Xanthopoulos, Amr A. Oloufa, (2014). Operaciones de grúas mejoradas en la construcción utilizando la optimización de tarea del servicio.

Suponiendo que un subconjunto de cuadrillas (*w* de *n*) solicite la entrega de material, existen *varias* alternativas para que el responsable de la toma de decisiones (operador de grúa) elija el primer punto de servicio de entrega. Una vez que se elige el primer nodo de demanda objetivo, el operador debe cargar el material solicitado por este equipo de trabajo desde un nodo de suministro y entregarlo al nodo objetivo. Si no hay una nueva tarea, el proceso continúa con $w - 1$ tareas hasta que se cumplan todas las tareas pendientes de servicio de la grúa. Dado que el operador es libre de elegir cualquier orden de entregas, hay un total de $(w!)$ (permutación de *w*) formas posibles de cumplir todas las tareas. Para minimizar el tiempo total de viaje, el operador necesita encontrar la secuencia de entrega óptima.

Según Zavichi et al. (2014) el CSSP es similar a un TSP, suponiendo que cada tarea (comenzando desde un nodo de suministro y terminando en un nodo de demanda) es una ciudad y el tiempo de viaje asociado con cada enlace es la distancia entre dos ciudades conectadas por el enlace. De acuerdo con este enfoque, el gancho de la grúa debe cumplir las tareas visitando los nodos de solicitud y regresar a su posición inicial una vez que no quede ninguna otra tarea pendiente. Esto convierte el CSSP en un TSP asimétrico, en el que el costo (tiempo de viaje) de pasar de i a j es diferente del costo de pasar de j a i.

El primer paso para resolver CSSP es desarrollar una matriz de tiempo de viaje (costo o distancia) (C) asociada con los arcos de conexión. Esta matriz se conoce como matriz de tiempo-distancia, que es una matriz cuadrada ($n \times n$) como se muestra en la Figura 2. (Zavichi et al,2014).

$$C = \begin{bmatrix} C_{11} & \cdots & C_{1n} \\ \vdots & C_{ij} & \vdots \\ C_{n1} & \cdots & C_{nn} \end{bmatrix}$$

Figura 2. Matriz de tiempo – distancia. Adaptado de Amir Zavichi, Kaveh Madani, Petros Xanthopoulos, Amr A. Oloufa, (2014). Operaciones de grúas mejoradas en la construcción utilizando la optimización de tarea.

Donde n es el número de nodos en el grafo y Cij refleja el tiempo que tarda el gancho de la grúa en viajar del nodo i al nodo j, donde i es el punto de suministro y j el punto de demanda.

Zavichi et al.(2014) presenta un método de optimización para resolver el Problema de Secuencia de Servicio de Grúa (CSSP) basado en el conocido método del Agente Viajero (TSP) para la secuenciación de solicitudes de servicio de grúa con la capacidad de considerar un número

relativamente grande de solicitudes para minimizar el tiempo total de viaje de la grúa, con el fin de reducir el tiempo total de inactividad de la mano de obra y de la maquinaria en un sitio de trabajo predefinido. Por lo tanto, el desarrollo de este modelo intenta desarrollar un nuevo marco para la gestión eficiente y en tiempo real de las operaciones de grúas utilizando modelos de optimización.

La formulación matemática del modelo de optimización para resolver el CSSP es la siguiente:

Minimizar	$\sum_k \sum_l p_{kl} y_{kl}$		(1)
Sujeto a	$\sum_{l:l\neq k} y_{kl=1}$	$\forall_k, l \in V, (k,l) \in A$	(2)
	$\sum_{k:k\neq l} y_{kl} = 1$	$\forall_k, l \in V, (k,l) \in A$	(3)
	$\sum_{k\in S} \sum_{l\in S} y_{kl} \geq 1$	S $\subset$ V, 2 $\leq$ |s| $\leq$ n-2, (k, l) $\in$ A	(4)
	$y_{kl} \in (0,1)$	$\forall_k, l \in V, k \neq l$	(5)

Donde P: p_{kl} es la matriz de solicitud de servicio asociado con A.

El modelo de optimización de enteros sugerido determina la secuencia de costo mínimo (tiempo de viaje) que satisface cada solicitud una sola vez. Tal secuencia se conoce como un *recorrido* o *secuencia hamiltoniano* (o *ciclo de recorrido hamiltoniano*). La programación de enteros es un método bien conocido en la literatura clásica de optimización. Toda clase de problemas

relacionados con la secuenciación, la programación y el enrutamiento son intrínsecamente programaciones enteras. En este problema, una variable de decisión binaria y_{kl} se asocia con cada arco (k, l) y se establece igual a 1 si y solo si se usa el arco (k, l) en la solución óptima ($k \neq l$). En otras palabras, $y_{kl} = 1$ si el gancho de la grúa va directamente desde el nodo de solicitud k al nodo de solicitud l y $y_{kl} = 0$ de lo contrario (restricción 5). Las Restricciones 2-3 son restricciones de grado que especifican que cada vértice es incidente de un arco de salida (restricción 2) y un arco entrante (restricción 3). La solución considerando solo las restricciones 2 y 3 podría llevar a una solución desconectada (subtour) que debe excluirse del conjunto de soluciones. Para eliminar las soluciones que consisten en subtemas (es decir, recorridos en subconjuntos de menos de n vértices), se necesita una restricción adicional (restricción 4). S es un subconjunto de V vértices y |s| es la cardinalidad de S. En adicionalmente, Ṡ es un complemento de S. La Restricción 4 solo es válida cuando $2 \leq |s| \leq n - 2$ y evita que la solución contenga dos o más subtour desunidos.

La matriz de tiempo de viaje CSSP (matriz de solicitudes de servicio (P: $p\,kl$) es una matriz dinámica compuesta por la matriz de tiempo de viaje de ubicación ($C : (c\,i,j)$) combinada con las solicitudes en un momento determinado. La matriz anterior refleja el tiempo de viaje entre los nodos de solicitud. Esta matriz es intrínsecamente asimétrica ($P^T \neq P$).

Para resaltar la utilidad del modelo de optimización sugerido, su rendimiento se compara con tres métodos de planificación heurística convencionales, a saber, el FIFO, el SPT y el EDD, que son los enfoques predominantes para los problemas de programación. El algoritmo FIFO no implica ninguna dificultad debido a que el operador de grúa procesa las solicitudes en función del pedido recibido. Con base en el algoritmo SPT, la solicitud con el menor tiempo de viaje tiene la

prioridad más alta que se debe cumplir. Con base en el algoritmo EDD, el operador de la grúa debe atender la solicitud con fecha de vencimiento más cercana después de cada entrega.

Para evaluar el rendimiento y la utilidad del modelo de optimización propuesto para resolver CSSP con diferentes tamaños, el ejemplo numérico se resuelve para 11 tamaños diferentes con diferente número de solicitudes (es decir, 10, 20, 30, 40, 50, 100, 200, 300, 400, 500 y 1000 solicitudes). Usando una distribución de probabilidad uniforme, se generan solicitudes aleatorias para cada CSSP con un tamaño dado. Para garantizar que los problemas generados sean aleatorios, se generan 100 CSSP y se resuelven para cada tamaño de problema, lo que hace que la cantidad total de problemas resueltos sea 1100 (100×11). Cada uno de los 1100 CSSP se resuelve utilizando el modelo de optimización sugerido y los tres algoritmos de operación de la grúa heurística. Las desviaciones medias y estándar de los tiempos totales de viaje se determinan para diferentes tamaños de problema usando todos los métodos de programación, es decir, optimización FIFO, SPT, EDD y CSSP. El modelo de optimización se resuelve utilizando CONCORDE, un solucionador TSP simétrico en integración con MATLAB.

Los ahorros promedio en el tiempo de viaje total para un número diferente de solicitudes bajo diferentes métodos de programación con respecto al tiempo de viaje bajo el método FIFO se presentan en la Tabla 2. Para facilitar la comparación, el tiempo total de viaje basado en el método de programación FIFO se establece como la línea de base y otros métodos de programación se comparan con esta línea de base. Los resultados muestran que el modelo de optimización de enteros sugerido al procesamiento de la secuencia reduce el tiempo de viaje promedio en un 9%, 32% y 35% utilizando SPT, EDD y el método de programación óptimo en comparación con el

método FIFO, respectivamente, para diferentes números de solicitudes. Como se puede ver, el método de programación óptimo propuesto supera a los otros métodos. La regla heurística EDD también proporciona una programación de secuencia satisfactoria sin necesidad de optimización. Las desviaciones estándar para todos los tamaños de problema fueron menos del 7% del tiempo medio de viaje. Además, la desviación estándar disminuyó a medida que aumentaba el número de solicitudes.

Tabla 2.

Ahorro de tiempo promedio bajo diferentes métodos de programación (SPT, EDD y Modelo de Optimización de Enteros) para el ejemplo de diseño de sitio predefinido.

Número de Solicitudes	**Ahorro de tiempo relativo al método FIFO (%)**			**Tiempo de Ejecución**
	SPT	**EDD**	**Modelo de Optimización de Enteros**	
10	-1%	20%	24%	0.16 ± 0.1
20	3%	24%	28%	0.21 ± 0.33
30	3%	26%	30%	0.27 ± 0.45
40	4%	29%	33%	0.56 ± 1.35
50	5%	30%	34%	0.83 ± 1.6
100	8%	35%	37%	6.42 ± 12.73
200	12%	37%	38%	11.14 ± 14.7
300	14%	38%	40%	29.71 ± 43.17
400	16%	39%	40%	105.2 ± 147.2
500	17%	40%	41%	258.4 ± 195.6
1000	19%	36%	41%	1727.48 ± 1129.6

5.7 Métodos de Solución

Dada la existencia de los problemas con una complejidad computacional ***NP***, surgen diferentes métodos de solución entre los cuales se encuentran los métodos exactos y los métodos aproximados; estos últimos son utilizados para los problemas ***NP-HARD*** debido a que con las características de estos problemas, el tiempo computacional utilizado para obtener la solución óptima es muy elevado, pero gracias a los métodos aproximados, se logra establecer una respuesta cercana a la óptima en un tiempo razonable. En este grupo de métodos se encuentran las Heurísticas y las Metaheurísticas.

5.7.1 Métodos Exactos. Hillier & Lieberman (2010). Los métodos son aquellos capaces de ofrecer la respuesta óptima de un problema determinado en un tiempo determinado. Existe una clase de problemas, denominada NP, con gran interés práctico para los cuales no se conocen algoritmos exactos con tiempos de convergencia en tiempo polinómico. Es decir, aunque existe un algoritmo que encuentra la solución exacta al problema, tardaría tanto tiempo en encontrarla que lo hace completamente inaplicable. Además, un algoritmo exacto es completamente dependiente del problema (o familia de problemas) que resuelve, de forma que cuando se cambia el problema se tiene que diseñar un nuevo algoritmo exacto y demostrar su optimalidad (p.82).

Algunos de los métodos exactos son nombrados a continuación.

5.7.1.1 Método simplex. Dantzig (como se citó en Hillier & Lieberman, 2010) desarrollo en 1947 un procedimiento con gran utilización debido a su extraordinaria eficiencia para solucionar

problemas de programación lineal. Este método parte de una solución inicial (si es posible se selecciona el origen), y examina si alguna de las aristas de la posición actual conduce a una tasa positiva de mejoramiento de la función objetivo. En caso de presentarse una mejor solución, se realiza una iteración para moverse a esta posición y se examina la tasa de mejoramiento de cada arista de la misma. El algoritmo finaliza, cuando ninguna de las aristas de la actual posición conlleva a una tasa de mejoramiento positiva, estableciendo ésta como la solución óptima.

5.7.1.2 Algoritmo de ramificación y acotamiento (Branch and Bound). Land & Doig (como se citó en Hong-Gui & Xing-Guo, 2009) desarrollaron en 1960 el primer algoritmo B&B para el problema general de programación lineal entera mixta y pura. La técnica de Ramificación y acotamiento se suele interpretar como un árbol de soluciones, donde cada rama lleva a una posible solución posterior a la actual. La característica de esta técnica con respecto a otras anteriores (y a la que debe su nombre) es que el algoritmo se encarga de detectar en qué ramificación las soluciones dadas ya no están siendo óptimas, para «acotar» esa rama del árbol y no continuar malgastando recursos y procesos en casos que se alejan de la solución óptima.

5.7.2 Métodos Heurísticos. Las heurísticas son algoritmos que encuentran soluciones de buena calidad para problemas combinatorios complejos del tipo NP. Los algoritmos heurísticos son más fáciles de implementar y encuentran buenas soluciones con esfuerzos computacionales relativamente pequeños.

Según Blum & Roli (como se citó en Velez 2007)‘‘los algoritmos heurísticos pueden clasificarse en los siguientes grupos’’ (p.11-14):

5.7.2.1 Algoritmos heurísticos constructivos. Este algoritmo, también denominado paso a paso, consiste en ir adicionando, generalmente uno a uno, componentes individuales de la solución hasta encontrar una solución (configuración) factible. El más popular de los algoritmos constructivos es el denominado algoritmo goloso (greedy).

5.7.2.2 Algoritmos Heurísticos de búsqueda local. García & Maheut (2011) afirman que básicamente estos algoritmos buscan una mejor solución alrededor del punto en el que se encuentran; comenzando en una solución que no necesariamente es aleatoria, realizan su búsqueda entre los vecinos más cercanos hasta llegar a una solución que es superior a las soluciones ofrecidas en su vecindario, esta solución es denominada óptimo local (p.78). Ahuja, Ergun, Orlin & Punnen (1999) definen que así es como existe una solución x que a medida que avanzan las iteraciones, se va moviendo alrededor del conjunto de soluciones cercanas, las cuales se denominan vecindario N(x) hasta encontrar una solución x' que es mejor que la solución x inicial. Es importante resaltar que cuanto más grande sea la vecindad o vecindario, mayor calidad tendrán las soluciones ofrecidas por el programa, pero también repercutirá en la utilización de más tiempo computacional (p.96).

Rodríguez Ortiz (2009) afirma que los algoritmos heurísticos no poseen ningún mecanismo que les permita escapar de los óptimos locales. Para solventar este problema se introducen otros algoritmos de búsqueda más inteligentes que eviten en la medida de lo posible quedar atrapados en óptimos locales. Estos algoritmos de búsqueda más inteligentes, denominados metaheurísticas, son procedimientos de alto nivel que guían a algoritmos heurísticos conocidos evitando que éstos caigan en óptimos locales (p.35).

5.7.2.3 ***Reglas de Asignación de Trabajo.*** Taillar (1990) provee una guía para la secuenciación de los trabajos que deben realizarse en sistemas productivos. Estas son de gran importancia dado que permiten generar una secuencia de los trabajos según un criterio basado en algún dato de entrada de los trabajos y buscan mejorar el desempeño de la programación de un indicador en particular, por ejemplo, minimizar la cantidad de trabajos tardíos, minimizar el tiempo de flujo medio, minimizar el atraso máximo entre otros. Dentro de las reglas de asignación más conocidas se encuentran:

- **FIFO** *(First In, First Out)*. Los trabajos son ejecutados en el mismo orden en el que llegan al centro de procesamiento. Esta regla tiene la ventaja de ser considerada como justa por los clientes lo cual es importante en sistemas de servicios.
- **SPT** *(Shortest Processing Time)*. Los trabajos con menor tiempo de procesamiento son programados primero en la secuencia de producción. Para la aplicación de esta regla se debe conocer el número de trabajos y los tiempos de procesamiento de cada uno de ellos. En general esta regla es la mejor para minimizar el flujo de trabajo y el flujo promedio de trabajo en el sistema, pero los trabajos con tiempos de procesamiento más largo podrían retrasarse de manera continua por darle prioridad a los más cortos, por lo tanto, requiere de un ajuste periódico para la realización de dichos trabajos.
- **EDD** *(Earliest Due Date)*. Los trabajos son programados según la fecha de entrega programada, por lo tanto, aquellos que están próximos a entregarse son programados primero que aquellos que tienen mayor tiempo de entrega.
- **LIFO** *(Last In, First Out).* Los trabajos son ejecutados de manera inversa al orden de llegada al centro de procesamiento, es decir, aquellos trabajos que llegan de últimos serán programados primero en la secuencia de programación. Como consecuencia, esta regla

suele elevar el número de trabajos tardíos por lo cual requiere de un ajuste periódico para programar aquellos trabajos que llegaron al inicio.

- **LPT** *(Large Processing Time)*. Los trabajos con el mayor tiempo de procesamiento son programados primero en la secuencia de producción, al igual que en el SJF, se debe conocer el número de trabajos y el tiempo de procesamiento del mismo.
- **CR** *(Critical Ratio)*. Los trabajos son programados de acuerdo con este índice, el cual es calculado como la diferencia entre la fecha de vencimiento y la fecha actual, dividida entre el número de días hábiles que quedan y posteriormente se ejecutan los trabajos con menor CR.

5.7.3 Métodos Metaheurísticos. Aarts, Lenstra, Blum, Roli & Martí (como se citó en Vélez & Montoya, 2007) piensan que debido a que la mayoría de los problemas de optimización combinatoria se clasifican como difíciles, la investigación se ha concentrado en desarrollar algoritmos de aproximación. Dentro de esta área, el término metaheurístico lo introdujo Glover (1986) al definir una clase de algoritmos de aproximación que combinan heurísticos tradicionales con estrategias eficientes de exploración del espacio de búsqueda.

Osman & Kelly (como se citó en Gallego Rendón et al. 2006) proponen la siguiente definición: Los metaheurísticos son métodos aproximados diseñados para resolver problemas de optimización combinatoria, en los que los heurísticos clásicos no son efectivos. Los metaheurísticos proporcionan un marco general para crear nuevos algoritmos híbridos, combinando diferentes conceptos derivados de la inteligencia artificial, la evolución biológica y los mecanismos estadísticos.

A continuación, se presentan las metaheurísticas que han sido utilizadas para resolver el problema de TSP y CSSP.

5.7.3.1 Búsqueda tabú (TS). Glover (como se citó en Gallego Rendon et al., 2006) la Búsqueda Tabú (Tabú Search, TS) fue desarrollada en la década de los 80 y sigue siendo utilizado para resolver problemas complejos en áreas muy variadas de la investigación operacional.

Es un procedimiento metaheurístico utilizado para gerenciar un algoritmo heurístico de búsqueda local con el propósito de evitar que el proceso quede atrapado en un óptimo local. Así TS realiza una exploración a través del espacio de configuraciones estudiando adecuadamente los óptimos locales. Ramón A et al. (2006).

Taha (2012) afirma que cuando la búsqueda se queda atrapada en un óptimo local, la búsqueda tabú (TS) selecciona el siguiente movimiento de búsqueda (posiblemente inferior) de una manera que prohíbe temporalmente, volver a examinar las soluciones anteriores. El instrumento principal para alcanzar este resultado es la *lista tabú* que impide que se repitan las soluciones encontradas antes durante un número específico de iteraciones sucesivas, llamado *periodo de tenencia* (p.416-417).

5.7.3.2 Algoritmos genéticos (AG). Es un método de búsqueda evolutivo que puede proporcionar soluciones óptimas o casi óptimas para problemas de optimización combinatoria. Gen & Cheng (como se citó en Taha, 2012). El algoritmo genético (AG) imita el proceso de evolución biológica

de ‘‘sobrevivencia del más apto’’. Cada solución factible de un problema se considera como un cromosoma codificado por un conjunto de genes. Los códigos genéticos más comunes son el binario (0,1) y el numérico (0,1, 2, ..., n).

“La idea general del AG es seleccionar dos padres a partir de una población. Los genes de los dos padres se cruzan entonces y (posiblemente) mutan para producir dos hijos. La descendencia reemplaza a los dos cromosomas más débiles (menos aptos) en la población, y el proceso de seleccionar nuevos padres se repite”. Taha (2012).

Alkriz & Mangin (como se citó en Taha (2012), desarrollaron un modelo para optimizar la ubicación de grúas torre en sitios de construcción utilizando algoritmos genéticos como un enfoque para obtener los mejores resultados. El modelo consideró cargas de trabajo equilibradas entre el grupo de grúas torre para minimizar la posibilidad de conflicto entre ellas y maximizar la eficiencia de la operación. El modelo de ubicación inicial clasificó las tareas en grupos e identificó la ubicación factible de cada grúa de acuerdo con la cercanía geométrica. Luego, los grupos de tareas se ajustan para producir cargas de trabajo suaves y conflictos mínimos durante el funcionamiento de la grúa. Finalmente, el modelo de optimización se aplica para optimizar la ubicación de la torre grúa y la ubicación del punto de suministro para minimizar el tiempo y el costo de transporte del material.

De acuerdo con Marzouk et al. (2016). A pesar de que, la técnica de optimización del algoritmo genético (GA) se utiliza para identificar: 1) diseño óptimo de grúas torre, 2) tipo de base de grúa ya sea torre grúa fija o grúa móvil, 3) longitud de riel más corta que puede proporcionar mayor

cobertura de la grúa y 4) óptimo plan de asignación de elevación de los módulos levantados. El análisis de sensibilidad se realiza para seleccionar los parámetros del algoritmo genético más efectivos. Los resultados obtenidos del modelo de optimización deben satisfacer las limitaciones de alcance y capacidad para todos los módulos elevados requeridos.

5.7.3.3 Modelo de optimización de torre grúa. El modelo T Crane _ Opt propuesto por Marzouk et al. (2016), optimiza la ubicación de un grupo de torres grúas en los sitios de construcción. La selección de la ubicación de la grúa se basa en minimizar el costo total de la torre grúa y maximizar su utilización con el fin de minimizar el tiempo de inactividad de cada torre grúa y satisfacer las restricciones operativas y las restricciones de las condiciones del sitio. Este modelo se desarrolla utilizando la técnica de optimización del algoritmo genético (GA). Según Springer Verlag & Berlin (2008), el modelo se desarrolla e implementa en un proyecto de construcción de caso real donde el proyecto consiste en dos niveles de sótanos, planta baja y tres niveles típicos con un diseño rectangular. Las torres grúas se utilizan para cubrir toda el área del edificio para ayudar a levantar elementos prefabricados y otros elementos utilizados en las actividades de construcción. El modelo GA se desarrolló en base a un procedimiento de tres pasos como se muestra en la Figura 3.

Como lo afirma Marzouk et al. (2016). El primer paso comienza estudiando la disposición y la especificación de la torre grúa necesaria para cubrir toda el área de elevación interceptando la zona de trabajo para cada plumín de grúa, y luego seleccionando la ubicación factible para el grupo de torres grúa. Por lo general, los gerentes de construcción tienden a aumentar las áreas de superposición entre el radio de trabajo de la grúa para estar en el lado seguro en la fase de construcción, incluso si esta solución es más costosa. El segundo paso comienza generando la

población inicial, donde los operadores de GA se aplican a los cromosomas seleccionados generando mejores generaciones donde cada cromosoma de la nueva generación se prueba contra la función de aptitud y los mejores cromosomas se seleccionan y almacenan en el grupo de apareamiento para la nueva generación. El tercer paso consiste en la condición de terminación que especifica cuándo el modelo terminará el proceso de iteración. El modelo T Crane _ Opt termina cuando no hay más mejoras en las próximas generaciones, es decir, hasta que termine de explorar la última generación se termina el proceso, siendo esto el criterio de parada.

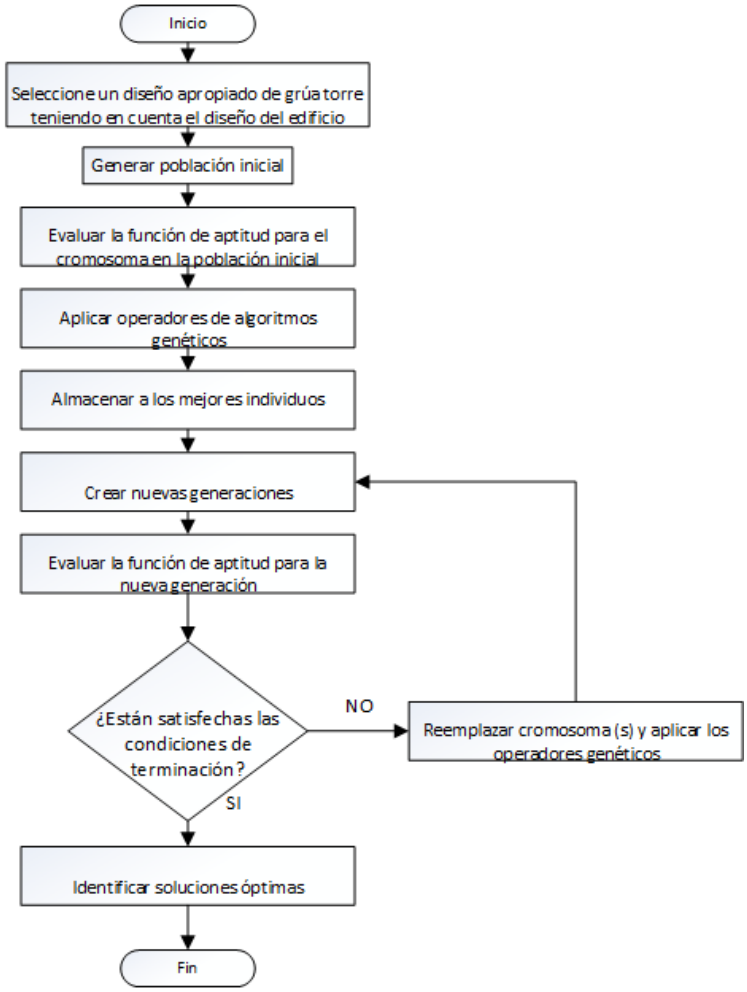

Figura 3. Diagrama de flujo del procedimiento del modelo de algoritmo genético. Adaptado de Mohamed Marzouk, Ahmed Abubakr, (2016). Soporte de decisión para la selección de grúas torre con modelos de información de construcción y algoritmos genéticos.

5.7.3.4 Algoritmo de abeja (Bee Algorithm, BA). Se han desarrollado algunos modelos para modelar los comportamientos inteligentes de los enjambres de abejas melíferas y se los aplica para resolver problemas de tipo combinatorio. Pham (como lo cito Lien et al. (2014) presentó un algoritmo de abeja original (Bee Algorithm, BA) y lo aplicó a dos problemas de optimización funcional estándar. Los resultados demostraron que BA es capaz de encontrar soluciones muy cercanas al óptimo, lo que demuestra que BA generalmente superó a GA. Sin embargo, mientras BA ofrece el potencial de realizar búsquedas globales y utiliza un mecanismo más simple en comparación con GA, su dependencia de la búsqueda aleatoria lo hace relativamente débil en las actividades de búsqueda local y no registra las experiencias de búsqueda anteriores durante el proceso de búsqueda de optimización.

Para mejorar BA y PSO, Lien et al. (2014) propusieron un algoritmo de enjambre híbrido llamado algoritmo de partícula de abeja (PBA) que imita un comportamiento inteligente particular de enjambres de aves y abejas e integra sus ventajas. El objetivo de este estudio es formular el problema del diseño la torre grúa (TCL) para un estudio de caso hipotético propuesto que implica ubicar grúas torre, puntos de suministro y demanda de materiales asociados en un programa lineal entera mixta para minimizar el costo total de operación.

5.7.3.5 Optimización de enjambre de partículas (Particle swarm optimization, PSO). La metaheurística de Optimización de Enjambre de Partículas o PSO por sus siglas en inglés (Particle Swarm Optimization), fue desarrollada por Kennedy & Eberhart (como se citó Lima & Barán, 2006) está inspirada en el comportamiento social observado en grupos de individuos tales como parvadas de pájaros, enjambres de insectos y bancos de peces. Tal comportamiento social se basa

en la transmisión del suceso de cada individuo a los demás individuos del grupo, lo cual resulta en un proceso cinegético que permite a los individuos satisfacer de la mejor manera posible sus necesidades más inmediatas, tales como la localización de alimentos o de un lugar de cobijo.

La metaheurística PSO ha mostrado ser muy eficiente para resolver problemas de optimización de un sólo objetivo con rápidas tasas de convergencia, haciendo atractiva la idea de su aplicación en la resolución problemas de optimización de múltiples objetivos. Básicamente, PSO consiste en un algoritmo iterativo basado en una población de individuos denominada enjambre, en la que cada individuo, llamado partícula, se dice que sobrevuela el espacio de decisión en busca de soluciones óptimas.

De acuerdo con Lien et al. (2014). La optimización del enjambre de partículas (PSO), que se ha vuelto bastante popular para muchos investigadores recientemente, modela el comportamiento social de las aves. El PSO se utiliza potencialmente en la búsqueda local y registra las experiencias pasadas de búsqueda durante el proceso de búsqueda de optimización. Sin embargo, converge temprano en problemas altamente discretos.

5.7.3.6 Algoritmo de partícula de abeja (Particle Bee Algorithm, PBA). El algoritmo de partícula de abeja (Particle Bee Algorithm, PBA) propuesto por Lien et al. (2014) se basa en los comportamientos inteligentes de los enjambres de aves y abejas melíferas. Basado en la cooperación entre las abejas (BA) y las aves (PSO), el PBA mejora la búsqueda de vecindarios de BA utilizando la búsqueda de PSO. Por lo tanto, PBA no emplea ninguna abeja recluta buscando alrededor de posiciones "élite" o "mejores" (como lo hace BA). En su lugar, se utiliza una búsqueda

de PSO para todas las élites y las mejores abejas. En otras palabras, después de la búsqueda de PSO, el número de abejas "élite", "mejor" y "aleatorias" es igual al número de abejas exploradoras.

En PBA, la colonia de abejas particuladas contiene cuatro grupos: (1) número de abejas exploradoras (n), (2) número de sitios de élite seleccionados de los sitios visitados (e), (3) número de los mejores sitios de n sitios visitados (b), y (4) número de abejas reclutadas para los otros sitios visitados (r). La primera mitad de la colonia de abejas está compuesta por abejas de élite, y la segunda mitad incluye las mejores y aleatorias abejas. La colonia de abejas en partículas contiene dos parámetros, es decir, el número de iteraciones para las abejas de elite por PSO (Pelite) y el número de iteraciones para las mejores abejas por PSO (Pbest).

El algoritmo de abejas de partículas (PBA) se utilizó para optimizar la ubicación de la torre grúa. El PBA se utilizó además para optimizar la distancia de operación y la frecuencia entre los puntos de demanda y suministro en términos de costos operativos totales en función de los requisitos de materiales en los puntos de demanda y suministro.

5.7.3.7 ***Algoritmo de Recocido Simulado (Simulated Annealing, SA).*** De acuerdo a Gallego Rendon et al. (2006), en este algoritmo se aplica una acción combinada del mecanismo de generación de alternativas y del criterio de aceptación. T_k Denota el valor del parámetro de control (temperatura) y N_k el número de alternativas generadas en la k - ésima iteración del algoritmo.

Inicialmente cuando T es grande, se aceptan empeoramientos grandes de la función objetivo; cuando decrece, solamente se aceptan pequeños deterioros y finalmente, cuando tiende a cero, sólo

se aceptan mejoramientos de la función objetivo. Esta característica hace que el algoritmo SA sea diferente a los algoritmos de búsqueda local.

A partir del estado i con costo $f(i)$ se genera el estado j con costo $f(j)$. El criterio de aceptación para el problema de minimización, determina si este nuevo estado es aceptado; para esto se calcula la siguiente probabilidad:

$$P_T(Acepta) = \begin{cases} 1, & Si\ f(j) \leq f(i) \\ e^{\left(\frac{f(i)-f(j)}{T}\right)}, & Si\ f(j) > f(i) \end{cases}$$

La estrategia seguida en SA es la de iniciar con una temperatura alta, lo cual equivale a una alta probabilidad de aceptar soluciones peores; y posteriormente, disminuir la temperatura, disminuyendo también la posibilidad de aceptar soluciones de peor calidad que la actual.

5.8 Diagrama de Gantt

Los diagramas de barras o "gráficos de Gantt" fueron concebidos por el Ingeniero Norteamericano Henry L. Gantt (como se citó en Hinojosa, A, 2003). Gantt procuro resolver el problema de la programación de actividades, es decir, su distribución conforme a un calendario, de manera tal que se pudiese visualizar el periodo de duración de cada actividad, sus fechas de iniciación y terminación e igualmente el tiempo total requerido para la ejecución de un trabajo Figura. 5. El instrumento que desarrolló permite también que se siga el curso de cada actividad, al proporcionar

información del porcentaje ejecutado de cada una de ellas, así como el grado de adelanto o atraso con respecto al plazo previsto.

El diagrama de Gantt consiste en una representación gráfica sobre dos ejes; en el eje vertical se disponen las tareas del proyecto y en el eje horizontal se representa el tiempo Figura 4.

Características

- Cada actividad se representa mediante un bloque rectangular cuya longitud indica su duración; la altura carece de significado.
- La posición de cada bloque en el diagrama indica los instantes de inicio y finalización de las tareas a que corresponden.
- Los bloques correspondientes a tareas del camino crítico acostumbran a rellenarse en otro color.

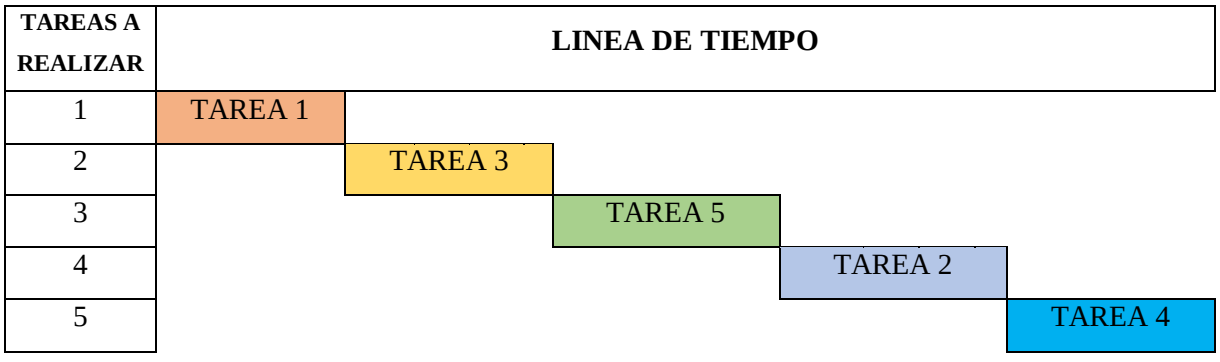

Figura 4. Diagrama de Gantt.

6. Descripción del Algoritmo de Recocido Simulado (SA)

Kirkpatrick & Cerni (como se citó en Metropolis & Rosenbluth 1953) establecieron una analogía entre el proceso de recocido del sistema físico (Termodinámica) y el reto de resolver problemas de optimización combinatoria de gran escala. Desde entonces el recocido simulado ha sido utilizado para resolver en forma exitosa una amplia variedad de problemas de optimización combinatoria, convirtiéndose en una metaheurística clásica.

El uso del recocido simulado en optimización combinatoria se basa en establecer analogías entre el sistema físico y el problema de optimización. En la tabla 3 se muestran estas analogías.

Tabla 3.

Analogía entre el sistema físico y el problema de optimización

Sistema físico (termodinámica)	**Problema de optimización**
Estados del sistema	Soluciones factibles
Energía	Función de costo (función objetivo)
Cambio de estado	Solución vecina
Temperatura	Parámetro de control T
Estado estable	Solución óptima

El método recocido simulado transpone el proceso de recocido a la solución de un problema de optimización, la función objetivo del problema, similar a la energía del material, es minimizada

con la ayuda de una temperatura ficticia, la cual es un parámetro de control del algoritmo. Este parámetro debe tener el mismo efecto que la temperatura del sistema físico: conducir hacia el estado óptimo. Si la temperatura es disminuida gradualmente y de manera controlada se puede alcanzar el mínimo global, si es disminuida abruptamente se puede llegar a un mínimo local.

Para implementar este algoritmo el primer paso es determinar cómo se representan las soluciones y expresar la función de costo que represente adecuadamente el costo de las soluciones Figura 5.

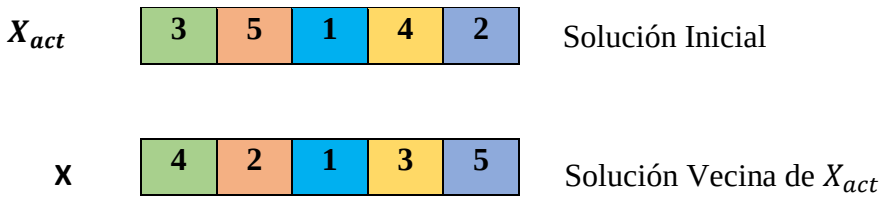

Figura 5. Representación del problema

El algoritmo de recocido simulado parte de una solución factible $\boldsymbol{X}_{act}$ que puede ser generada en forma aleatoria y se evalúa la función de costo (función objetivo) para esa solución, llámese a este valor f ($\boldsymbol{X}_{act}$). Se comienza con una temperatura ficticia alta y se genera una nueva solución X que corresponde a una solución vecina de $\boldsymbol{X}_{act}$, para la cual se calcula el valor asociado de la función de costo f (X). Además, debe calcularse la diferencia ΔE en la función de costo entre ambas soluciones:

$$\Delta E = f(X) - f(\boldsymbol{X}_{act}) \qquad (6)$$

Si ΔE < 0 el costo de la nueva solución X es menor al costo de la solución actual $\boldsymbol{X}_{act}$, la nueva solución es aceptada, es decir, que una solución de menor costo siempre se acepta. En caso contrario, la nueva solución es aceptada con una probabilidad $P(\Delta E)$:

$$P(\Delta E): e^{-(\frac{\Delta E}{t})} \qquad (7)$$

En la práctica se selecciona un número aleatorio entre 0 y 1, y si este número es menor que $P(\Delta E)$ la nueva solución se acepta.

Aceptar una solución de menor calidad permite salir de un posible mínimo local y explorar otras áreas del espacio de soluciones. Como la simulación comienza con una temperatura alta, $P(\Delta E)$ es cercana a 1 y por lo tanto una nueva solución con un costo mayor tiene una alta probabilidad de ser aceptada. La probabilidad de aceptar una solución peor va disminuyendo a medida que la temperatura decrece.

Para cada nivel de temperatura, el sistema debe alcanzar un equilibrio, es decir, un número de nuevas soluciones debe ser ensayado antes de que la temperatura sea reducida. Se puede demostrar que el algoritmo encontrará, bajo ciertas condiciones el mínimo global y no se estancará en un mínimo local (Moins, 2002). El diagrama de flujo corresponde al algoritmo básico de recocido simulado para minimización se presenta en la figura 6.

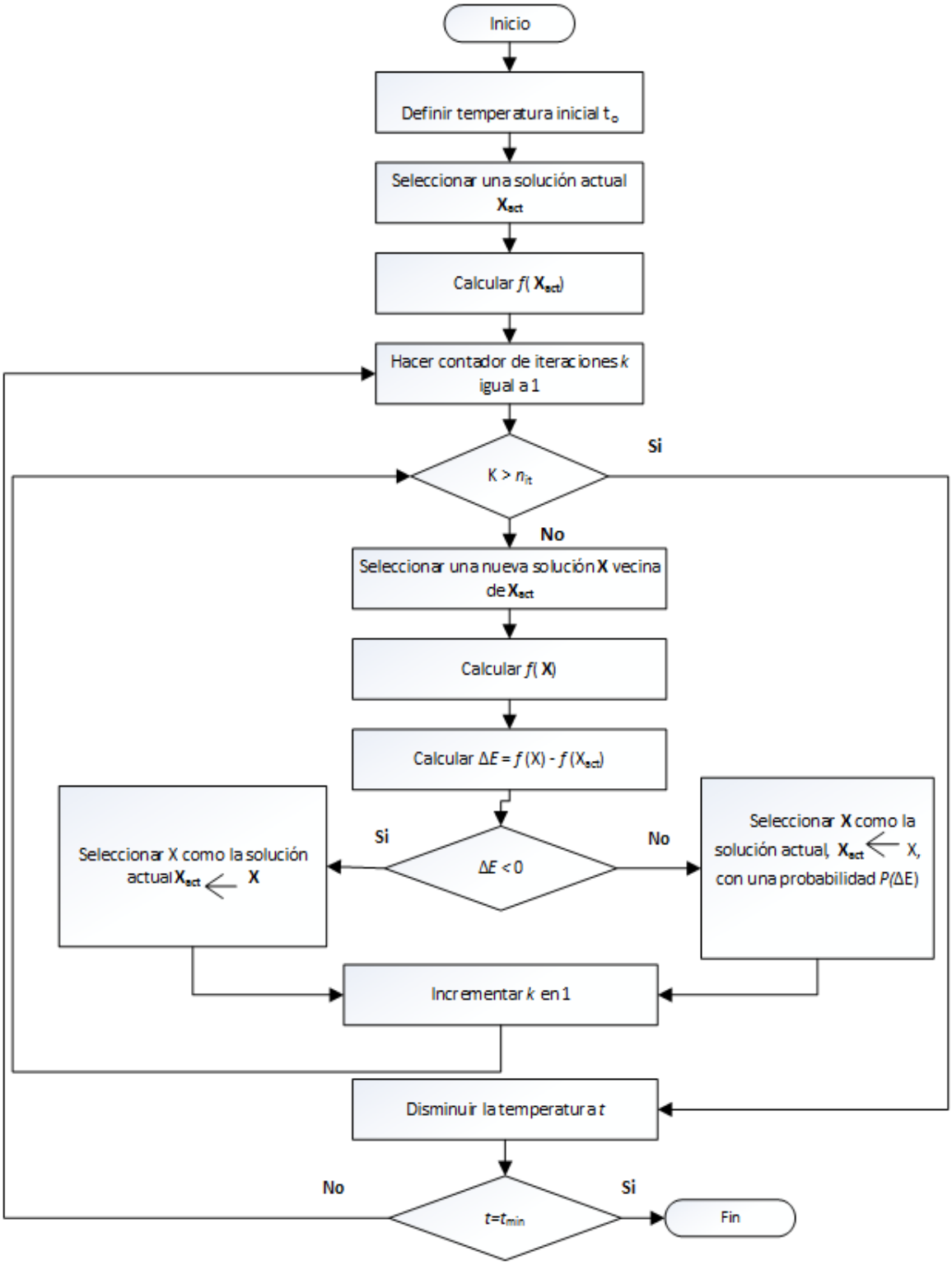

Figura 6. Pasos básicos del Recocido Simulado. Adaptado de María Alejandra Quintero M. Universidad de los Andes, (2009). Métodos heurísticos para la planificación y el manejo forestal.

El bucle interno genera una nueva solución en el entorno de la solución actual y la acepta si es mejor, en el caso de que esta nueva solución sea de menor calidad se acepta con una probabilidad P (ΔE). Este proceso se repite n veces. Cuando este ciclo iterativo se completa, la temperatura se disminuye y comienza nuevamente el bucle interno, repitiendo el proceso de creación, evaluación y posible aceptación de soluciones vecinas. Cuando la temperatura es lo suficientemente baja (t = t min) el algoritmo finaliza y debe indicar cuál fue la mejor solución obtenida.

Según Metropolis et al. (1953) para poder implementar el algoritmo de recocido simulado para resolver un problema concreto es necesario tomar ciertas decisiones referentes a la definición de parámetros (temperatura inicial t_o, número de iteraciones en cada nivel de temperatura n, temperatura mínima t min), la tasa de disminución de la temperatura, el mecanismo para generar soluciones vecinas, entre otras. Estas decisiones se pueden clasificar en decisiones genéricas y específicas.

6.1 Decisiones Genéricas

Las decisiones genéricas están relacionadas con los parámetros que rigen el programa de enfriamiento, incluyendo la temperatura inicial, la forma en que se disminuye la temperatura, número de iteraciones en cada nivel de temperatura y el criterio de parada del algoritmo (temperatura mínima). En ausencia de resultados teóricos generales, es necesario hacer ajustes empíricos de estos parámetros y para ciertos problemas, la tarea es complicada debido a la gran sensibilidad de los resultados al valor de dichos parámetros. En las siguientes secciones se presentan algunas consideraciones generales que pueden orientar las decisiones genéricas.

6.1.1 Temperatura Inicial. El algoritmo debe comenzar con una temperatura alta que permita que muchos movimientos o soluciones vecinas sean aceptados. En la práctica, se requiere conocer el valor de la función de costo para las soluciones vecinas, por ejemplo, si el mayor incremento en la función objetivo (ΔE) entre soluciones vecinas es conocido, sería posible calcular un valor t_o que aceptase un movimiento con cierta probabilidad usando la ecuación (7). En la ausencia de tal conocimiento, se puede seleccionar una temperatura que parezca ser un valor alto, ejecutar el algoritmo para un tiempo corto y observar la tasa de aceptación. Si esta tasa es ‘‘convenientemente alta’’, la temperatura con la cual se experimentó puede ser usada como el valor inicial t_o. El significado de ‘‘convenientemente alta’’ varía de una situación a otra, pero en muchos casos una tasa de aceptación entre 40% y 60% parece dar buenos resultados (Reeves, 1996; Dowsland & Díaz, 2003).

6.1.2 Disminución de Temperatura. Se refiere a la forma de la curva de enfriamiento que se utiliza en el recocido simulado, la cual determina la velocidad de disminución de la temperatura a medida que avanzan las iteraciones del algoritmo.

Existen diferentes maneras de abordar el decrecimiento de la temperatura, una de las más utilizadas debido a su simplicidad y a los buenos resultados que ha dado en numerosas aplicaciones es la forma exponencial o geométrica:

$$t_{k+1} = \alpha\, t_k \qquad (8)$$

Donde t_{k+1} es la temperatura en la iteración k+1, t_k es la temperatura en la iteración k y α es una constante cercana a 1, escogida por lo general en el rango de 0.9 a o.99.

Otro método bastante utilizado es el propuesto por Lundy y Mess (1986).

$$t_{k+1} = \frac{t_k}{1+\beta * t_k} \qquad (9)$$

Siendo β una constante cuyo valor está cerca de 0.

Se pueden utilizar otras formas funcionales en el programa de enfriamiento, sin embargo, no hay en la literatura recomendaciones concisas acerca de cuál es la mejor, puede depender del problema y en ese caso debe decidirse por experimentación.

6.1.3 Número de Iteraciones. Según (Dowsland et al. 2003). El recocido simulado ejecuta cierto número de iteraciones en cada nivel de temperatura para alcanzar el equilibrio, una vez alcanzado este estado la temperatura se reduce y el proceso se repite. Un esquema obvio es mantener un número de iteraciones constante en cada temperatura o alternativamente, se puede variar según desciende la temperatura, dedicando suficiente tiempo de búsqueda a temperaturas bajas para garantizar que se visita el óptimo local, es decir que conforme disminuye la temperatura se aumenta el número de repeticiones.

El número de iteraciones en cada temperatura está relacionado con el tamaño del problema. De acuerdo a Nareyek (2001) el número de iteraciones puede ser calculado de la siguiente manera:

$$n = Factor\ x\ |N| \qquad (10)$$

Donde $|N|$ es el tamaño del entorno actual y *Factor* es un parámetro adecuado.

Dréo (2006) sugieren que se puede cambiar a otro nivel de temperatura inferior cuando ocurra alguna de las siguientes condiciones:

12 *x G* movimientos son aceptados

100 *x G* movimientos evaluados

Donde G indica los grados de libertad (o parámetros del problema)

Otro enfoque relacionado al número de iteraciones consiste en reducir la temperatura una pequeña cantidad después de cada movimiento, es decir que el número de iteraciones en cada temperatura es igual a 1. Este es menos complicado y es el más utilizado en la práctica (Reeves, 1996).

6.1.4 Temperatura Final. Teóricamente la temperatura debería reducirse hasta 0, pero en la práctica no es necesario debido a que la búsqueda converge por lo general a su óptimo local final antes de llegar a ese valor nulo de la temperatura. Además, el algoritmo puede hacerse demasiado largo y los tiempos de computación pueden ser muy grandes, especialmente si se utiliza una tasa de decrecimiento geométrica. Hasta cierto grado la temperatura final puede depender del problema específico, y en el caso de la temperatura inicial, se puede hacer una estimación a partir de establecer la probabilidad de aceptación que se desea en la etapa final del algoritmo.

Un criterio diferente es detener la búsqueda cuando se haya producido un número determinado de iteraciones sin alguna aceptación. Así, por ejemplo, Dréo (2006) sugieren que se debe terminar

el algoritmo después de tres etapas sucesivas de temperatura sin que haya habido alguna aceptación.

6.2 Decisiones Especificas

Este conjunto de decisiones está relacionadas al problema específico que se desea resolver. Se refieren principalmente a la función de costo, la definición de la estructura de entornos y el espacio de soluciones.

6.2.1 Función de Costo. Esta función debe medir la calidad de una solución y se define de tal manera que represente el problema que se está tratando de resolver. En los casos de optimización restringida, además de la función objetivo se incluyen en la función de costo las restricciones con ciertos criterios. Las restricciones ‘‘fuertes’’ o ‘‘duras’’ son consideradas en la función de costo con un criterio alto de tal manera que, una solución que viole estas restricciones tendría un valor de la función de costo mayor que una solución factible. Las restricciones ‘‘débiles’’ o ‘‘suaves’’ dependiendo de su importancia pueden incluirse en la función de costo con un criterio asociado.

Un aspecto que es importante tomar en cuenta relacionado con la función de costo es el cálculo de su valor para una solución determinada. En cada iteración del algoritmo cuando se halla una nueva solución es necesario calcular el cambio de la función de costo con respecto a la solución anterior. Es posible disminuir el tiempo de computación si este cambio se calcula usando información relativa a qué parte de la solución ha cambiado, sin proceder a evaluarla totalmente sin tener en cuenta la información previamente disponible (Dowsland et al. 2003)

6.2.2 Estructura de Vecindario. La decisión más importante en un problema de recocido simulado es la definición de la estructura de vecindario, o lo que es igual, establecer cómo pasar de una solución a otra solución vecina. Algunos de los primeros desarrollos teóricos se basaban en entornos simétricos, es decir, si se pasa de una solución i a una solución j entonces debe ser posible moverse de la solución j a la solución i (Gökçe, (2007). "Resultados teóricos demuestran que es suficiente con exigir que cualquier solución pueda alcanzarse desde cualquier otra a través de una serie de movimientos válidos" (Reeves, (1996), Dowsland et al., 2003). Este requerimiento se denomina condición de ''alcanzabilidad''.

De esta manera, al momento de trabajar con un problema determinado es necesario asegurarse de que esta condición se cumpla.

6.2.3 Espacio de Soluciones. Posee una topología que se origina en el concepto de proximidad entre dos soluciones (estructura de entornos). Existe un costo asociado a cada solución, de tal forma que el espacio de soluciones está caracterizado por un ''paisaje de costo''. La dificultad de un problema de optimización radica en que este paisaje comprende un gran número de valles de profundidad variable y que corresponden a mínimos locales. La forma de este paisaje depende bastante de la función de costo y de la estructura de entornos.

7. Programación de las Actividades en las Grúas de construcción mediante un Método Metaheurístico para Minimizar los Costos de Operación.

Se realizó una revisión de los modelos propuestos en la literatura para el problema del Agente Viajero TSP para representar el problema CSSP debido a su similitud y ninguno se ajusta ya que en este problema se plantean diferentes condiciones para dar cumplimiento al objetivo mencionado anteriormente.

7.1 Descripción del Problema

Según Zavichi el Problema de Secuencia de Servicio de Grúa (CSSP) se puede formular como el problema del Agente Viajero (TSP), el cual implica un nivel más de decisión debido a que el gancho de la grúa debe cumplir las tareas (comenzando desde un punto de suministro y terminando en un punto de demanda) y el tiempo de viaje asociado con cada tarea es la distancia entre el punto de suministro y el punto de demanda. El problema bajo consideración tiene una torre grúa y n tareas. De acuerdo con este enfoque, se consideran los tiempos de carga y de descarga del material, es decir, el gancho de la grúa debe cumplir las tareas teniendo en cuenta que, para la primera tarea, el tiempo de desplazamiento es desde el punto de origen donde se encuentre la grúa hasta el punto de suministro TOS, donde realiza un tiempo de servicio que comprende: el tiempo de carga del material TC, el tiempo de desplazamiento desde el punto de suministro hasta el punto de demanda TSD y el tiempo de descarga del material TD.

Cuando la grúa termina de cumplir la primera tarea, se toma ese punto de demanda como punto de inicio para realizar el desplazamiento de la siguiente tarea hasta el punto de suministro TDS y así sucesivamente hasta cumplir con todas las tareas programadas en el horizonte de tiempo planificado. Además, se deben tener en cuenta los atrasos y las tareas que se encuentren en la ruta crítica del proyecto.

En la definición del problema se tienen en cuenta las siguientes consideraciones:

- La grúa no puede ir de suministro a suministro ni de demanda a demanda
- Una tarea no puede comenzar a ejecutarse sin haber terminado la tarea anterior.
- El tiempo de finalización de la tarea anterior es igual al tiempo de inicio de la siguiente tarea.
- Todas las tareas tienen un tiempo programado de finalización.
- Una tarea es considerada atrasada cuando su tiempo de ejecución es mayor al tiempo de programación.
- Todas las tareas deben ser asignadas así se encuentren atrasadas.
- Una tarea incurre en un porcentaje de penalización si se encuentra atrasada y a su vez se encuentra en la ruta crítica del proyecto.

7.2 Modelado del tiempo de transporte

Como lo expresa Zavichi (2014) las ubicaciones cartesianas (x, y) indican la posición de los puntos de suministros (m) y de los puntos de demanda (n), Figura 7, estas ubicaciones se usan para calcular el ángulo (θ) usando la ecuación (1):

$$\Theta = \tan^{-1}\frac{y}{x} \qquad (1)$$

Donde se interpreta como la tangente inversa de dos argumentos que toma en cuenta los signos de x y y para determinar el cuadrante en el que theta (Θ) se encuentra.

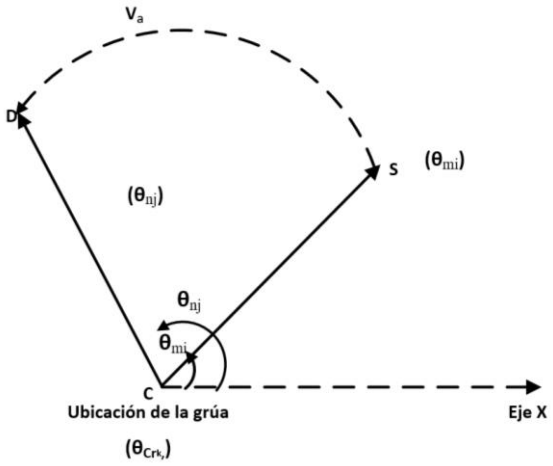

Figura 7. Coordenadas de la grúa. Adaptado de Amir Zavichi, Kaveh Madani, Petros Xanthopoulos, Amr A. Oloufa, (2014). Operaciones de grúas mejoradas en la construcción utilizando la optimización de tarea del servicio.

La velocidad angular Va (rpm) se puede basar en las especificaciones de fabricación de la grúa. Los componentes angulares Ta del tiempo de recorrido del gancho entre las ubicaciones de los puntos de suministro (m) y los puntos de demanda (n) se calculan usando la ecuación (2) respectivamente:

$$\mathrm{Ta} = \frac{|\theta s - \theta_d|}{\mathrm{Va}} \qquad (2)$$

Usando la ecuación (2) se construyen las matrices de tiempo de desplazamiento de ubicación de la grúa, las cuales consisten en ir del punto de suministro (m) al punto de demanda (n) TSD, del punto de demanda (n) al punto de suministro (m) TDS y del punto de origen a cualquier punto de suministro (m) TOS.

Como parámetros de entrada se tienen los tiempos de carga de las tareas en los puntos de suministro TC y los tiempos de descarga de las tareas en los puntos de demanda TD, se generan aleatoriamente con una distribución de probabilidad uniforme y representan el 30% del tiempo de duración total de cada tarea. Otro parámetro de entrada son los pares de solicitudes PS el cual consiste en saber cuál demanda (n) hace la solicitud y a cuál suministro (m) debe recoger el material.

Con las matrices de tiempo de desplazamiento de ubicación y los parámetros de entrada, se calcula la matriz de tiempo de operación utilizando las siguientes ecuaciones:

$$\mathrm{W_o} = \mathrm{TOS_m} + \mathrm{TC_m} + \mathrm{TSD_{m,n}}\,(PS_{(m,1)}, PS_{(n,2)}) + \mathrm{TD_n} \qquad (3)$$

$$\mathrm{W} = \mathrm{TDS_{n,m}} + \mathrm{TC_m} + \mathrm{TSD_{m,n}}\,(PS_{(m,1)}, PS_{(n,2)}) + \mathrm{TD_n} \qquad (4)$$

La ecuación (3) es específica para la primera tarea a realizar la cual inicia desde su punto de origen hacia un punto de suministro (m) más un tiempo de carga del material TC, luego tiene un tiempo de desplazamiento desde el punto de suministro (m) hacia un punto de demanda (n) TSD

teniendo en cuanta los pares de solicitudes TS mas un tiempo de descarga del material TD. El punto de demanda (n) en donde quedaría el gancho al finalizar la tarea sería el punto de inicio de la siguiente tarea.

La ecuación (4) comenzaría a realizar su tarea desde el punto de demanda (n) en donde finalizo la tarea anterior hacia un punto de suministro (m) más un tiempo de carga del material TC, luego tiene un tiempo de desplazamiento desde el punto de suministro (m) hacia un punto de demanda (n) TSD teniendo en cuanta los pares de solicitudes TS más un tiempo de descarga del material TD.

La matriz de tiempo de viaje total, (Apéndice B) es la combinación de la matriz que se utiliza para la primera tarea a realizar Wo y la matriz que se utiliza para las siguientes tareas a realizar W, esta matriz se calcula usando la ecuación (5).

$$\mathrm{WT} = \mathrm{W},\ \mathrm{Wo} \qquad (5)$$

7.3 Modelo MILP para el Problema de Secuencia de Servicio de Grúa (CSSP)

Para resolver el problema en estudio se desarrolla el siguiente modelo de programación lineal entera mixta. Su formulación es propuesta por las autoras del proyecto teniendo en cuenta algunas consideraciones del modelo desarrollado por Antero & Marín (2016) para el problema de Flow Shop; el cual, al utilizar variables binarias para modelar la secuencia de las tareas, facilitan la creación de las restricciones relacionadas con los tiempos de ejecución de la secuencia.

Conjuntos

$\boldsymbol{j}$	Tareas desde el punto de origen	j= {0,1, 2, …, n}
$\boldsymbol{i}$	Tareas a programar	i= {1, 2, …, n}

Parámetros

$\boldsymbol{P_i}$ Horario Programado de la tarea i

$\boldsymbol{CA_i}$ Costo de atraso de la tarea i

$\boldsymbol{Cg}$ Costo de la grúa por unidad de tiempo

$\boldsymbol{\beta}$ Porcentaje de penalización por ser una tarea crítica

$\boldsymbol{WT_{ij}}$ Tiempo de duración de la tarea programada

$$\boldsymbol{RC_i} = \begin{cases} 1, & \text{Indica que la tarea pertenece a la ruta critica} \\ 0, & \text{De lo contrario} \end{cases}$$

7.3.1 Variables

$\boldsymbol{TF_i}$ Tiempo de finalización de la tarea i

$\boldsymbol{TA_i}$ Tiempo de atraso de la tarea i

$\boldsymbol{TAD_i}$ Tiempo de adelanto de la tarea i

$\boldsymbol{C_{max}}$ Makespan

Variable binaria

$$\boldsymbol{X_{ij}} = \begin{cases} 1, & \text{Si la tarea j se programa inmediatamente despues de la tarea i} \\ 0, & \text{De lo contrario} \end{cases}$$

7.3.2 Función de costos

$$\text{Min Z} = \left(\mathrm{C_{max}} * \ \mathrm{C_g}\right) + \ \sum_{\mathrm{i=1}}^{\mathrm{n}} \left(\mathrm{CA_i} * \ TA_i * \ (1 + \ \mathrm{RC_i} * \beta)\right)$$

7.3.3 Restricciones

$$\mathrm{TF_i} - P_i = \mathrm{TA_i} - \ \mathrm{TAD_i} \qquad \forall_i \qquad (6)$$

$$TF_i \geq \ TF_j + WT_{ij} - M * (1 - X_{ij}) \qquad \forall_i \qquad \forall_j > 0 \qquad (7)$$

$$TF_i \geq \ WT_{i,0} * X_{i,0} \qquad \forall_i \qquad (8)$$

$$\sum_{j \neq i} X_{ij} = 1 \qquad \forall_i \qquad (9)$$

$$\sum_i X_{ij} = 1 \qquad \forall_j \qquad (10)$$

$$\mathrm{C_{max}} \geq \ \mathrm{TF_i} \qquad \forall_i \qquad (11)$$

La función objetivo minimiza el costo de utilización de la grúa determinado a partir del Makespan, el costo de atraso de las tareas y las tareas que además de atrasarse se encuentren en la

ruta crítica tendrán una penalización por su impacto en la finalización del proyecto; la restricción (6) determina el tiempo de atraso o de adelanto de cada tarea. La restricción (7) indica cual es el tiempo de finalización de cada tarea según su secuencia. La restricción (8) es específica para la primera tarea. Las restricciones (9) y (10) aseguran que cada tarea debe tener una tarea predecesora y una sucesora a excepción de la última tarea que solo debe tener una tarea predecesora. Por último, en la restricción (11) se define el Makespan como el tiempo de finalización de la última tarea programada.

7.4 Implementación en GAMS

El modelo MILP es ejecutado en un computador con procesador Intel Core i5-4570 de 3,2 GHz, memoria RAM de 8 GB y un sistema operativo Windows 10 Pro de 64 bits (apéndice C).

La etapa de validación se realizó solo para instancias pequeñas debido a que los tiempos computacionales que arrojan son muy grandes, por ejemplo, se corrieron las instancias correspondientes a 10 tareas; donde se puede corroborar que se obtuvo un tiempo computacional de 16 minutos con 9 segundos y un elapsed time de 0:16:14.942 (s).

Por este motivo, la evaluación de las instancias para la programación de las actividades en la torre grúa se deben realizar mediante una metaheurística para lograr un tiempo computacional razonable; en este caso se escogió el algoritmo de recocido simulado el cual busca escapar del atrapamiento de un óptimo local utilizando una condición de probabilidad que acepta o rechaza un

movimiento inferior, si la solución mejora, se acepta, de lo contrario habrá la probabilidad de aceptarla con una probabilidad que dependa de la temperatura. Apéndice D

7.5 Ejemplo

Para validar la eficacia del modelo propuesto, se considera un ejemplo de referencia adaptado de Zavichi et al (2014) en el cual se supone que la ubicación inicial del gancho de la torre grúa está en (0,0) y se determinan las otras ubicaciones con respecto a eso. Las especificaciones técnicas de la torre grúa son: 100 Liebherr 4000 HC de carga pesada, utilizan una Va= 0,5 rpm. El diseño del sitio de construcción consiste en seis ubicaciones de puntos de suministro y ocho ubicaciones de puntos de demanda, junto con una sola torre grúa Figura. 8. La torre grúa se estableció como el origen del sistema de coordenadas por simplicidad y las coordenadas de los puntos de suministro y los puntos de demanda se generan aleatoriamente con una distribución de probabilidad uniforme.

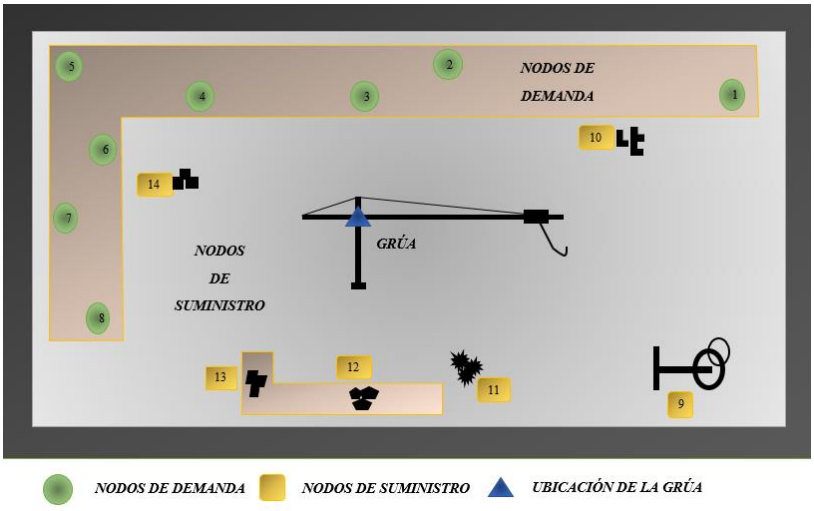

Figura 8. Plano del sitio. Adaptado de Amir Zavichi, Kaveh Madani, Petros Xanthopoulos, Amr A. Oloufa, (2014). Operaciones de grúas mejoradas en la construcción utilizando la optimización de tarea.

Debido a que en la literatura no se encuentran instancias para los parámetros de entrada presentados en el presente trabajo, estos valores se generan de forma aleatoria con una distribución de probabilidad uniforme o a criterio de la persona encargada de la programación de actividades de la torre grúa, estos parámetros cambian dependiendo del tamaño de las tareas y del número de instancias que se realicen para generar el costo de cada solución.

A continuación, se presentan estos parámetros:

- **Número de tareas (n):** Cantidad de tareas que deben ser programadas en la torre grúa dependiendo de la unidad de tiempo planificada.
- **Matriz de Tiempos de Duración (WT):** Esta matriz indica el tiempo de duración de las tareas, las cuales consisten en desplazarse desde su punto de origen hasta un punto de suministro, cargar el material, luego desplazarse hacia el punto de demanda teniendo en cuenta los pares de solicitudes y descargar el material. En el Apéndice B se puede observar el código en Matlab de cómo se realiza la matriz de tiempos de duración. En la Tabla 4 se muestra un ejemplo para n=10 tareas.

Tabla 4.

Ejemplo: Matriz de Tiempos de duración para n= 10 tareas

	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
1	7,69	0,00	10,76	13,93	11,69	9,25	14,56	6,51	13,27	12,25	9,41
2	7,49	6,64	0,00	10,39	8,21	6,55	9,93	7,94	10,24	9,72	7,23
3	8,91	7,80	4,50	0,00	5,73	7,84	7,43	10,49	5,06	5,85	8,46
4	8,42	6,88	8,61	11,26	0,00	8,10	11,32	9,01	10,46	9,33	7,05

Continuación de la Tabla 4

	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
5	8,02	5,87	9,93	10,87	9,42	0,00	11,35	5,86	11,04	9,57	8,04
6	15,09	12,20	15,68	17,92	16,82	13,91	0,00	10,59	16,98	17,54	13,16
7	7,04	6,58	9,81	9,74	8,47	5,78	10,42	0,00	8,17	9,45	6,40
8	8,26	4,91	6,13	7,22	7,06	5,47	7,73	8,81	0,00	5,85	4,87
9	7,24	6,01	3,64	5,54	3,30	6,26	6,13	10,13	3,14	0,00	6,69
10	8,18	7,02	9,88	11,16	9,06	6,32	11,77	5,61	10,18	10,23	0,00

Nota: La columna cero (0) indica el tiempo de duración de la primera tarea a programar en relación a la secuencia que se establezca, mientras que desde la columna uno (1) hasta la columna diez (10) indican los tiempos de duración dependiendo del orden en que se realicen las tareas.

- **Horario programado (P):** Cada tarea tiene programado un horario específico de entrega del material en el punto de demanda en la unidad de tiempo planificada. Este horario se genera aleatoriamente con una distribución de probabilidad uniforme dependiendo del número de tareas. (Apéndice B)
- **Costo de atraso (CA):** Cada tarea tiene un costo diferente de atraso en la entrega del material, si no cumple con el horario específico de entrega. Este costo hace referencia a los recursos destinados para la ejecución de la tarea que se encuentran ociosos. Se puede dar a criterio de la persona encargada de dicha programación, en este caso se generan de forma aleatoria con una distribución de probabilidad uniforme entre 50.000 y 150.000 unidades de tiempo. (Apéndice B)
- **Ruta crítica (RC):** Representa las tareas de mayor importancia y de las cuales dependen las demás tareas del proyecto; en caso de que algunas de ellas se atrasen, las demás que van en la ruta permanecen ociosas y la duración del proyecto será mayor. Esta penalización se representa con el valor de 1 y 0, donde 1 indica que la tarea pertenece a la ruta crítica y 0 de lo contrario. Por lo tanto, si la tarea toma el valor de 1 se incurre en un porcentaje de

penalización adicional al costo de atraso. Para este ejemplo, los valores de la ruta crítica se generan aleatoriamente con una distribución de probabilidad uniforme. (Apéndice B). En la Tabla 5 se presenta un ejemplo de la ruta crítica para n=10 tareas.

Tabla 5.

Ejemplo: Ruta Crítica para n= 10 tareas

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

- **Porcentaje de penalización (β):** Este porcentaje depende de los problemas por incumplimiento del tiempo de entrega del proyecto (penalizaciones, nombre de la empresa, pólizas, etc.); está dado para las tareas que pertenecen a la ruta crítica del proyecto y se atrasan. En este caso, el porcentaje de penalización está dado a criterio de las autoras y su valor es de β= 15%.
- **Costo de la grúa (Cg):** Este costo depende del número de tareas que se van a programar y está dado por unidad de tiempo. En este caso, este costo es dado a criterio de las autoras en

base a una entrevista estructurada con el jefe de programación de la obra en curso situada en el barrio San Francisco, el costo de la grúa es de 300.000 por unidad de tiempo.

Para evaluar el rendimiento del algoritmo de Recocido Simulado propuesto, se generan 4 tamaños de 10, 20, 50 y 100 tareas, cada tamaño tiene 5 instancias generadas a partir de los parámetros anteriores. Cada cantidad de tareas, se resuelve utilizando el algoritmo de Recocido Simulado y también las siguientes heurísticas: FIFO, SPT y EDD con el fin de validar cual es el método más eficiente a la hora de realizar la programación de las actividades en las grúas. El método de solución propuesto y las heurísticas se resuelven utilizando Matlab (apéndice H).

8. Algoritmo de Recocido Simulado propuesto al CSSP

El algoritmo desarrollado se ha basado como respuesta a recomendaciones encontradas en la literatura, en las cuales se destaca el desempeño de esta metaheurística en comparación a los métodos exactos y heurísticos para la solución de problemas combinatorios con un alto esfuerzo computacional.

Para analizar el comportamiento del Algoritmo Recocido Simulado S.A en el Problema de Secuencia de Servicio de Grúa (CSSP) para minimizar los costos de operación se realizó una ANOVA de un solo Factor. El algoritmo se desarrolló en Matlab ® versión R2017a y la ANOVA de un solo Factor se ejecutó en el Software estadístico Minitab ® 18, (Apéndice D)

8.1 Representación de la solución

La manera de representar la solución del problema CSSP influye en la eficiencia del algoritmo propuesto. En este trabajo se utiliza la representación de un vector que contiene la secuencia de ejecución de las tareas a realizar.

Para ilustrar la solución del problema mediante la representación del vector de secuenciación, se utiliza el ejemplo de la Figura 9 para n=10 tareas.

Figura 9. Vector de secuenciación de 10 tareas programadas.

De acuerdo con el vector de secuenciación, la primera tarea a ejecutar es la 5, después la tarea 10 y así sucesivamente hasta completar el total de tareas programadas.

8.2 Parámetros de entrada

Para todos los análisis de ANOVA de un solo factor, se determinó una muestra de 30 datos

8.2.1 Análisis Estadístico para determinar el Tamaño de Vecindario. Este análisis se realiza por separado tanto para los tamaños de las tareas pequeñas como para los tamaños de las tareas grandes debido a que, dependiendo del número de tareas, se requiere un tamaño de vecindario diferente.

- En primer lugar, se realiza la validación para 10 y 20 tareas a programar con los tamaños de vecindario de 50, 100 y 150, se analiza el comportamiento y se obtienen los siguientes resultados. Ver Tabla 6 y Figuras 10 y 11

Tabla 6.

Análisis de medias de los tamaños de vecindario entre 50, 100 y 150

Factor	N	Media	Desv. Est.	IC de 95%
50	30	29288725	804522	(29014943; 29562508)
100	30	28906930	739209	(28633148; 29180712)
150	30	28677257	716894	(28403474; 28951039)

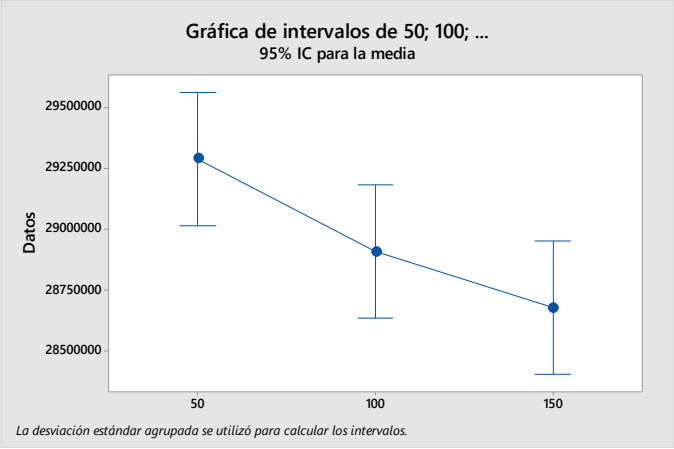

Figura 10. Gráfica de Intervalos de los vecindarios 50, 100 y 150

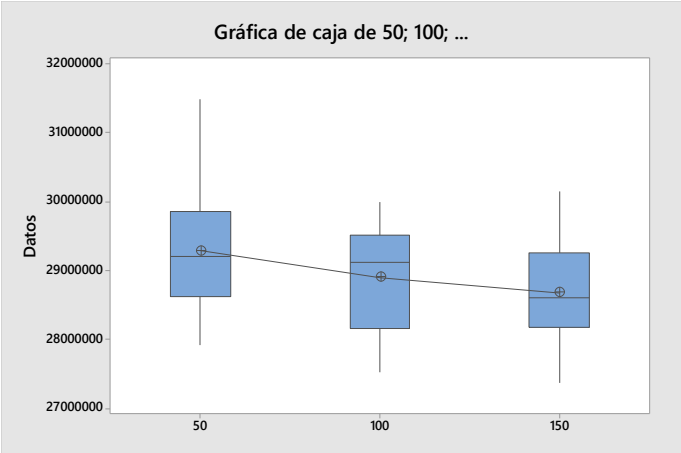

Figura 11. Gráfica de caja de los vecindarios 50, 100 y 150

Analizando la tabla de medias y las gráficas de intervalos y de caja se evidencia que, en los cambios del tamaño de vecindario entre 50, 100 y 150, se obtuvo que el tamaño de vecindario de 150 mejora los resultados en costo.

- Por último, se realiza la validación para 50 y 100 tareas a programar con los tamaños de vecindario de 150, 200 y 250, se analiza el comportamiento y se obtienen los siguientes resultados. Ver Tabla 7 y Figuras 12 y 13

Tabla 7.

Análisis de medias de los tamaños de vecindario entre 150, 200 y 250

Factor	N	Media	Desv.Est.	IC de 95%
150	30	94265115	1824453	(93718029; 94812201)
200	30	93361156	1328333	(92814070; 93908243)
250	30	93201022	1313568	(92653936; 93748109)

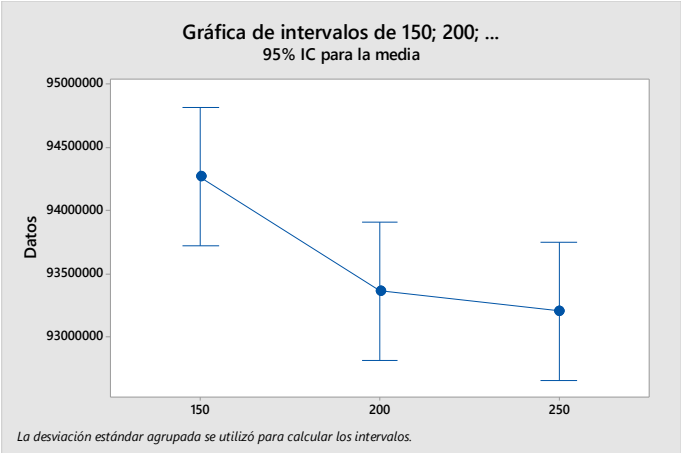

Figura 12. Gráfica de Intervalos de los vecindarios 150, 200 y 250

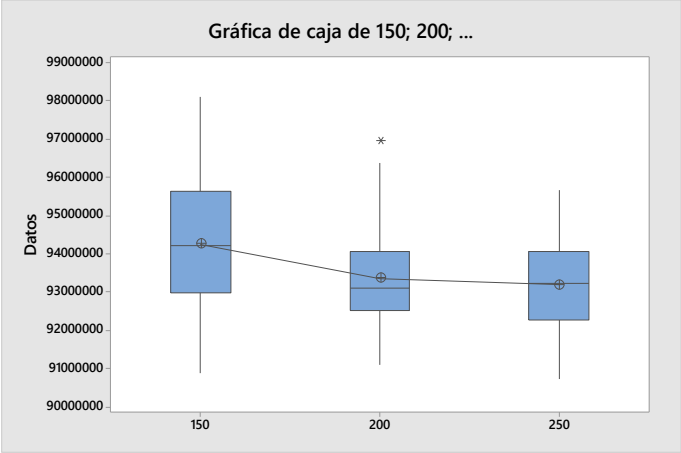

Figura 13. Gráfica de caja de los vecindarios 150, 200 y 250

Analizando la tabla de medias y las gráficas de intervalos y de caja se evidencia que, en los cambios del tamaño de vecindario entre 150, 200 y 250, se obtuvo que el tamaño de vecindario de 250 mejora los resultados en costo.

Se obtuvo el comportamiento con cada uno de estos vecindarios, se analizaron los resultados y se determinó que el vecindario que da mejores resultados paras los tamaños de 10 y 20 tareas es de tamaño 150 y para los tamaños de 50 y 100 tareas es de tamaño 250 (apéndice E).

8.2.2 Análisis Estadístico para determinar el ALPHA. El Alpha es la velocidad de enfriamiento e indica la forma de variación de la temperatura, es decir, cuanto puede ir bajando la temperatura.

La validación se realiza para los tamaños de 10, 20, 50 y 100 tareas a programar, en las cuales se estudia el comportamiento del algoritmo con los α= 0.7, 0.8 y 0.9 y se obtienen los siguientes resultados. Ver Tabla 8 y Figuras 14 y 15

Tabla 8.

Análisis de medias de los Alpha α= 0.7, 0.8 y 0.9

Factor	N	Media	Desv. Est.	IC de 95%
0,7	30	94657558	1769207	(94054805; 95260310)
0,8	30	94604885	1593599	(94002133; 95207637)
0,9	30	93027285	1614645	(92424533; 93630038)

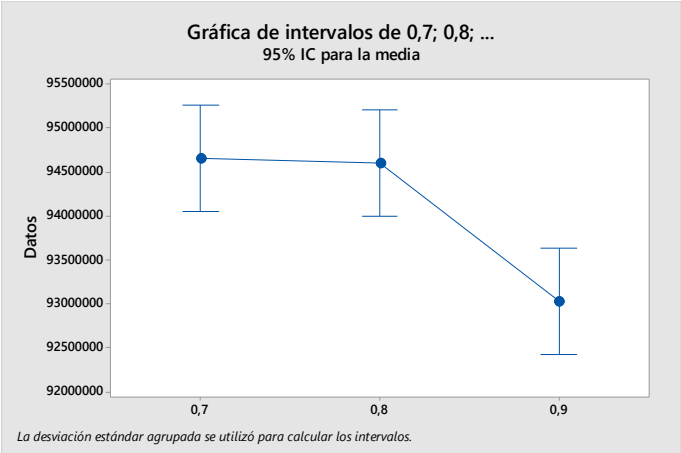

Figura 14. Gráfica de Intervalos de los Alpha α = 0.7, 0.8 y 0.9

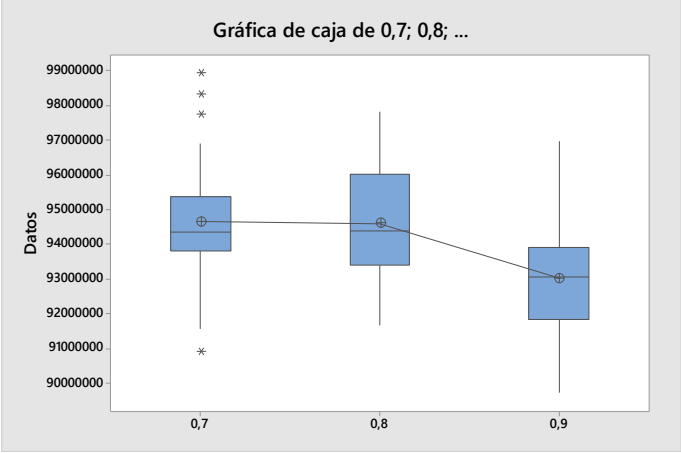

Figura 15. Gráfica de caja de los Alpha α= 0.7, 0.8 y 0.9

Analizando la tabla de medias y las gráficas de intervalos y de caja para el alpha (α), se evidencia que el mejor alpha para todos los tamaños de tareas es el α= 0.9, ya que obtuvo buenos resultados en las instancias (apéndice F).

8.2.3 Análisis Estadístico para determinar los Cambios de Temperatura. La temperatura se halló en base a la solución inicial, se determina cuantos cambios de temperatura va a realizar debido a que la temperatura va a variar dependiendo de la solución inicial.

La validación se realiza para los tamaños de 10, 20, 50 y 100 tareas a programar, en las cuales se estudia el comportamiento del algoritmo con los cambios de temperatura entre 20, 50 y 100, donde se obtienen los siguientes resultados. Ver Tabla 9 y Figuras 16 y 17

Tabla 9.

Análisis de medias de los cambios de temperatura entre 20, 50 y 100

Factor	N	Media	Desv.Est.	IC de 95%
20	30	114998703	1507387	(114436706; 115560700)
50	30	111452266	757470	(110890269; 112014263)
100	30	109814460	2085506	(109252463; 110376458)

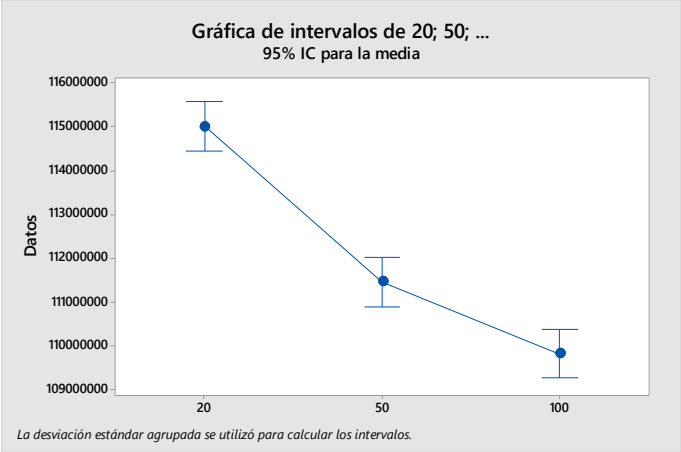

Figura 16. Gráfica de Intervalos de los cambios de temperatura entre 20, 50 y 100

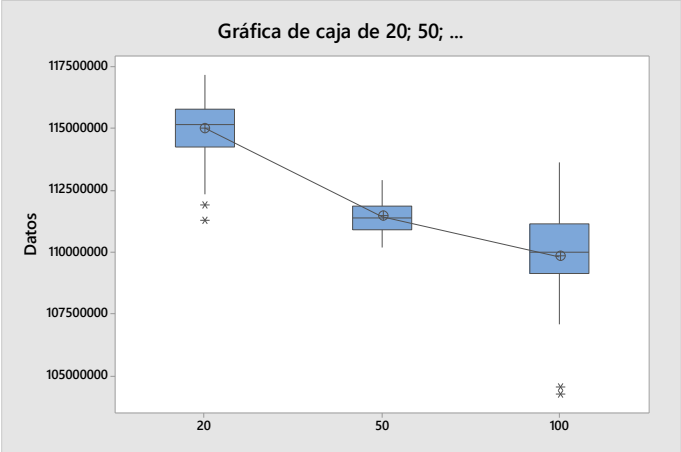

Figura 17. Gráfica de caja de los cambios de temperatura entre 20, 50 y 100

Analizando la tabla de medias y las gráficas de intervalos y de caja, se evidencia que, variando 100 veces la temperatura se obtuvo mejores resultados en la función de costo. (Apéndice G)

8.3 Recocido Simulado

- **Solución Inicial:** Se da a partir del número de tareas a programar, para esto, se genera un vector aleatorio el cual indica la secuencia de duración de las tareas. En el apéndice D se puede encontrar el código para generar la solución inicial.
- **Temperatura inicial:** Se calculó mediante la siguiente fórmula

$$e^{-\left(\frac{\alpha\ x\ f_0}{T_0}\right)} \geq 0.9 \qquad Para\ valores\ de\ \alpha \leq 0.2$$

Según Bula (2004) “La temperatura de inicio se establece para que haya una probabilidad de 0.9 de aceptar soluciones que desmejoren la solución inicial hasta en un 20% del valor inicial como lo propuso” (p.105).

- **Criterio de parada:** Es el número de iteraciones el cual consiste en el tamaño de vecindario por el número de cambios de temperatura, es decir, hasta que termine de explorar el último vecino, en el último vecindario con la temperatura más baja se termina el proceso.
- **Estructura de vecindario:** Consiste en hacer un intervalo en el vector y cambiar el orden de esas soluciones, lo cual representa cambiar las posiciones para generar un nuevo vector de secuenciación a partir de la solución inicial. Para los tamaños de 10 y 20 tareas se hace un intervalo de 70-30%, lo cual equivale generar 3 cambios de tareas para 10 y 6 cambios de tareas para 20, mientras que para los tamaños de 50 y 100 tareas se hace un intervalo de 90-10%, lo cual equivale generar 5 cambios de tareas para 50 y 10 cambios de tareas para 100. Ver el Apéndice D y Figura 18.

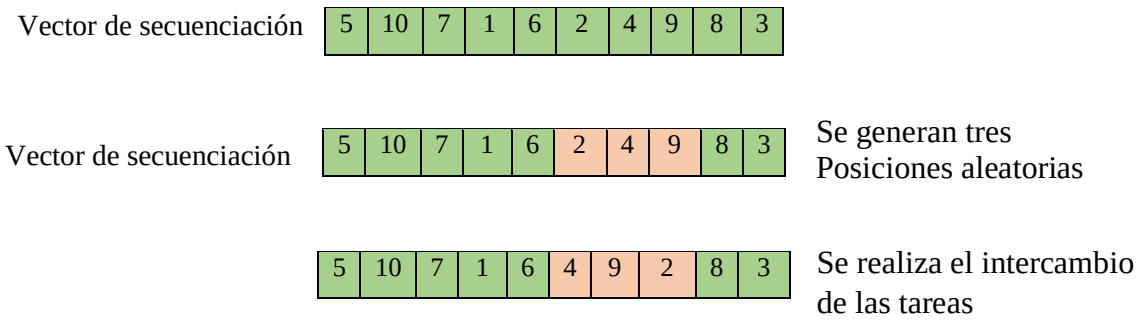

Figura 18. Ejemplo estructura de vecindarios entre tareas.

8.3.1 Diagrama de flujo de Algoritmo de Recocido Simulado.

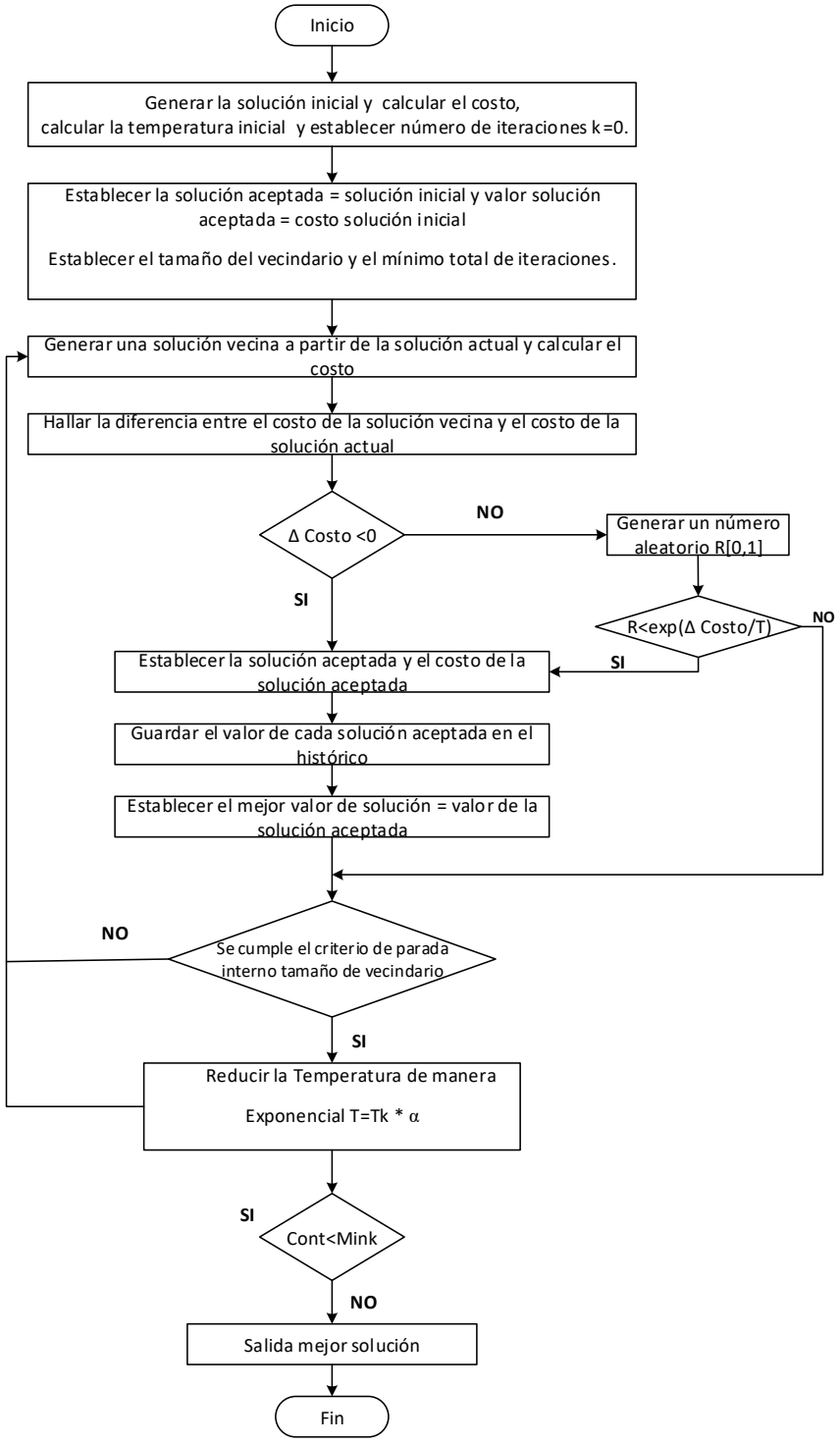

Figura 19. Diagrama de Flujo del Recocido Simulado

9. Validación del Algoritmo con las Heurísticas FIFO, EDD y SPT

Para evaluar el desempeño del algoritmo propuesto, se utilizó el tamaño de 10, 20, 50 y 100 tareas simultáneas generadas aleatoriamente y se comparan los resultados con los obtenidos en las heurísticas FIFO, EDD Y SPT para las mismas instancias. Los resultados de las heurísticas para estas instancias se encuentran en el apéndice L

Los siguientes métodos de planificación heurística fueron escogidos debido a que son los enfoques predominantes para los problemas de programación en base al estudio realizado por (Zavichi et. al 2014)

Estas heurísticas se realizaron en Matlab y se les modificó la cantidad de tareas para las instancias dadas, donde se analizó el makespan, el costo, el elapsed time (s) y el vector de secuenciación, el cual indica el orden de las tareas que se deben programar. Cabe destacar que el tiempo y costo de atraso es independiente para cada tarea (apéndice I).

9.1 Parámetros para el Algoritmo de Recocido Simulado

En la Tabla 10 se describen los parámetros tomados para resolver el Algoritmo de Recocido Simulado (apéndice K).

Tabla 10.

Parámetros del Recocido Simulado

INSTANCIAS	VALORES
Temperatura	100
Alpha	0.9
Vecindario para 10 y 20 Tareas	150
Vecindario para 50 y 100 Tareas	250
Tamaño de tareas	10, 20, 50 y 100
Instancias por cada tamaño de tareas	5

A partir de los datos de entrada del Algoritmo de Recocido Simulado se obtiene la Tabla 11 que representa los resultados de la función objetivo y el tiempo computacional dado en unidades de tiempo para los cuatro tamaños de tareas de 10, 20, 50 y 100, cada tamaño se resuelve con cinco instancias.

Tabla 11.

Resultados Función Objetivo y Tiempo Computacional del Algoritmo de Recocido Simulado

TAMAÑO DE TAREAS	INSTANCIAS	FUNCIÓN OBJETIVO	TIEMPO COMPUTACIONAL
10 TAREAS	1	$ 21.629.421	0,218
	2	$ 31.239.079	0,212
	3	$ 26.818.573	0,209
	4	$ 26.508.936	0,208
	5	$ 17.996.890	0,217
20 TAREAS	1	$ 29.192.532	0,274
	2	$ 24.769.496	0,271
	3	$ 33.424.173	0,266

Continuación de la Tabla 11.

TAMAÑO DE TAREAS	INSTANCIAS	FUNCIÓN OBJETIVO	TIEMPO COMPUTACIONAL
	4	$ 25.651.368	0,268
	5	$ 28.806.600	0,272
50 TAREAS	1	$ 47.313.135	0,747
	2	$ 58.393.802	0,745
	3	$ 56.271.889	0,765
	4	$ 63.254.327	0,765
	5	$ 57.554.627	0,795
100 TAREAS	1	$ 93.986.481	1,483
	2	$ 110.405.198	1,498
	3	$ 98.514.929	1,470
	4	$ 63.770.268	1,482
	5	$ 63.770.268	1,482

9.2 FIFO (Primeras en entrar, primeras en salir)

Esta heurística permite que el operador de la grúa procese las tareas según el pedido recibido.

La rutina en Matlab realiza las siguientes acciones para operar la heurística (apéndice J):

1. Carga las tablas de la instancia correspondiente junto con los parámetros (Tiempo de duración de las tareas, horario programado, ruta crítica, costo de atraso, costo de la grúa y el porcentaje de penalización por ser una tarea crítica).
2. Genera un vector de ceros (1 y 0) para llenar el tiempo de finalización de cada tarea
3. Genera los tiempos de la primera tarea para calcular el makespan y el costo
4. Genera los tiempos desde la segunda tarea hasta la última

5. Calcula el tiempo de finalización desde la tarea 2 hasta la n, teniendo en cuenta el tiempo de finalización de la tarea anterior y el tiempo de duración de la tarea que se esté teniendo en cuenta en el momento.
6. Calcula el makespan (el tiempo de finalización de la última tarea programada)
7. Calcula el tiempo de atraso por cada tarea.
8. Ordena de menor a mayor los tiempos de programación.
9. Genera un tiempo de atraso temporal el cual indica el tiempo de finalización de cada tarea menos el tiempo programado.
10. Calcula la función de costo.
11. Calcula el tiempo de ejecución de cada tarea.
12. Genera el vector de secuenciación para cada tamaño de tareas.

La diferencia porcentual en el costo (GAP) nos indica en cuanto mejora la solución inicial:

$$GAP = \frac{Valor\ encontrado - La\ mejor\ solución}{La\ mejor\ solución}$$

La diferencia porcentual entre el Recocido Simulado y la heurística FIFO estuvo entre:

- Instancia de 10 tareas: 9.27 y 35.56 %
- Instancia de 20 tareas: 17.54 y 45.78%
- Instancia de 50 tareas: 16.12 y 33.85%
- Instancia de 100 tareas: 17.70 y 66.10%

9.3 EDD (Fecha de vencimiento más temprana)

Esta heurística permite programar la tarea que tiene menor horario programado que aquellas tareas que tienen mayor horario programado. (Ver Apéndice J).

La rutina en Matlab realiza las siguientes acciones para operar la heurística:

1. Carga las tablas de la instancia correspondiente junto con los parámetros (Tiempo de duración de las tareas, horario programado, ruta crítica, costo de atraso, costo de la grúa y el porcentaje de penalización por ser una tarea crítica).
2. Genera un vector del horario programado desde la tarea 1 hasta la n
3. Ordena de menor a mayor el horario programado.
4. Genera una matriz del horario programado para mantener las posiciones de las tareas y que no se pierdan.
5. Ordena de menor a mayor el horario programado de la matriz para mantener las posiciones de las tareas.
6. Genera un vector de ceros (1 y 0) para llenar el tiempo de finalización de cada tarea
7. Genera los tiempos de la primera tarea para calcular el makespan y el costo
8. Genera los tiempos desde la segunda tarea hasta la última
9. Calcula el tiempo de finalización desde la tarea 2 hasta la n, teniendo en cuenta el tiempo de finalización de la tarea anterior y el tiempo de duración de la tarea que se esté teniendo en cuenta en el momento.
10. Calcula el makespan (el tiempo de finalización de la última tarea programada)
11. Calcula el tiempo de atraso por cada tarea.

12. Genera un tiempo de atraso temporal el cual indica el tiempo de finalización de cada tarea menos el tiempo programado.
13. Calcula la función de costo.
14. Calcula el tiempo de ejecución de cada tarea.
15. Genera el vector de secuenciación para cada tamaño de tareas

La diferencia porcentual en el costo (GAP) nos indica en cuanto mejora la solución inicial:

$$GAP = \frac{Valor\ encontrado - La\ mejor\ solución}{La\ mejor\ solución}$$

La diferencia porcentual entre el Recocido Simulado y la heurística EDD estuvo entre:

- Instancia de 10 tareas: 15.15 Y 30.95%
- Instancia de 20 tareas: 11.77 Y 39.13%
- Instancia de 50 tareas: 5.86 Y 48.09%
- Instancia de 100 tareas: 16.15 Y 70.48%

9.4 SPT (Tiempo de procesamiento más corto)

Esta heurística permite que las tareas con menor tiempo de duración sean programadas primero en la secuencia de producción. (Ver Apéndice J)

Esta heurística es la mejor para minimizar el flujo de trabajo y el flujo promedio de trabajo en la torre grúa, pero las tareas con tiempos de duración más larga podrían retrasarse de manera continua por darle prioridad a los más cortos, por lo tanto, requiere de un ajuste periódico para la realización de dichas tareas.

La rutina en Matlab realiza las siguientes acciones para operar la heurística:

1. Carga las tablas de la instancia correspondiente junto con los parámetros (Tiempo de duración de las tareas, horario programado, ruta crítica, costo de atraso, costo de la grúa y el porcentaje de penalización por ser una tarea crítica).
2. Genera una matriz gemela a los tiempos de duración de las tareas para calcular la solución (el vector).
3. Genera un vector de ceros (1 y 0) para llenar el tiempo de finalización de cada tarea
4. Calcula los tiempos de duración desde la tarea 1 hasta la n.
5. Busca para la primera tarea después del cero en la primera columna de la matriz de tiempos, la que tenga el menor tiempo de duración.
6. Genera un vector con todos los tiempos de duración diferentes de cero y luego se evalúa cual es el mínimo de todos los tiempos de duración.
7. Encuentra en el vector de los tiempos de duración cuál es la posición de la tarea.
8. La función find me indica en que tarea está el valor mínimo de los tiempos de duración para ejecutarse de primera, se hace cero para no volverla a tener en cuenta y así sucesivamente hasta que se programen todas las tareas. Cuando todas mis tareas sean cero, no va a seguir buscando y ese será el fin del bucle.
9. Genera un vector de ceros (1 y 0) para llenar el tiempo de finalización de cada tarea

10. Genera los tiempos de la primera tarea para calcular el makespan y el costo
11. Genera los tiempos desde la segunda tarea hasta la última
12. Calcula el tiempo de finalización desde la tarea 2 hasta la n, teniendo en cuenta el tiempo de finalización de la tarea anterior y el tiempo de duración de la tarea que se esté teniendo en cuenta en el momento.
13. Calcula el makespan (el tiempo de finalización de la última tarea programada)
14. Calcula el tiempo de atraso por cada tarea.
15. Genera un tiempo de atraso temporal el cual indica el tiempo de finalización de cada tarea menos el tiempo programado.
16. Calcula la función de costo.
17. Calcula el tiempo de ejecución de cada tarea.
18. Genera el vector de secuenciación para cada tamaño de tareas

La diferencia porcentual en el costo (GAP) nos indica en cuanto mejora la solución inicial:

$$GAP = \frac{Valor\ encontrado - La\ mejor\ solución}{La\ mejor\ solución}$$

La diferencia porcentual entre el Recocido Simulado y la heurística SPT estuvo entre:

- Instancia de 10 tareas: 5.59 y 19.34%
- Instancia de 20 tareas: 1.60 y 14.05%

- Instancia de 50 tareas: 4.65 y 11.17%
- Instancia de 100 tareas: 4.23 y 23.88%

10. Resultados Computacionales

En la tabla 12 se muestra la validación de los resultados obtenidos del Algoritmo de Recocido Simulado después de realizar 5 instancias para cada tamaño de tareas con las Heurísticas: FIFO (Primeras en entrar, primeras en salir), EDD (Fecha de vencimiento más temprana) y SPT (Tiempo de procesamiento más corto). En este caso, se puede observar que el costo del algoritmo propuesto en las 5 instancias para los tamaños de 10, 20, 50 y 100 tareas es menor frente a las heurísticas mencionadas anteriormente, lo cual validaría que la programación de las actividades en la torre grúa sería muy efectiva al aplicar el método de solución propuesto. (Ver Tabla 12) y Apéndice L

Tabla 12.

Resultados de la Función Objetivo del Algoritmo de Recocido Simulado VS Heurísticas FIFO, EDD y SPT

TAMAÑO DE TAREAS	INSTANCIAS	Algoritmo S. A	FIFO	EDD	SPT
10 TAREAS	1	$ 21.629.421	$ 25.827.739	$ 28.324.470	$ 22.932.785
	2	$ 31.239.079	$ 35.185.522	$ 35.971.431	$ 35.174.247
	3	$ 26.818.573	$ 29.305.772	$ 31.418.650	$ 28.316.714
	4	$ 26.508.936	$ 31.597.291	$ 32.131.371	$ 30.849.223
	5	$ 17.996.890	$ 24.397.284	$ 22.588.606	$ 21.478.553

Continuación de la Tabla 12.

TAMAÑO DE TAREAS	INSTANCIAS	Algoritmo S. A	FIFO	EDD	SPT
20 TAREAS	1	$ 29.192.532	$ 37.831.492	$ 34.159.704	$ 30.249.458
	2	$ 24.769.496	$ 32.167.211	$ 34.461.214	$ 25.164.906
	3	$ 33.424.173	$ 40.476.440	$ 37.443.387	$ 35.312.515
	4	$ 25.651.368	$ 30.151.798	$ 28.669.946	$ 27.368.586
	5	$ 28.806.600	$ 41.994.842	$ 37.446.778	$ 32.855.250
50 TAREAS	1	$ 47.313.135	$ 63.327.274	$ 62.611.691	$ 49.514.699
	2	$ 58.393.802	$ 68.832.250	$ 70.581.589	$ 63.407.593
	3	$ 56.271.889	$ 74.638.611	$ 83.335.435	$ 61.026.461
	4	$ 63.254.327	$ 73.454.270	$ 66.958.471	$ 67.162.832
	5	$ 57.554.627	$ 72.976.534	$ 74.323.307	$ 63.985.804
100 TAREAS	1	$ 93.986.481	$ 130.383.778	$ 133.423.786	$ 97.957.614
	2	$ 110.405.198	$ 129.952.692	$ 128.230.524	$ 118.762.609
	3	$ 98.514.929	$ 139.121.115	$ 132.948.762	$105.204.697
	4	$ 63.770.268	$ 91.590.528	$ 90.946.058	$ 66.704.625
	5	$ 63.770.268	$ 105.924.959	$ 108.713.814	$ 78.997.303

En la Tabla 13 se muestra la diferencia porcentual del costo (% GAP) de los resultados obtenidos del Algoritmo de Recocido Simulado después de realizar 5 instancias para cada tamaño de tareas con las Heurísticas: FIFO (Primeras en entrar, primeras en salir), EDD (Fecha de vencimiento más temprana) y SPT (Tiempo de procesamiento más corto).

Tabla 13.

Diferencia porcentual del costo entre el Algoritmo de Recocido Simulado VS Heurísticas FIFO, EDD y SPT

ALGORITMO RECOCIDO SIMULADO S. A		DIFERENCIA PORCENTUAL DEL COSTO (%GAP)		
TAMAÑO DE TAREAS	INSTANCIAS	FIFO	EDD	SPT
10 TAREAS	1	19,410%	30,953%	6,026%
	2	12,633%	15,149%	12,597%

Continuación de la Tabla 13.

ALGORITMO RECOCIDO SIMULADO S. A		DIFERENCIA PORCENTUAL DEL COSTO (%GAP)		
TAMAÑO DE TAREAS	**INSTANCIAS**	**FIFO**	**EDD**	**SPT**
	3	9,274%	17,152%	5,586%
	4	19,195%	21,209%	16,373%
	5	35,564%	25,514%	19,346%
	1	29,593%	17,015%	3,620%
	2	29,866%	39,128%	1,596%
20 TAREAS	3	21,099%	12,025%	5,649%
	4	17,544%	11,768%	6,694%
	5	45,782%	29,994%	14,054%
	1	33,847%	32,335%	4,653%
	2	17,876%	20,872%	8,586%
50 TAREAS	3	32,639%	48,094%	8,449%
	4	16,125%	5,856%	6,179%
	5	26,795%	29,135%	11,174%
	1	38,726%	41,961%	4,225%
	2	17,705%	16,145%	7,570%
100 TAREAS	3	41,218%	34,953%	6,790%
	4	43,626%	42,615%	4,601%
	5	66,104%	70,477%	23,878%

En la Tabla 14 se muestra el intervalo de la diferencia porcentual del costo (% GAP) de los resultados obtenidos entre el Algoritmo de Recocido Simulado y las Heurísticas. En primer lugar, se realiza el análisis de los resultados obtenidos del algoritmo propuesto frente a la heurística FIFO, donde se obtiene una diferencia porcentual del costo para cada tamaño de tareas. Para el tamaño de 10 tareas se encuentra entre un 9,274% y 35,563%, mientras que para el tamaño de 20 tareas estuvo entre 17,544% y 45,782%. Para las tareas de mayor tamaño como lo son las de 50 y 100 tareas, se encuentra que la diferencia porcentual del costo para 50 tareas está entre un 16,125% y 33,847% y para 100 tareas está entre un 17,705% y 66,103%, lo cual indica que el algoritmo

propuesto es mejor que la heurística FIFO, lo cual se evidencia en la efectividad del Algoritmo propuesto ya que minimiza los costos de operación respecto a todos los tamaños de tareas.

En segundo lugar, se realiza el análisis de los resultados obtenidos del algoritmo propuesto frente a la heurística EDD, donde se obtiene una diferencia porcentual del costo para cada tamaño de tareas. Para el tamaño de 10 tareas se encuentra entre un 15,149% y 30,953%, mientras que para el tamaño de 20 tareas estuvo entre 11,768% y 39,128%. Para las tareas de mayor tamaño como lo son las de 50 y 100 tareas, se encuentra que la diferencia porcentual del costo para 50 tareas está entre un 5,856% y 48,094% y para 100 tareas está entre un 16,145% y 70,477%, lo cual indica que el algoritmo propuesto es mejor que la heurística EDD, lo cual se evidencia en la efectividad del Algoritmo propuesto ya que minimiza los costos de operación respecto a todos los tamaños de tareas.

Por último, se realiza el análisis de los resultados obtenidos del algoritmo propuesto frente a la heurística SPT, donde se obtiene una diferencia porcentual del costo para cada tamaño de tareas. Para el tamaño de 10 tareas se encuentra entre un 5,586% y 19,346%, mientras que para el tamaño de 20 tareas estuvo entre 1,596% y 14,054%. Para las tareas de mayor tamaño como lo son las de 50 y 100 tareas, se encuentra que la diferencia porcentual del costo para 50 tareas está entre un 4,653% y 11,174% y para 100 tareas está entre un 4,225% y 23,878%, lo cual indica que el algoritmo propuesto es mejor que la heurística SPT, lo cual se evidencia en la efectividad del Algoritmo propuesto ya que minimiza los costos de operación respecto a todos los tamaños de tareas. (Ver Tabla 14) y Apéndice L

Tabla 14.

Diferencia porcentual del costo (GAP) entre el Algoritmo de Recocido Simulado VS Heurísticas FIFO, EDD y SPT.

ALGORITMO RECOCIDO SIMULADO S. A	**DIFERENCIA PORCENTUAL (% GAP)**		
TAMAÑO DE TAREAS	**FIFO**	**EDD**	**SPT**
10 TAREAS	9,274% - 35,563%	15,149% - 30,953%	5,586% - 19,346%
20 TAREAS	17,544% - 45,782%	11,768% - 39,128%	1,596% - 14,054%
50 TAREAS	16,125% - 33,847%	5,856% - 48,094%	4,653% - 11,174%
100 TAREAS	17,705% - 66,103%	16,145% - 70,477%	4,225% - 23,878%

En la tabla 15 se evidencian los tiempos computacionales promedio dados en unidades de tiempo para 4 tamaños de 10, 20, 50 y 100 tareas con 5 instancias cada tamaño, los cuales permiten inferir que el Algoritmo propuesto es mejor en tiempo computacional frente a las heurísticas FIFO, EDD y SPT.

En primer lugar, para el tamaño de 10 tareas el algoritmo presenta un tiempo de 0,213 mientras que para FIFO 0,288, EDD 0,278 y SPT 0,356 unidades de tiempo.

En segundo lugar, para el tamaño de 20 tareas el algoritmo presenta un tiempo de 0,270 mientras que para FIFO 0,569, EDD 0,500, y 0,825 unidades de tiempo.

En tercer lugar, para el tamaño de 50 tareas el algoritmo presenta un tiempo de 0,763 mientras que para FIFO 0,828, EDD 0,741 y SPT 0,597 unidades de tiempo. Aunque las heurísticas EDD y SPT muestran mejores resultados, no hay diferencia significativa debido a que el algoritmo presenta mejores resultados en la función de costo.

Por último, para el tamaño de 100 tareas el algoritmo presenta un tiempo de 1,483 mientras que para FIFO 1,081, EDD 0,800 y SPT 0,725 unidades de tiempo. En este caso, aunque las 3 heurísticas muestran mejores resultados, no hay diferencia significativa debido a que el algoritmo presenta mejores resultados en la función de costo. (ver Tabla 15) Apéndice L

Tabla 15.

Tiempo computacional del Algoritmo Recocido Simulado y las Heurísticas FIFO, EDD y SPT.

TAMAÑO DE TAREAS	**ALGORITMO S. A**	**FIFO**	**EDD**	**SPT**
10 TAREAS	0,213	0,288	0,278	0,356
20 TAREAS	0,270	0,569	0,500	0,825
50 TAREAS	0,763	0,828	0,741	0,597
100 TAREAS	1,483	1,081	0,800	0,725

11. Conclusiones

Según la revisión de literatura pocos autores han abordado el Problema de Secuencia de Servicio de Grúa (CSSP) con el fin de minimizar los costos de operación y a su vez, ningún autor ha abordado este problema teniendo en cuenta los parámetros estudiados en este trabajo, lo cual permite que este proyecto sea un referente para futuras investigaciones en esta temática.

El algoritmo propuesto presenta un mejor desempeño que el MILP para instancias pequeñas, dado que existe una diferencia en el tiempo de ejecución bastante significativa, donde el modelo MILP tarda 16 minutos en resolver las instancias para 10 tareas mientras el algoritmo lo hace en 0,213 unidades de tiempo.

Como resultado del análisis de medias presentado en los Apéndices E, F y G se observó que al aplicar el ANOVA de un solo factor para variar el valor de α, los cambios de temperatura y el tamaño de vecindario, se determina una muestra de 30 iteraciones que mejora el desempeño del algoritmo de Recocido Simulado en los tamaños de 10, 20, 50 y 100 tareas, siendo una buena alternativa para la solución de problemas complejos.

Validando el algoritmo propuesto de Recocido Simulado (S.A.) con las heurísticas FIFO, EDD, SPT se puede observar que el algoritmo obtiene mejores resultados en todas las instancias superando estás tres heurísticas, de las cuales, con la que tuvo mayor diferencia fue la heurística FIFO.

12. Recomendaciones

Para futuras investigaciones considerar restricciones adicionales tales como: la localización de la grúa, averías, costos de maquinaria ociosa, múltiples grúas, entre otras, de modo que se acerque al entorno real de un proyecto de construcción tales como, conjuntos residenciales, centros comerciales, etc.

Evaluar para el Algoritmo de Recocido Simulado otras estructuras del vecindario con el fin de determinar con cuales se tiene mejor rendimiento.

Resolver el problema planteado con otras metaheurísticas que han sido utilizadas en la literatura para validar los resultados obtenidos con los del algoritmo propuesto.

Profundizar en el estudio de heurísticas y metaheurísticas en la asignatura de Investigación de Operaciones I para incentivar a los estudiantes a explorar el área de investigación con el fin de adquirir habilidades en los diferentes lenguajes de programación que se abordan en este tipo de proyectos.

Referencias Bibliográficas

Abdelmegid, M. A., Shawki, K. M., & Abdel-Khalek, H. (2015). GA optimization model for solving tower crane location problem in construction site s. *Alexandria Engineering Journal*, *54*(3), 519–526. https://doi.org/10.1016/j.aej.2015.05.011

Al Hattab, M., Zankoul, E., Barakat, M., & Hamzeh, F. (2018). Crane overlap and operational flexibility: balancing utilization, duration, and safety. *Construction Innovation*, *18*(1), 43–63. https://doi.org/10.1108/CI-11-2016-0062

Al Hattab, M., Zankoul, E., & Hamzeh, F. R. (2017). Near-Real-Time Optimization of Overlapping Tower Crane Operations: A Model and Case Study. *Journal of Computing in Civil Engineering*, *31*(4), 05017001. https://doi.org/10.1061/(ASCE)CP.1943-5487.0000666

Antero, J., & Marín, A. (2016). *Mejora de tiempos de entrega en un flow shop híbrido flexible usando técnicas inteligentes. Aplicación en la industria de tejidos técnicos*. Retrieved from http://bdigital.unal.edu.co/52637/7/9111502.2016.pdf

Bard, & Feo. (1989). Operations sequencing in discrete parts manufacturing. *Journal of Management Science*, *35*, 249.

Bermúdez Colina, Y. (2007). *Aplicaciones de programación lineal, entera y mixta Applications of linear, mixed and integer programming*. Retrieved from http://www.redalyc.org/pdf/2150/215024822007.pdf

BLUM, Christian, y ROLI, A. (2008). Metaheuristics in combinatorial optimization: Overviewand conceptual comparison. *ACM Computing Surveys*, *35*, 268.

Bula, G. (2004). Programación de operaciones en el taller de trabajo utilizando la Meta-Heurística de Recocido Simulado. *Revista de La Faculta de Fisico - Mecanicas*, *3(1)*, 21–28. Retrieved from http://revistas.uis.edu.co/index.php/revistauisingenierias/article/view/2269

Castillo, & Fandiño. (2005). *Diseño de un modelo general para la planeación operativa y la secuenciación de actividades en pequeñas empresas con procesos de manufactura*. Universidad Industrial de Santander, Bucaramanga.

CHASE, R., JACOBS, R., & AQUILANO, N. (2005). *Administración de Operaciones*. (ed. Mc Graw Hill, Ed.) (décima). México D.F. Retrieved from https://es.scribd.com/doc/255783326/Administracion-de-Operaciones-Conceptos-y-Casos-Contemporaneos

Conway, M., & Miller. (1967). *Theory of scheduling* (Dover Publications). New York. Retrieved from https://books.google.com.co/books?hl=es&lr=&id=Yr5_kQDa_ssC&oi=fnd&pg=PA1&dq=Theory+of+scheduling.+Dover+Publications&ots=uc6opHB8W-&sig=HZ0o99xF_Usn2M5hpDQNR6pj7-Q#v=onepage&q=Theory of scheduling. Dover Publications&f=false

Dowsland, K. A., & Díaz, B. A. (2003). Heuristic design and fundamentals of the Simulated Annealing. *Revista Iberoamericana de Inteligencia Artificial Revista Iberoamericana de Inteligencia Artificial. No*, *7*(019), 1137–3601. https://doi.org/1137-3601

Dréo, J. (Johann). (2006). *Metaheuristics for hard optimization : methods and case studies*. Springer.

Duarte Muñoz, A., Pantrigo Fernández, J. J., & Gallego Carrillo, M. (2007). *Metaheurísticas*. Dykinson.

Forero Ramírez, S. (2018). Camacol presentó sus propuestas sectoriales para el próximo cuatrienio. Retrieved May 1, 2018, from https://camacol.co/prensa/noticias/camacol-presentó-sus-propuestas-sectoriales-para-el-próximo-cuatrienio

Forero, S. (2018). Constructores esperan crecer en un 4,6% para el 2018 pese a grandes retos. Retrieved April 25, 2018, from https://www.dinero.com/economia/articulo/constructores-esperan-crecer-en-2018-pese-a-grandes-retos/253252

Fuentes Penna, A. (2014). Problema del agente viajero. *Universidad Autonoma Del Estado de Hidalgo*, 1–10. Retrieved from https://www.uaeh.edu.mx/scige/boletin/tlahuelilpan/n3/e5.html

Gallego Rendón, R. A., Escobar Zuluaga, A. H., & Romero Lázaro, R. A. (2006). *Técnicas de Optimización Combinatoria*. (Universidad Tecnológica de Pereira, Ed.) (1st ed.). Pereira.

HILLIER, Frederick s; LIEBERMAN, G. J. (2010). *Introducción a la Investigación de Operaciones* (9th ed.). Mc Graw Hill.

Huang, C., Wong, C. K., & Tam, C. M. (2011). Optimization of tower crane and material supply locations in a high-rise building site by mixed-integer linear programming. *Automation in Construction*, *20*(5), 571–580. https://doi.org/10.1016/j.autcon.2010.11.023

Lien, L. C., & Cheng, M. Y. (2014). Particle bee algorithm for tower crane layout with material

quantity supply and demaLien, L. C., & Cheng, M. Y. (2014). Particle bee algorithm for tower crane layout with material quantity supply and demand optimization. Automation in Construction, 45, 25. *Automation in Construction*, *45*, 25–32. https://doi.org/10.1016/j.autcon.2014.05.002

Liu, & Maccarthy. (2015). General Heuristic Procedures and Solutions Strategies for FMS Scheduling. *International Journal of Production Research*, *37*, 3305.

Lundy, M., & Mees, A. (1986). Convergence of an annealing algorithm. *Mathematical Programming*, *34*(1), 111–124. https://doi.org/10.1007/BF01582166

María, A. :, & Hinojosa, A. (2003). *Diagrama de Gantt*.

Martí, R. (2001). Procedimientos Metaheurísticos en Optimización Combinatoria. *Departament d'Estadística i Investigació Operativa*, 1–60. Retrieved from http://www.uv.es/rmarti/paper/docs/heur1.pdf

Martí, R. (2003). *Procedimientos Metaheurísticos en Optimización Combinatoria*. Retrieved from https://www.uv.es/rmarti/paper/docs/heur1.pdf

Marzouk, M., & Abubakr, A. (2016). Decision support for tower crane selection with building information models and genetic algorithms. *Automation in Construction*, *61*, 1–15. https://doi.org/10.1016/j.autcon.2015.09.008

Metropolis, N., Rosenbluth, A. W., Rosenbluth, M. N., Teller, A. H., & Teller, E. (1953). Equation of State Calculations by Fast Computing Machines. *The Journal of Chemical Physics*, *21*(6), 1087–1092. https://doi.org/10.1063/1.1699114

Moins, S. (2002). Implementation of a Simulated Annealing algorithm for Matlab. Retrieved

from http://www.diva-portal.org/smash/record.jsf?pid=diva2%3A18667&dswid=-9464

Monghasemi, S., Nikoo, M. R., & Adamowski, J. (2016a). Sequential ordering of crane service requests considering the pending times of the requests: An approach based on game theory and optimization techniques. *Automation in Construction*, *70*, 62–76. https://doi.org/10.1016/j.autcon.2016.06.006

Monghasemi, S., Nikoo, M. R., & Adamowski, J. (2016b). Sequential ordering of crane service requests considering the pending times of the requests: An approach based on game theory and optimization techniques. *Automation in Construction*, *70*, 62–76. https://doi.org/10.1016/J.AUTCON.2016.06.006

Nareyek, A., & Alexander. (2001). *Constraint-based agents : an architecture for constraint-based modeling and local-search-based reasoning for planning and scheduling in open and dynamic worlds*. Springer. Retrieved from https://dl.acm.org/citation.cfm?id=1773873

Pinedo, M. (2005). *Planning and scheduling in manufacturing and services*. Springer. Retrieved from https://books.google.com.co/books/about/Planning_and_Scheduling_in_Manufacturing.html?id=6G2VB-GJ0fMC&redir_esc=y

Reeves, C. R. (1996). *Modern Heuristic Techniques*. (Modern Heuristic Search Methods, Ed.). Chichester, Inglaterra: Jon Wiley & Sons.

Rodríguez Ortíz, C. (2009). *Algoritmos heurísticos y metaheurísticos para el problema de localización de regeneradores.* Universidad Rey Juan Carlos. https://doi.org/49015032

Sadat, Z., Nadoushani, M., Hammad, A. W. A., & Akbarnezhad, A. (2017). Location

Optimization of Tower Crane and Allocation of Material Supply Points in a Construction Site Considering Operating and Rental Costs. *Journal of Construction Engineering and Management*, *147*(1), 04016089. https://doi.org/10.1061/(ASCE)CO.1943-7862.0001215

Shapira, A., Lucko, G., & Schexnayder, C. J. (2007). Cranes for Building Construction Projects. *Journal of Construction Engineering and Management*, *133*(9), 690–700. https://doi.org/10.1061/(ASCE)0733-9364(2007)133:9(690)

Springer,Verlag; Berlin, H. (2008). Introducción al Algoritmo Genético, (SN Sivanandam, SN Deepa).

Taha, H. A. (2012). *Investigación de operaciones*. Pearson Educación. Retrieved from https://drive.google.com/drive/folders/0B4A1MvEBEWXESy1hSzJrZGZSMlk

Taillar, E. (1990). some efficient heuristic methods for the flow- shop secuensing problem. *European Journal of Operational Research*, 65–74.

Takagi, K., & Nishimura, H. (2003). Control of a jib-type crane mounted on a flexible structure. *IEEE Transactions on Control Systems Technology*, *11*(1), 32–42. https://doi.org/10.1109/TCST.2002.806435

Tubaileh, A. (2016). Working time optimal planning of construction site served by a single tower crane. *Journal of Mechanical Science and Technology*, *30*(6), 2793–2804. https://doi.org/10.1007/s12206-016-0346-8

Tupia Anticona, M. F., & Tupia Anticona, M. F. (2005). Un Algoritmo GRASP para resolver el problema de la programación de tareas dependientes en máquinas diferentes (task scheduling). *Universidad Nacional Mayor de San Marcos*. Retrieved from

http://cybertesis.unmsm.edu.pe/handle/cybertesis/3241

Vargas Z, Juan Carlos A. (2014). ANÁLISIS SECTOR CONSTRUCCIÓN EN COLOMBIA. *Bogotá Colombia Chapter*, 1–7.

Vaughan, J., Smith, A., Kang, S. J., & Singhose, W. (2011). Predictive graphical user interface elements to improve crane operator performance. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, *41*(2), 323–330. https://doi.org/10.1109/TSMCA.2010.2064303

Velez, M., M. J. (2007). *Metaheuristicos: Una Alternativa Para La Solución De Problemas Combinatorios En Administración De Operaciones*. (EIA, Ed.), *Revista EIA*. Antioquia: Escuela de Ingeniería de Antioquia. Retrieved from www.scielo.org.co/scielo.php?script=sci_arttext&pid=S1794-12372007000200009

Wang, J., Zhang, X., Shou, W., Wang, X., Xu, B., Kim, M. J., & Wu, P. (2015). A BIM-based approach for automated tower crane layout planning. *Automation in Construction*, *59*, 168–178. https://doi.org/10.1016/j.autcon.2015.05.006

Zabala, P. (2014). Algoritmos y estructuras de datos III. Retrieved from http://www.dc.uba.ar/materias/aed3

Zavichi, A., Madani, K., Xanthopoulos, P., & Oloufa, A. A. (2014). Enhanced crane operations in construction using service request optimization. *Automation in Construction*, *47*, 69–77. https://doi.org/10.1016/j.autcon.2014.07.011