Solución al Problema de Programación de Proyectos con Restricción de Recursos (RCPSP) mediante un Algoritmo Híbrido Genético

Wilmer Buitrago Duarte

Enrique Romero Gualdrón

Trabajo de Grado para Optar al Título de Ingeniería Industrial

Director

Carlos Eduardo Díaz Bohórquez

Magister en Ingeniería Industrial

Codirector

Lina Mayerly Lozano Suarez

Ingeniera Industrial

Universidad Industrial de Santander

Facultad de Ingenierías Físico-Mecánicas

Escuela de Estudios Industriales y Empresariales

Bucaramanga

2019

Agradecimientos

A Dios, por ser el dador de todas y cada uno de mis logros y aprendizajes.

A mi madre Margarita Duarte quién ha sido mi apoyo incondicional y mi ejemplo de vida a seguir, a mis hermanos Gladys Buitrago y Edwin Buitrago quienes me han inspirado a ser el mejor en lo que me propongo.

A Henry Prieto quién incondicionalmente me ha acompañado durante todo el proceso de elaboración del presente trabajo y quien desde su comprensión ha sido fuente de apoyo y disciplina en todo momento y me ha enseñado a tener siempre la frente en alto, a Emilse Ascencio y Edwing Parra quienes son amigos muy valiosos por las lecciones que me han enseñado y el apoyo que siempre he obtenido por parte de ellos.

A nuestra codirectora Lina Lozano y a Fabian Torres quienes con su trayectoria recorrida nos guiaron de la mejor manera ayudándonos a dar lo mejor de nosotros en este trabajo.

Al profesor Carlos Díaz quien desde su amor por la academia nos orientó con mucho ahínco hacia la consecución de los objetivos propuestos.

A amigos y compañeros de Opalo y de la carrera por su apoyo a lo largo del desarrollo del proyecto y de la carrera.

Wilmer Buitrago Duarte.

A Dios, por darme la sabiduría y fortaleza de llegar a esta etapa de mi vida.

A mis padres Florinda y Pablo Antonio, a mis hermanos, mi cuñado Sergio y mi abuela Rosa Eva, por su apoyo incondicional.

Infinitas gracias a Lina M. Lozano y Fabian A. Torres. Por su apoyo incondicional durante todo el proceso de investigación

A los compañeros del grupo Opalo y al profesor Carlos Díaz por su orientación en el desarrollo del proyecto.

Enrique Romero Gualdrón

Contenido

Pág.

Lista de tablas

Pág.

Lista de Figuras

Pág.

Resumen

Título: Solución al Problema de Programación de Proyectos con Restricción de Recursos (RCPSP) mediante un Algoritmo Híbrido Genético[*].

Autores: Wilmer Buitrago Duarte
Enrique Romero Gualdrón[**]

Palabras clave: Problema de programación de proyectos, Algoritmo Híbrido Genético, recursos renovables, restricción de precedencia y de recursos.

Descripción:

En la presente investigación se presenta el Problema de Programación de Proyectos con Restricción de Recursos (Resource Constrained Project Scheduling Problem, RCPSP), el cual consiste en ordenar y secuenciar un conjunto de actividades que están sujetas a restricción de recursos y de precedencia entre cada una de estas, con el objetivo de minimizar el tiempo de ejecución del proyecto (Makespan). Este problema pertenece a los de tipo NP Hard, ya que al aumentar el número de actividades a procesar y estar limitado por la disponibilidad de recursos aumenta la complejidad computacional, por tal razón es necesario ser desarrollado mediante la implementación de métodos de aproximación (Metaheurísticas) que, si bien no dan la solución óptima, se puede llegar a una mejor solución en términos de tiempo computacional razonable.

Para dar solución a este tipo de problema se diseñó un Algoritmo Híbrido Genético, En el que el algoritmo principal es el Algoritmo Genético donde la población inicial es generada de forma aleatoria, el criterio de selección aplicado fue el de selección por torneo, posteriormente el operador de cruce fue por un punto, como operador de mutación se implementó el Algoritmo de Recocido Simulado, el cual modifica la secuenciación de las actividades respetando la restricción de precedencia y de recursos.

Se realizó un diseño experimental 2^3 con el objetivo de determinar como influyen los factores en la variable respuesta Makespan, finalmente se validó el algoritmo propuesto con 10 instancias de cada tamaño (J30, J60, J90 y J120) encontrados en la librería PSPLIB, obteniendo buenas soluciones para las instancias de menor tamaño, aunque para las soluciones de mayor tamaño también se llegó a la solución de la librería, aunque en menor proporción respecto a las instancias inferiores.

[*] Proyecto de grado

[**] Facultad de Ingenierías Físico Mecánicas. Escuela de Estudios Industriales y Empresariales. Programa de Ingeniería Industrial. Director M.Sc. Carlos Eduardo Díaz Bohórquez

Abstract

Title: Solution to the Resource Constrained Project Scheduling Problem (RCPSP) using a Genetic Hybrid Algorithm[*].

Authors: Wilmer Buitrago Duarte
Enrique Romero Gualdrón[**]

Keywords: Project programming problem, Genetic Hybrid Algorithm, renewable resources, restriction of precedence and resources.

Description:

This research presents the Problem of Programming with Resource Constrained Project (RCPSP), which consists in ordering and sequencing a set of activities that are subject to resource restrictions and precedence between each of these, with the objective of minimizing the project execution time (Makespan). This problem belongs to those of type NP Hard, since increasing the number of activities to be processed and being limited by the availability of resources increases computational complexity, for this reason it is necessary to be developed through the implementation of approximation methods (Metaheuristics) that, while not giving the optimal solution, a better solution can be reached in terms of reasonable computational time.

To solve this type of problem, a Genetic Hybrid Algorithm was designed, in which the main algorithm is the Genetic Algorithm where the initial population is generated randomly, the selection criteria applied were the selection by tournament, then the operator The crossover was for one point, as a mutation operator the Simulated Annealing Algorithm was implemented, which modifies the sequencing of activities respecting the restriction of precedence and resources.

An experimental 2^3design was carried out with the objective of determining how the factors influence the Makespan response variable, finally the proposed algorithm with 10 instances of each size (J30, J60, J90 and J120) found in the PSPLIB library was validated, obtaining good solutions for smaller instances, although for larger solutions the library solution was also reached, although to a lesser extent compared to lower instances.

[*] Proyecto de grado
[**] Facultad de Ingenierías Físico Mecánicas. Escuela de Estudios Industriales y Empresariales. Programa de Ingeniería Industrial. Director M.Sc. Carlos Eduardo Díaz Bohórquez.

Introducción

En el proceso de globalización, las empresas necesitan ser cada vez más competitivas y es necesario tomar decisiones de tipo táctica para dar tiempos de respuesta cada vez más cortos a los proyectos, esto se logra con una adecuada planificación de las actividades y a su vez mediante la asignación y uso efectivo de los recursos involucrados en ellas, este es uno de los retos a los que están expuestos los gerentes de proyectos en las organizaciones.

Los problemas de planeación de proyectos son muy comunes en cualquier industria u organización, ya que pueden ser aplicados en la programación industrial, proyectos de construcción, prestación de servicios, actividades cotidianas y rutinarias, entre otras (Rivera y Celín, 2010). Actualmente el problema de programación de proyectos con restricción de recursos (Resource Constrained Project Scheduling Problem, RCPSP) es uno de los problemas más importantes en el contexto de la programación de proyectos (Abbas, Shadrokh y Arkat, 2006), en el cual se considera como función objetivo minimizar la duración del proyecto (Makespan). De acuerdo con los recursos involucrados en cada una de las actividades programadas, están sujetas a restricciones de precedencia y de recursos, es por esto que en los últimos años los investigadores han centrado sus estudios en dar solución a esta problemática.

De acuerdo a lo mencionado anteriormente, el RCPSP representa una clase importante de problemas prácticos. A lo largo de los años, se han propuesto muchos algoritmos de optimización para dar solución y se han evaluado sus resultados utilizando instancias de prueba establecidas con los diferentes niveles de complejidad.

Durante el proceso de revisión de literatura, se evidenció los distintos métodos de aproximación empleados en solucionar el RCPSP, tales Metaheurísticas como: Algoritmo Genético (GA), Optimización de Colonia de Hormigas (ACO), Búsqueda Tabú (TS), Búsqueda de Vecindario Variable (VNS), Recocido Simulado (SA) y Colonia de Abejas Artificiales (ABC).

En particular, ciertos problemas de programación de proyectos para los cuales existen algoritmos de tiempo polinomial son fácilmente resueltos, sin embargo, se pueden transformar en NP-hard (Blazewicz, Lenstra y Rinnooy, 1983), al aumentar el número de actividades a programar, es por esta razón que se utilizan las Heurísticas, Metaheurísticas e Híbridos para dar solución.

En esta investigación, se plantea un Algoritmo Híbrido Genético (AHG), iniciando con el Algoritmo Genético, a través del cual se garantiza la búsqueda global (Valls, Ballestín y Quintanilla, 2008) estando éste conformado por los siguientes componentes: población, inicialización, evaluación, selección, cruce, mutación y reemplazo; posteriormente se complementa con la Metaheurística de Recocido Simulado en el proceso de mutación, la cual define un criterio de aceptación o rechazo de cada mutación del Algoritmo Genético, esto se obtiene al disminuir la temperatura gradualmente, en donde para una temperatura baja aumenta la probabilidad de encontrar una buena mutación permitiendo así enfocar gradualmente un área específica del espacio de búsqueda para lograr un ajuste fino.

Es de resaltar que con la hibridación a aplicar se garantiza abarcar un espacio de búsqueda amplio a través del Algoritmo Genético, y refinar la búsqueda local con el Algoritmo de Recocido Simulado permitiendo así hallar una buena solución.

La presente investigación está organizada por capítulos de la siguiente forma: En el capítulo 1 se presenta el planteamiento del problema RCPSP, en el capítulo 2 la justificación del proyecto, en el capítulo 3 los objetivos planteados, en el capítulo 4 se encuentra la revisión de literatura.

Luego, en el capítulo 5 está el marco teórico, en el capítulo 6 se realiza la descripción del Algoritmo de Recocido Simulado. En el siguiente, capítulo 7 se encuentra la descripción del Algoritmo Genético, posteriormente, en el capítulo 8 se define el problema a tratar en la presente investigación, siguiendo el orden, en el capítulo 9 se detalla el Algoritmo Híbrido propuesto para resolver el RCPSP, posteriormente, en el capítulo 10 se encuentra la validación del algoritmo propuesto. En el capítulo 11 se resumen los resultados computacionales, en el capítulo 12 se describen las conclusiones y finalmente en el capítulo 13 se plantean las recomendaciones para futuras investigaciones.

Tabla 1.

Cumplimiento de objetivos del proyecto

Objetivo	**Cumplimiento**
Realizar una revisión de literatura del problema de programación de proyectos con restricción de recursos (RCPSP) y los enfoques propuestos para dar solución.	Capítulo 4 y apéndice A
Definir un modelo matemático que represente al problema de programación de proyectos con restricción de recursos.	Capítulo 8
Diseñar un Algoritmo Híbrido Genético para dar solución al RCPSP y programarlo en MATLAB.	Capítulo 9 y apéndice B
Validar el desempeño del algoritmo propuesto y compararlo con instancias de la literatura.	Capítulo 11
Elaborar un artículo de carácter publicable con los resultados encontrados en este trabajo de investigación.	Apéndice E

1. Planteamiento del problema

Debido al proceso de globalización e interconexión, constantemente la exigencia en tiempo de respuesta es cada vez más corta y en cada empresa representa un reto cumplir a tiempo y con las exigencias requeridas por el cliente.

Por tal razón, la fabricación global se puede ver como un entorno orientado a proyectos, donde un cronograma de referencia efectivo del proyecto sirve de base para planificar actividades internas y externas como: la adquisición de recursos como materiales, mano de obra, maquinaria; el mantenimiento preventivo y el compromiso con las fechas de envío a los clientes. (Liu, Yung, Ip, 2007, p. 347), es por ello que surge una alternativa para representar el problema de programación de proyectos con recursos limitados (Resource Constrained Project Scheduling Problem, RCPSP).

El "(RCPSP) se discute con el objetivo de minimizar el tiempo de terminación de un proyecto. Debido a su universalidad, tiene una variedad de aplicaciones como fabricación, planificación de producción, gestión de proyectos y otros." (Agarwal, Tiwari, Mukherjee, 2007, p.584). En cada empresa es necesario parametrizar cada uno de sus proyectos teniendo en cuenta las variables que afectan, por ejemplo, el uso de recursos necesarios con sus respectivas precedencias y manejo óptimo de costos para poder modelar dichos proyectos con el fin de generar un algoritmo el cual genere un cronograma de referencia efectivo para la ejecución de actividades con la finalidad de reducir el tiempo de ejecución del proyecto (minimizar makespan), cumpliendo así con las exigencias requeridas en el mercado actual.

Para ejemplos de aplicaciones en la industria, es necesario mantener altos niveles de producción, la energía eléctrica y las plantas de fabricación se apagan para que las actividades de mantenimiento se puedan realizar en las máquinas / equipos. Estas paradas planificadas (cortes) generalmente ocurren al menos una vez al año y son más frecuentes en plantas con máquinas / equipos más antiguos. Debido al alto costo incurrido por la pérdida de producción, el objetivo es programar las actividades de mantenimiento de manera que se minimice la duración de la interrupción. Dado que los recursos necesarios para realizar las actividades de mantenimiento son muy limitados, el problema de programar estas actividades se define como un problema de programación de proyectos con recursos limitados (RCPSP). (Mckendall, Noble, Klein, 2008 p.53). En este ejemplo bajo la solución al modelo RCPSP se consigue una buena organización de la planeación para no afectar en pérdidas a la empresa.

El objetivo del RCPSP es determinar la programación óptima, dando como resultado la disminución del makespan del proyecto, cumpliendo con las restricciones de precedencia y de recursos, debido a la alta complejidad del problema pertenece a NP-hard (Blazewicz, Lenstra y Rinnooy, 1983), es por esta razón que se utilizan métodos aproximados para encontrar una buena solución. En el presente proyecto se pretende estudiar el RCPSP mediante un Algoritmo Híbrido Genético, en el cual se usará el Recocido Simulado en el proceso de mutación.

2. Justificación del proyecto

Actualmente, en las organizaciones es necesario desarrollar los proyectos en un periodo de tiempo cada vez más corto, siendo esta una ventaja competitiva. Para lograrlo, es necesario planificar efectivamente las actividades haciendo uso eficiente de los recursos y dar cumplimiento a las exigencias del cliente, esto se logra mediante la elaboración de un programa factible que se ajuste a la disponibilidad de los recursos de la empresa.

Por tal razón surge la necesidad de investigar acerca del Problema de Programación de Proyectos con Restricción de Recursos (RCPSP), dado que este problema está presente en cualquier tipo de industria y es el reto al cual se enfrentan los gerentes de proyectos, independientemente del objetivo planteado por la organización.

Es así, que durante el proceso de revisión de literatura se evidenció grandes aportes y numerosos estudios realizados en esta línea de investigación para dar solución a este problema, esto se debe a su importancia en el ámbito práctico y teórico; inicialmente, se aplicaron métodos exactos, sin embargo al incrementar el número de actividades a procesar, dado que en el entorno real estas son numerosas, los autores hacen uso de métodos de aproximación como las distintas Heurísticas y Metaheurísticas con la finalidad de encontrar buenas soluciones a dicho problema.

Por tal motivo, es de gran interés el estudio de este problema para el grupo de investigación OPALO y así contribuir a los avances que puedan surgir.

Finalmente, la presente investigación tiene como objetivo diseñar un Algoritmo Híbrido Genético que permita dar solución al problema RCPSP a bajo costo y realizar benchmarking con las soluciones encontradas en la librería PSPLIB.

3. Objetivos

3.1 Objetivo general

Construir un algoritmo Híbrido Genético para el Problema de Programación de Proyectos con Restricción de Recursos (RCPSP).

3.2 Objetivos específicos

- Realizar una revisión de literatura del problema de programación de proyectos con restricción de recursos (RCPSP) y los enfoques propuestos para dar solución.
- Definir un modelo matemático que represente al problema de programación de proyectos con restricción de recursos.
- Diseñar un Algoritmo Híbrido Genético para dar solución al RCPSP y programarlo en MATLAB.
- Validar el desempeño del algoritmo propuesto y compararlo con instancias de la literatura.

- Elaborar un artículo de carácter publicable con los resultados encontrados en este trabajo de investigación.

4. Revisión de literatura

Según Herroelen, De Rick y Demeulemeester (1998) el problema de programación de proyectos con recursos limitados (Resource Constrained Project Scheduling Problem, RCPSP), se remonta a finales de los años cincuenta; debido a su importancia en la "planeación de proyectos en el área de la investigación operativa, la cual se centra en la administración de tiempos, recursos y actividades para conseguir un objetivo a mediano o corto plazo" (Morillo, Moreno y Díaz, 2014, p.6). Para dar solución desarrollaron la técnica de revisión y evaluación de programas (Program Evaluation and Review Technique, PERT) y el método la ruta crítica (Critical Path Method, CPM), a partir de estas han surgido investigaciones de secuenciación complejas, sin embargo, por su naturaleza combinatoria no es posible resolver en tiempo razonable, Blanzewicz, Lenstra y Rinnooy (1983) afirmaron que la adición de restricciones de recursos a un problema de programación puede afectar su complejidad computacional. En particular, ciertos problemas bien resueltos, para los cuales existen algoritmos de tiempo polinomial, pueden transformarse en NP-hard (p.12). Dando como resultado que en las últimas tres décadas tome mayor importancia para los investigadores, quienes han realizado numerosos estudios implementando diferentes Algoritmos Exactos, Heurísticas, Metaheurísticas y Algoritmos Híbridos.

Inicialmente, Salewski, Schirmer y Drexl (1997) presentaron un enfoque basado en reglas de prioridad estocásticas y dinámicas al problema, generando una solución aleatoria paralela personalizado llamado Método de Muestreo Aleatorio (Randomized Mode Selection and Scheduling, RAMSES); otros autores que se basaron en dichas reglas son Hartmann y Kolisch, (2000) quienes las usaron para incorporarlas en el método X- pass, el cual también se fundamentan en Algoritmos Metaheurísticos con el fin de construir uno o más programas. Luego, Herroelen, De Ryck y Demeulemeester (1998) realizaron estudios del RCPSP utilizando el Método de Ramificación y Acotamiento (Branch and Bound, B&B) haciendo énfasis en el esquema de ramificación y las reglas de prioridad, crearon una lista de actividades que se llevó a cabo durante la ejecución del proyecto aplicando modelos de programación lineal.

Años más tarde, (Zamani, 2004) utilizó la técnica de Recocido Simulado (Simulated Annealing, SA) para encontrar el problema base, posteriormente agrega una ventana de tiempo para mejorar el resultado disminuyendo el esfuerzo computacional, encontrando así la solución que reduce la duración del proyecto, también Tseng y Chen (2006) presentaron una Metaheurística Híbrida para el RCPSP llamado ANGEL (Hybrid Metaheuristic ANGEL), que combina el Algoritmo de Optimización de Colonia de Hormigas (Ant Colony Optimization, ACO), Algoritmo Genético (GA) y estrategia de búsqueda local, arrojando buenos resultados para J30 y J60 (siendo J el conjunto de actividades). Posteriormente, Kolisch y Hartmann (2006) encontraron en su investigación mejores resultados al reducir el tiempo de ejecución de proyecto (makespan) mediante la hibridación entre el Algoritmo Genético (GA) y el Recocido Simulado (SA) para dar solución al RCPSP.

En el mismo año, Zhang, Li y Tam (2006) formularon un método basado en el método de Optimización de Enjambre de Partículas (Particle Swarm Optimization, PSO), en el cual cada

partícula representa una actividad con prioridad dentro de la ejecución del proyecto, ésta se elaboró bajo el enfoque de programación de actividades paralelas para transformar dicha partícula en un programa factible cumpliendo las restricciones de precedencia y de recursos; se comprobó que el PSO es más eficiente que el Algoritmo Genético (GA), esto se debe a sus características, pues utiliza experiencias locales y globales durante el proceso de búsqueda y se refleja con el menor número de iteraciones para encontrar la solución más cercana al óptimo. Siguiendo con este algoritmo, Fahmy, Hassan y Bassioni (2014) abordaron un nuevo enfoque del PSO, utilizando la técnica de justificación para mejorar la calidad del programa obtenido mediante este algoritmo, el cual se basa en combinar las soluciones en sí mismas, es decir, las actividades a una solución mejor en el vecindario si existe para encontrar una mejor. Bettemir y Sonmez (2014) desarrollaron un algoritmo híbrido conformado por el Algoritmo Genético (GA) y el Recocido Simulado (SA), el cual consiste en integrar la capacidad de búsqueda paralela de GA y de ajuste fino de SA, logrando así un algoritmo híbrido eficiente al ser comparado con los problemas de prueba los cuales se encuentran en librería PSPLIB.

Posteriormente, Yamashita y Morabito (2007) colocaron a prueba un Algoritmo de ramificación y Acotamiento (Branch and Bound, B&B) para tratar el RCPSP, en el mismo año, Debels y Vanhoucke (2007) utilizaron el Algoritmo Genético Basado en Descomposición (Decomposition Based Genetic Algorithm, DBGA), el cual resuelve de forma iterativa subpartes del proyecto al dividirlo.

Un año más tarde, Chtourou y Haouari (2008) desarrollaron un método Heurístico basado en reglas de prioridad mediante un esquema mejorado de listas de actividades en serie generadas al azar y luego se comprobó la robustez del cronograma teniendo en cuenta el tiempo obtenido al implementarlas; Shukla, Son y Tiwari (2008) recurrieron a la Heurística de Recocido Simulado

Adaptado por Muestreo (Fuzzy Based Adaptive Sample Sort Simulated Annealing, FASSA), inicialmente se genera una lista de programación de las actividades en serie (Serial-Schedule Generation Scheme, SGS), luego se implementa el Recocido Simulado por Muestreo (Sample-sort Simulated Annealing, SSA) y finalmente un Controlador Lógico Difuso (Fuzzy Logic Controller, FLC) implementados para resolver de manera eficiente el RCPSP y Valls, Ballestín y Quintanilla, (2008) abordaron el mismo problema mediante un Algoritmo Híbrido Genético (HGA), al introducir operadores de cruce, uno de mejora local y una nueva forma de seleccionar la población inicial, la cual consiste en aplicar primero un operador de doble justificación a cada uno de los horarios (individuos) para luego obtener la correspondiente lista de actividades.

Siguiendo con el orden cronológico, Mendes, Goncalves y Resende (2009) en su investigación utilizaron un enfoque del Algoritmo Genético (GA) utilizando las reglas de prioridad para construir horarios activos parametrizados, creando un esquema de generación de planificadores que hace incrementos de tiempo; comprobaron los resultados con los 1560 casos de 3 clases de problemas de prueba: J30 (480 casos con 30 actividades), J60 (480 casos con 60 actividades) y J120 (600 casos con 120 actividades), todos necesitaron 4 tipos de recursos y se compararon con 14 enfoques, que se encuentran en la biblioteca virtual de problemas (Project Scheduling Problem Library, PSPLIB), dando como resultado la reducción del makespan por el algoritmo propuesto. Luego, Moumene y Ferland (2009) realizaron una generalización del RCPSP al trabajarlo como Multi-Modo con una gran variedad de restricciones generando nuevas listas de actividades; al mismo tiempo, Mahdi, Rabbani, Amalnik, Razmi y Rahimi (2009) diseñaron una búsqueda de dispersión mejorada, generando una nueva ruta al implementar operadores de mutación y cruce en el Algoritmo Genético (GA).

Por otra parte, Montoya, Gutiérrez y Pirachicán (2010) trabajaron con el GA utilizando un modelo de Programación Orientada a Objetos (Object Oriented Programming, OOP), en el cual los cromosomas estaban basados en las características de los objetos, de esta manera se define el comportamiento de la población, creando una ventaja al programador cuando se utiliza este enfoque en todo el algoritmo para dar solución al RCPSP; los resultados fueron comparados con las instancias que se encuentran en la biblioteca virtual de problemas (Project Scheduling Problem Library, PSPLIB), se obtuvo un menor tiempo de cálculo. También, Chen, Shi, Teng, Lan y Hu (2010) presentaron un Algoritmo Hibrido Eficiente llamado ACOSS aplicado a la solución del RCPSP, ACOSS es una combinación de una estrategia de búsqueda local: Optimización de Colonias de Hormigas (ACO) y una Búsqueda de Dispersión (Scatter Search, SS) en un proceso iterativo, el cual inicia con ACO, este genera una lista de actividades en el espacio de solución, posteriormente SS construye un conjunto de referencias mediante las feromonas de ACO, el Algoritmo ACOSS se compara con los algoritmos modernos utilizando un conjunto de problemas estándar disponibles en la literatura. Los resultados experimentales validan la eficiencia del algoritmo propuesto.

Baradaran, Fatemi, Mobini y Hashemin (2010) llevaron a cabo un Algoritmo de Búsqueda de Dispersión Híbrida (Hybrid Scatter Search, HSS), utilizando las Técnicas de Revisión y Evaluación de Programas (Program Evaluation and Review Techniques, PERT) en donde se elige la ruta y dos operadores de cruce aplicando la permutación, obteniendo resultados apropiados para redes pequeñas y problemas del mundo real; mientras que Atli (2011) usó las reglas heurísticas, combinación de aleatoriedad y esquemas de prioridad, posteriormente las incorporó en el Algoritmo de Búsqueda Tabú (Tabu Search Algorithm, TSA), para dar solución a los 110 proyectos de Patterson, los cuales representan una acumulación de todos los problemas de recursos

múltiples existentes en la literatura y están en PSPLIB, obteniendo resultados favorables y cercanos al óptimo.

En el mismo año, Suresh, Dutta y Jain (2011) implementó el GA mediante la mutación repetitiva, mejorando la población inicial, generando una solución eficiente en la adquisición de proyectos de ingeniería de construcción (Engineering Procurement Construction, EPC), además, Ziarati, Akbari y Zeighami (2011) desarrollaron tres algoritmos por aparte, el Algoritmo de Abejas (Bee Algorithm, BA), Colonia de Abeja Artificial (Artificial Bee Colony, ABC) y Optimización de Enjambre de Abejas (Bee Swarm Optimization, BSO) en el cual cada algoritmo utilizó diferentes tipos de abejas en el espacio de búsqueda con el fin de encontrar la solución más adecuada, con dichos algoritmos se produjeron resultados competitivos en comparación con otros algoritmos investigados en este trabajo. Proon y Jim (2011) incorporaron en el GA un enfoque de búsqueda en vecindarios, mediante los operadores de búsqueda lograron mejorar la eficiencia del algoritmo.

Después, Zeighami, Akbari y Ziarati (2013) presentaron un Algoritmo Híbrido entre la Optimización de Enjambre de Partículas (PSO) empleado para la búsqueda global y la Optimización de Colonia de Termitas (Optimization Termite Colony, TCO) el cual genera soluciones en vecindarios para el problema estándar con referencias J30, J60, J90 y J120 pertenecientes de PSPLIB.

Posteriormente, Kellenbrink y Helber (2015) abordaron el RCPSP mediante una estructura flexible, en el cual es necesario tener en cuenta actividades opcionales que surgen durante el desarrollo del proyecto y a su vez las restricciones de precedencia, para dar solución implementaron el Algoritmo Genético (GA). Luego, Joshi, Jain y Bilolikar (2016) presentaron una Metaheurística Híbrida entre Búsqueda de Vecindad Variable (Variable Neighbourhood Search,

VNS) y el Algoritmo Genético (GA). La solución inicial fue generada por el VNS y se basaron en 8 reglas de prioridad, la cuales fortalecieron la búsqueda global del GA; el hibrido fue probado con J30 y J60, con 960 instancias del problema, arrojando mejores resultados la hibridación en comparación individual de los algoritmos que lo conforman; igualmente, Giran, Temur y Bekdas (2017) trabajaron con un método basado en la Heurística de Búsqueda Armónica (Harmony Search, HS) para dar solución al RCPSP en los proyectos de construcción, utilizando el método de la ruta crítica (Critical Path Method, CPM) ya que muestra las relaciones de precedencia, las actividades críticas y no críticas a desarrollar; otro autor es Eshraghi (2016) quien utilizó el Algoritmo de Evolución Diferencial (Differential Evolution Algorithm, AED) para los problemas J30 y J60 en los cuales se realizaron búsquedas locales con el fin de obtener soluciones cercanas al óptimo y a gran escala, se demostró la eficacia y velocidad de este algoritmo.

Finalmente, Coelho y Vanhoucke (2018) abordaron el RCPSP utilizando el Algoritmo de Ramificación y Acotamiento (Branch and Bound, B&B), variando estrategias y búsqueda de ramificación para encontrar el límite inferior y encontrar así soluciones cercanas al óptimo en tiempo razonable y reducir el tiempo de ejecución del proyecto.

5. Marco teórico

5.1 Optimización combinatoria

La optimización combinatoria es una rama de las matemáticas aplicadas, la cual está directamente relacionada con la investigación de operaciones, inteligencia artificial, ingeniería del software y teoría algorítmica. Siguiendo esta idea, Martí (2003) el objetivo de un problema de optimización es "encontrar el valor de unas variables de decisión para los que una determinada función objetivo alcance su valor máximo o mínimo. El valor de las variables en ocasiones está sujeto a unas restricciones" (p.1). La ventaja de aplicar los algoritmos de optimización combinatoria es resolver instancias de problemas difíciles, examinando el espacio de soluciones que casi siempre es grande. Estos algoritmos logran reducir el tamaño efectivo del espacio y explorar búsquedas de forma eficiente.

Estos tipos de problemas se pueden evidenciar en la industria o en la ciencia, como se dijo anteriormente, como ejemplo claro están los problemas de diseño de redes de telecomunicación, construcción y organizaciones de la producción, entre otros, hasta los más novedosos y sofisticados de ingeniería y reingeniería de software. Algunos ejemplos de problemas de optimización son:

- Problema del agente viajero (TSP)
- Problemas de la mochila
- Programación entera
- El problema de las n-reinas

- Problema de asignación cuadrática (QAP).

5.1.1 Tipos de complejidad computacional. La teoría de la complejidad computacional comienza con uno de los tantos aportes realizados por la máquina de Turing hacia el año de 1936, la cual era en su momento una computadora flexible y robusta. La complejidad computacional estudia la cantidad de recursos computacionales necesarios para resolver el problema, los cuales son: el tiempo mediante una aproximación al número de pasos de ejecución que un algoritmo emplea y el espacio mediante una aproximación a la cantidad de memoria utilizada para dar solución al problema (Moreno, Huecas, Sánchez y Almuneda, 2007). Existen varios tipos de complejidad computacional y son los siguientes):

- Clase P: Los problemas de esta clase se resuelven en tiempo polinomial, al utilizar un algoritmo que es menor en cierto valor calculado mediante la variable de entrada (N).
- Clase NP (No determinista polinomial): Son problemas de decisión, los cuales se resuelven haciendo uso de una máquina no determinista en un tiempo polinómico. Por definición P es una subclase de NP ($P \subseteq NP$).
- Clase NP Completos: Son problemas que no se pueden encontrar en P y son problemas de mayor complejidad que pertenecen a NP. Concluyendo, si el problema es NP y otro problema NP es reducible a este, es decir basta resolver el problema en cuestión para resolver cualquier otro.
- Clase NP hard: Son problemas muy difíciles de resolver y en su gran mayoría se utilizan métodos aproximados mediante heurísticas que permiten aproximarse a una solución óptima.

Figura 1. Tipos de complejidad computacional. Adaptado de Pastor R. (1999). Metalgoritmo de Optimización combinatoria mediante la exploración de grafos. Universidad Politécnica de Cataluña. Barcelona- España. doi:9788469274071

La adición de restricción de recursos a un problema de programación afecta su complejidad computacional. En particular, ciertos problemas bien resueltos, para los cuales existen algoritmos de tiempo polinómico, pueden transformarse en NP- hard. Blazewicz, Lenstra y Rinnooy (1983).

5.2 Problemas de programación de proyectos (Project Scheduling Problems).

En la programación de proyectos, (Herroelen, Demeulemeester y De Reyck, 1999) las tareas se refieren a las actividades que pertenecen a uno o más proyectos. La ejecución de las actividades del proyecto requiere del uso de diferentes tipos de recursos (dinero, mano de obra, maquinaria, etc.), también existen distintos objetivos de programación como la minimización del Makespan, minimización de costos y optimización de fecha de vencimiento, entre otros.

5.2.1 Características del recurso (α). Las características de un problema de programación de recursos se especifican mediante un conjunto que contienen como máximo tres elementos $\boldsymbol{\alpha_1}$, $\boldsymbol{\alpha_2}$ y $\boldsymbol{\alpha_3}$ los cuales se describen a continuación:

α_1 Denota el número de recursos de tipo de recursos

α_1= 0, No se consideran tipos de recursos en el problema de programación.

α_1= 1, Un tipo de recurso es considerado.

α_1= m, m número de tipos de recursos.

α_2 Denota específicamente el tipo de recursos usado:

α_2= 0, Ausencia de cualquier tipo de recurso

α_2= 1, Recurso renovable, su disponibilidad se especifica por periodo (Hora, día, semana, mes)

α_2= T, Recurso no renovable, su disponibilidad se especifica para todo el horizonte del proyecto T

α_2= 1T, Recursos renovables y no renovables (incluye los doblemente restringidos, cuya disponibilidad se especifica tanto para un periodo como en el horizonte del proyecto)

α_2= v, Recursos parcialmente no renovables, cuya disponibilidad se renueva en ciertos periodos de tiempo.

α_3 Describe las características de disponibilidad del recurso en el problema de programación de proyectos.

α_3= 0, Los recursos renovables están disponibles en cantidades constantes

α_3= va, Los recursos renovables están disponibles en cantidades variables.

5.2.2 Características de las actividades (β). Especifica las características de la actividad del proyecto.

β_1 Indica la preferencia de la actividad

β_1= 0 No Se permite preferencia de la actividad

β_1= pmtn, Se permiten las preferencias del tipo de prioridad – reanudación

β_1= pmtn - rep, Se permiten las preferencias del tipo de repetición de preferencias

β_2 Refleja las restricciones de precedencia

β_2= 0, Restricción de no precedencia

β_2= Cpm, Las actividades están sujetas a restricciones estrictas de precedencia

β_2= Min, Las actividades están sujetas a restricción de diagramación de precedencia del tipo inicio- inicio, finalización – finalización, inicio – finalización con retrasos mínimos de tiempo.

5.2.3 Medidas de rendimiento (γ). Denota criterios de optimización (medidas de rendimiento).

γ= C_{max}, Minimización del Makespan del proyecto

γ= T_{max}, Minimizar el retardo del proyecto

γ= av, Minimizar las disponibilidades de recursos para cumplir con la fecha del proyecto.

5.2.4 Características y restricciones de procesamiento. Se especifican a continuación:

Restricción de No interrupción de una actividad: no se permite que una actividad sea interrumpida mientras se realiza hasta su tiempo de finalización.

Restricción de recursos limitados: existen k tipos de recursos limitados R_k con la propiedad de que cada actividad j requiere r_{jk} unidades de R_k en todo momento durante su ejecución. Vale aclarar que, en ningún momento, algún recurso exceda el 100% en uso de su capacidad.

Restricción de precedencia: dado el listado de las actividades a realizar, en este listado se evidencia que hay ciertas actividades con restricción de precedencia, es decir, esas actividades no pueden comenzar hasta que todas sus actividades predecesoras se hayan concluido.

5.2.5 Función objetivo. Para la función objetivo se abordó dos enfoques, para ambos enfoques se tuvo en cuenta un conjunto de criterios que comprende los siguientes elementos:

Tiempo de Finalización S_j o Makespan: el cual representa el tiempo total que tardó el proyecto en ser terminado, se definió como $S_j=\text{Máx}\{S_1,\ldots,S_j,\ldots,S_J\}$.

El Retraso L_j fue definido en función de la Fecha Límite de Terminación del Proyecto G_j y el Tiempo de Finalización S_j, $L_j=S_j-G_j$. Se observa que si $L_j<0$ es porque el proyecto se terminó antes de la Fecha Límite de Terminación del proyecto, en el caso $L_j > 0$ el proyecto excedió dicha fecha límite.

La Tardanza T_j indica cuánto tiempo de más se tomó para la finalización del proyecto en el caso en que no se cumpla con tal fin antes o en la Fecha Límite de Terminación del Proyecto G_j, se define como sigue $T_j=\text{Máx}\{0,S_j-G_j\}$. Si T_j aparece cuando el proyecto no es terminado a tiempo, es decir, en la fecha límite y además nunca será un valor negativo.

Penalización por Unidad U_j se define como un criterio que asume el valor de 0 si $S_j \leq G_j$, o asume el valor de 1 en caso contrario. Este criterio aparece cuando existe una Tardanza.

Una vez definidos los criterios anteriormente mencionados, se inicia con el primer enfoque $\gamma=f_{max}$ que ha sido el más común, el cual se centra en minimizar ya sea el tiempo de finalización S_j o del retraso L_j. $f_{max} \epsilon \{S_{max}, L_{max}\}$. donde $f_{max}=\text{Max}\{f_j(S_j)\}$ con $f_j(S_j)=S_j,L_j$.

El segundo enfoque $\gamma=\sum f_i$ se ha centrado en totalizar los diferentes criterios.

$\sum f_j \epsilon \{\sum S_j, \sum T_j, \sum U_j, \sum (w_j*S_j), \sum (w_j*T_j), \sum (w_j*U_j)\}$ Donde $\sum f_j = \sum_{j=1}^{j=J} f_j(S_j)$

Con $f_j(S_j)=S_j,T_j,U_j,w_jS_j,w_jT_j,w_jU_j$, respectivamente donde w_j indica una ponderación para la actividad j.

5.2.6 Clases de programas. Un programa (Schedule), según Schwindt y Zimmemann (2015) es la asignación de trabajos a un conjunto de máquinas y se clasifican en tres tipos: semi-active, active, and non-delay schedules.

Semi-active scheduling (programación semiactiva): Es un programa factible en el que ninguna de las actividades se puede cambiar localmente.

Active scheduling (programación activa): Es un programa factible donde ninguna de las actividades puede ser desplazada a la izquierda local o globalmente.

Non-delay scheduling (programación sin demora): Es un programa factible, en el cual se puede iniciar ciertas actividades antes (las cuales no tengan precedencia) y sin violar la capacidad de recursos.

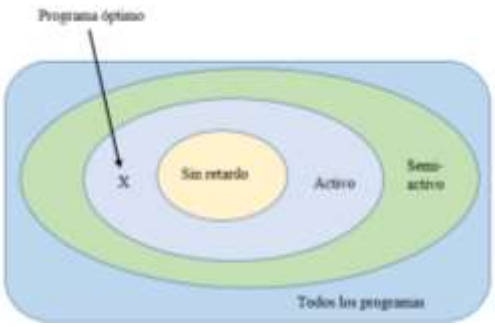

Figura 2. Diagrama de Ven de clase programas preemptivos para talleres de trabajo. Adaptado de: Pinedo M.L. (2012). Detreministics Models: Preliminares. Springer New York Dordrecht Heidelberg London (Eds.), Scheduling: Theory, algorithms y systems (pág. 25). doi:10.1007/978-1-4614-2361-4

5.3 Elementos de los problemas de secuenciación de proyectos

Este tipo de problema está formado por actividades, recursos, relaciones de precedencia y funciones de evaluación (Slowinki et al., 1994).

5.3.1 Actividades. Como se mencionó anteriormente, el proyecto está formado por un conjunto de actividades a procesar, cada una de ellas se debe realizar de un único modo, también tiene asociada una fecha de ejecución que indica el tiempo máximo en que se debe completar y una fecha de disponibilidad que marca el instante de tiempo a partir del cual se puede empezar a desarrollar.

5.3.2 Relación de precedencia. Según Ballestín (2002) una actividad no puede comenzar sin que la anterior haya finalizado, se visualiza está relación en un arco dirigido en el que la actividad se suele representar en el nodo o vértice y mediante una arista dirigida la relación de precedencia entre las dos actividades. Esta forma se conoce con el nombre de actividad en los nodos (Activity On Node, AON).

5.3.3 Recursos. Para llevar a cabo el desarrollo de las actividades es necesario realizar algún tipo de consumo, este consumo se modeliza mediante el uso de los recursos y se clasifican de la siguiente manera:

5.3.3.1 Recursos renovables. Este tipo de recursos están limitados en cada unidad de tiempo, es decir, hay cierta cantidad constante de recurso en cada periodo de tiempo y es renovable de un periodo a otro. Ejemplo: mano de obra, máquinas, herramientas, equipos, espacio.

5.3.3.2 Recursos no renovables. Están limitados por la duración total del proyecto y no hay restricciones en cada periodo de tiempo. Ejemplo: dinero, materias primas, energía.

5.3.3.3 Recursos doblemente restringidos. Se caracterizan porque están limitados sobre el horizonte de planificación y a su vez en cada periodo de tiempo. Ejemplo: flujo de caja por periodo.

5.3.3.4 Recursos parcialmente renovables. Estos se identifican porque están disponibles para subconjuntos de periodos; a esta clase pertenecen las clases anteriormente mencionadas.

5.3.4 Objetivos de la secuenciación de proyectos. La finalidad de los problemas de optimización es encontrar una solución factible al tema tratado, por esta razón es necesario cuantificar la calidad de la solución mediante una función de evaluación, distintos enfoques se han tratado como la minimización de la duración del proyecto (Makespan), maximización del valor presente neto (VPN), maximización de la calidad y minimización de los costos.

5.4 Resource Constrained Project Scheduling Problem (RCPSP)

También llamado problema de programación de proyectos de modo único (Single Mode Project Scheduling Problem, SMPSP). Según, Kwan, Mitsuo y Genji (2003), el clásico problema de RCPSP, es un proyecto en el cual las actividades se programan con el objetivo de disminuir el tiempo de ejecución del proyecto (Makespan). A continuación, se presentan las consideraciones que presenta el RCPSP:

- Es un solo proyecto que tiene una serie de actividades con duración conocida.
- La hora de inicio de cada actividad depende de la finalización de otra actividad (restricción de precedencia).

- Los recursos están disponibles en cantidades limitadas pero renovables de un periodo a otro.
- Las actividades no pueden ser interrumpidas y solo hay un modo de ejecución de cada una de ellas.
- El objetivo general es minimizar la duración del proyecto (Makespan).

Concluyendo, las actividades están interrelacionadas por dos tipos de restricciones: precedencia, una actividad no puede iniciarse antes que su predecesora haya terminada completamente y la otra es que cada una de ellas requiere para su ejecución cierta cantidad de recursos que están disponibles en el periodo.

5.5 Representación de la solución del problema

5.5.1 Diagrama de Gantt. Es una técnica desarrollada por Henry L. Gantt durante la primera guerra mundial. Este es un gráfico de barras que especifica la hora de inicio y finalización de cada actividad en el eje horizontal (abscisa), mientras que en el eje vertical (ordenada) representa el consumo de recursos para cada actividad.

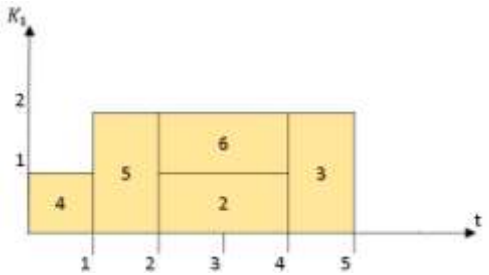

Figura 3. Diagrama de Gantt del ejemplo. Adaptado de Sprecher y Kolisch (1995). Semi-active, active, and non-delay schedules for the resource-constrained project scheduling problem. European Journal of Operational Research. (Pág. 99). doi: 10.1016/0377-2217(93) E0294-8.

5.5.2 Grafo disyuntivo. Según Viveros y Rivera (2017). Un grafo representa un proyecto, donde el nodo representa la actividad y las flechas la relación de precedencia entre estas, en cada nodo del grafo posee una duración y consumo de cada tipo de recurso (las cuales se encuentran en la parte superior e inferior respectivamente en cada nodo). La disponibilidad de cada tipo de recursos es de cuatro unidades y se denota por la letra b.

El objetivo es encontrar un programa S con tiempos de inicio S_i, para toda actividad i ∈ J de tal forma que se satisfagan las restricciones de precedencia y de recursos, y que el Makespan (duración) del proyecto (Z= Sn) sea mínimo.

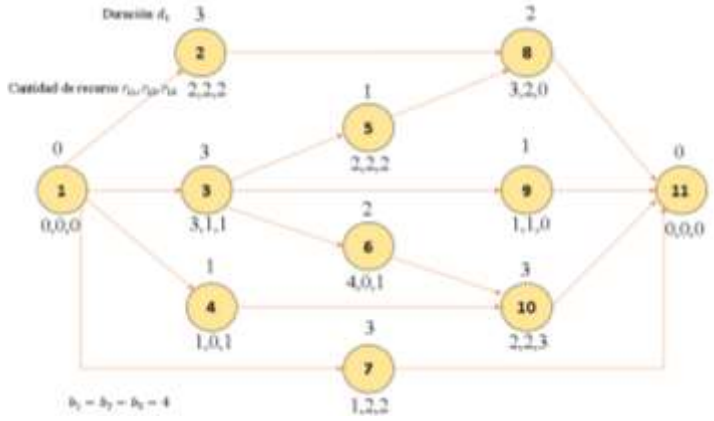

Figura 4. Representación del ejemplo del RCPSP. Adaptado de Morillo Torres, et al. (2014). Metodologías Analíticas y Heurísticas para la Solución del Problema de Programación de Tareas con Recursos Restringidos (RCPSP): una revisión. Parte 1. (pág. 250). Ingeniería y Ciencias. Universidad EAFIT. Medellín, Colombia. doi: 10.17230/ingciencia.10.20.

5.6 Métodos de solución

En Figura 7. Ejemplo de codificación y decodificación de cadenas cromosómicas.. se evidencia la clasificación de los métodos empleados para dar solución al RCPSP.

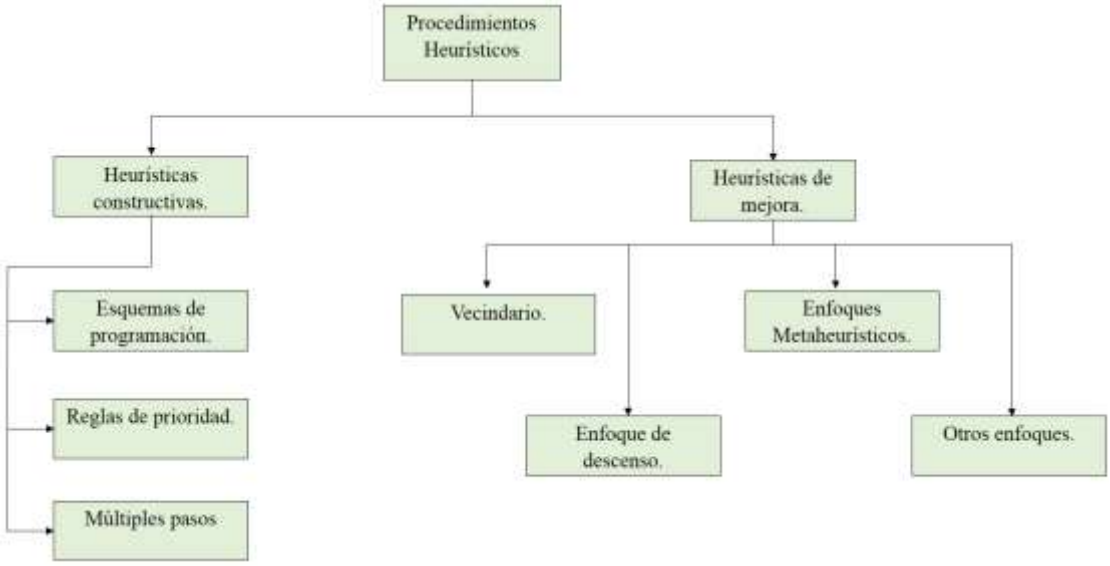

Figura 5. Clasificación de los métodos empleados para dar solución al RCPSP. Adaptado de Demeulemeester y Herroelen (2002). Project Scheduling: A Research Handbook. Capítulo 6. The Resource Constrained Project Scheduling Problem.

5.6.1 Métodos Exactos. Los métodos exactos se caracterizan por la obtención de la solución óptima del problema si esta existe. Este método se basa en técnicas de optimización como Programación Lineal Entera Mixta, Branch and Bound, Programación Dinámica, entre otras. (Demeulemeester y Herroelen, 2002, pág. 2)

Como se mencionó anteriormente son técnicas analíticas y matemáticas que generan una solución óptima si existe, este método se basa en supuestos y características específicas como la continuidad, espacio de búsqueda pequeño, entre otros. Sin embargo, solo se aplica a instancias de tamaño J30. Pero en la medida que crecen, aumentan la complejidad computacional y debido a sus desventajas en problemas aplicados, es por esto que mediante este enfoque no se puede dar solución al RCPSP, ya que posee complejidad computacional de naturaleza NP-Hard Blazewicz et al. (1983).

5.6.1.1 Ramificación y Acotamiento (Método de Branch and Bound): Según Morillo, Moreno y Díaz (2014) consiste en dividir el espacio factible y buscar en donde puede estar el óptimo, desechando los demás espacios de soluciones factibles que no mejoran la solución actual, la división se realiza truncando dicho espacio mientras se construye la solución. Otra característica de este método es la realización de la búsqueda de manera organizada y sistemática (pág. 260) y crean niveles al liberan recursos, lo que genera un conjunto de actividades elegibles para ser realizadas.

5.6.1.2 Programación Lineal Entera Mixta. Es una técnica que permite modelar y resolver problemas de optimización, el cual está conformado por variables enteras y continuas. Tao y Dong (2017) diseñaron una representación de red del proyecto llamada AND-OR para el RCPSP, luego desarrollaron un cronograma de programación lineal entera mixta y finalmente se introdujo al Algoritmo de Recocido Simulado, generando una nueva lista de selección de actividades; logrando la minimización del Makespan del proyecto.

5.6.1.3 Programación Dinámica. Morillo, Moreno y Díaz (2014) es una metodología empleada para resolver problemas en los que se deben tomar decisiones de manera secuencial y consiste en encontrar la solución global para un problema complejo, teniendo como base soluciones de etapas secuenciales previamente resueltas. Los autores Stanimirovic, Petkovic, Stanimirovic y Ciric (2009) utilizaron este método para resolver el RCPSP, en cada tiempo de programación t consideraron el conjunto elegible de actividades que pueden iniciar sin violar las restricciones de precedencia y de recursos.

5.6.2 Heurísticas constructivas. Se parte de un programa vacío y se agregan actividades hasta obtener un programa factible, para lograrlo se emplean las reglas de prioridad. A continuación, se presentan las heurísticas constructivas aplicadas en el concepto de programación de proyectos.

5.6.2.1 Esquemas de generación de programas (Schedule Generation Schemes, SGS). Se crean mediante la inclusión de algoritmos, los cuales generan un historial de recursos para el RCPSP. Los SGS tienen gran valor en la implementación de las heurísticas al dar solución al problema, esto se debe al límite de tiempo máximo y el mínimo requerido, buscando minimizar el tiempo de ejecución del proyecto (Makespan). A continuación, se presentan los tipos de esquemas de generación:

- Esquema de generación de horarios en serie (Serial Schedule Generation Scheme): Consiste en agregar una a una cada actividad al programa de forma secuencial hasta obtener un programa completo factible. En cada iteración, se elige la siguiente actividad en la Lista de Actividades y para esa actividad se asigna la primera hora de inicio posible, de tal manera que no se viole la restricción de precedencia y de disponibilidad de recursos.
- Esquema de generación de horarios paralelos (Parallel Schedule Generation Scheme): Consiste en agregar un conjunto de etapas al programa de forma secuencial hasta obtener un programa completo factible. En cada iteración se eligen las actividades a programar de la Lista de Actividades, asignando a cada una de estas la primera hora de inicio posible, de tal manera que no se viole la restricción de precedencia y de disponibilidad de recursos.
- Planificación hacia atrás (Backward planning): Según Demeulemeester y Herroelen, (2002) en este tipo de programa se inicia con la actividad final ficticia (dummy) y gradualmente se programan las demás actividades hasta llegar a la actividad de inicio

(ficticia) asignada su hora de inicio; esto se obtiene al invertir todas las relaciones de precedencia y al aplicar el esquema de programación en serie o en paralelo, sin embargo, se debe tener en cuenta la lista de prioridad.

- Planificación Bidireccional (Bidirectional planning): Demeulemeester y Herroelen, (2002) consiste en la programación hacia adelante y hacia atrás, para ello es necesario crear una lista de prioridad con sus respectivas reglas de prioridad.

5.6.2.2 Reglas de prioridad (Priority rules). Demeulemeester y Herroelen, (2002) en su libro definieron cinco categorías basadas en las reglas de prioridad, las cuales determinan la siguiente actividad a seleccionar durante el desarrollo del proyecto y estas dan como resultado una lista de prioridades; las reglas son las siguientes:

Tabla 2.

Reglas de prioridad

Regla de prioridad	Descripción	Clasificación
Basada en la actividad	Relaciona información con la actividad en sí y no con el resto del proyecto.	SPT: Shortest Processing Time (Tiempo de procesamiento más corto) LPT: Longest Processing Time (Tiempo de procesamiento más largo) RND: Random (Aleatoriamente)
Basada en la red	Relaciona información de precedencia de las diferentes actividades	MIS: Most Immediate Successors (Sucesores más inmediatos) MTS: Most Total Successors (La Mayoria de los Sucesores Totales) LNRJ: Least Non-related Jobs (Trabajos menos relacionados)

Regla de prioridad	Descripción	Clasificación
		GRPW: Greatest Rank Positional Weight (Rango de mayor peso posicional)
Basada en la ruta crítica	Se basa en los cálculos de la ruta crítica hacia adelante y hacia atrás	EST: Earliest Start Time (Tiempo de inicio más Temprano) EFT: Earliest Finish Time (Tiempo de incio más Tardío) LST: Latest Start Time (Última hora de inicio) LFT: Latest Finish Time (Tiempo de finalización más Tardío) MSLK: Minimum Slack (Holgura mínima) RSM: Resource Scheduling Method (Método de planificación de recursos) ESTD: Dynamic Earliest Start Time (Tiempo de inicio más temprano dinámico) EFTD: Dynamic Earliest Finish Time (Tiempo de finalización más temprano dinámico) MSLKD: Dynamic Minimum Slack (Holgura mínima dinámica)
Basada en el recurso	Se fundamenta en el uso del recurso de la actividad que se está realizando y también de las sucesoras	GRD: Greatest Resource Demand (Mayor demanda de recursos) GCUMRD: Greatest Cumulative Resource Demand (Mayor demanda acumulada de recursos) RED: Resource Equivalent Duration (Duración equivalente del recurso) CUMRED: Cumulative Resource Equivalent Duration (Duración equivalente del recurso acumulado)

Regla de prioridad	Descripción	Clasificación
Compuestas	Consiste en la combinación de las reglas anteriores.	WRUP: Weighted Resource Utilisation and Precedence (Utilización de recursos ponderados y precedencia) IRSM: Improved Resource Scheduling Method (Método mejorado de planificación de recursos) WCS: Worst Case Slack (Holgura en el peor de los casos)

5.6.2.3 Método de Múltiples pasos. Consiste en la combinación de los esquemas de programación con una o varias reglas de prioridad para obtener variedad de soluciones heurísticas, entre las cuales se elige la mejor opción. Este método se clasifica en:

- Método de esquemas multiprogramación: Conformado por dos esquemas de programación: serie y paralelo y tres direcciones: hacia adelante, hacia atrás y bidireccional, usando una sola regla de prioridad.
- Método de reglas de prioridad múltiple: Usando un único esquema de programación y única dirección, se usan múltiples reglas de prioridad.
- Método de muestreo: se suele usar un único esquema de programación y el tipo de programación hacia adelante con única regla de prioridad.

5.6.3 Heurísticas de mejora. Se basa en un programa factible obtenido a partir de una heurística constructiva, las operaciones se realizan en un cronograma que transforma una solución en una mejorada, el proceso se repite hasta obtener una buena solución. Dentro de las heurísticas de mejora se encuentra los enfoques metaheurísticos como los algoritmos: Búsqueda Tabú, Recocido Simulado, Algoritmo Genético, entre otros.

5.6.4 Métodos Iterativos de Aproximación. En esta categoría están aquellos en los que se incorpora un proceso iterativo, en la que cada iteración mediante el razonamiento realiza una búsqueda que permite la construcción o mejoramiento de la solución actual, Morillo et al. (2014) afirma que "estos métodos heurísticos comparten la característica de que no pueden garantizar una solución óptima (aunque a veces llegan a obtenerla)" (p. 261). En ocasiones, el razonamiento es simple pero no genera resultados adecuados, por lo tanto, se implementan Heurísticas y Metaheurísticas para mejorar la solución.

5.6.5 Metaheurísticas. Son estrategias inteligentes para diseñar o mejorar procedimientos heurísticos muy generales con un alto rendimiento. Muchos de los enfoques metaheurísticos son el resultado de observación de los procesos en la naturaleza. Morillo, Moreno y Díaz (2014) afirmaron en su investigación que estos algoritmos dan "solución a múltiples problemas complejos, como los combinatorios, grupo al cual pertenece el RCPSP" (p.205) cada uno de estos algoritmos tiene el objetivo de converger la búsqueda a la mejor solución en un vecindario local (intensificación) o dirigirla hacia el mejor vecindario entre muchos existentes en el espacio de búsqueda global (diversificación).

A continuación, se presentan las metaheurísticas más empleadas para dar solución al RCPSP:

5.6.5.1 Búsqueda Tabú (TS). Utiliza adecuadamente la memoria, siendo ésta una lista de soluciones visitadas previamente. Se mantienen varios tipos de listas, cada una con un propósito diferente; las de corto plazo incluyen soluciones visitadas recientemente, si hay una solución en esta lista a corto plazo, entonces no se debe explorar más el vecindario actual y la búsqueda se dirige a otro vecindario utilizando un punto de partida diferente. También existen listas deficientes,

en la cual la búsqueda se dirige hacia vecindarios fuera del actual sin encontrar una solución adecuada y la última es la lista de soluciones de buena calidad para encontrar buenos vecinos, en la cual se intensifica la búsqueda al generar un esquema de generación de estos. (Schwindt y Zimmermann, 2015, p.113). En su investigación Atli (2010) propuso un procedimiento Heurístico de alto nivel "Tabu Search Algorithm (TSA)" para proporcionar buenas soluciones a los problemas de programación de proyectos con restricción de recursos y duración de actividad determinístico, el cual presentó resultados computacionales que establecen la superioridad de la Búsqueda Tabú sobre los Algoritmos Heurísticos existentes.

5.6.5.2 Optimización de Enjambre de Partículas (PSO). Es una técnica de optimización que imita el comportamiento de un enjambre en búsqueda de alimento. En está, todas las partículas buscan alimentos en el espacio de búsqueda multidimensional basado en dos características: posición (solución) y velocidad (camino de la partícula). Si alguna partícula encuentra un mejor camino hacia la ubicación de los alimentos, atrae a otras para seguir su camino, en el que todas se mueven lentamente hacia la solución obtenida ya que esta es la global y finalmente todas alcanzan la misma posición siguiendo el camino óptimo más probable. Neetesh y Deo Prakash (2015), en su investigación (Rajeev, Kurian y Brijesh, 2015) formuló un esquema de programación en serie modificado mediante la Optimización de Enjambre de Partículas para resolver el RCPSP, teniendo como objetivo programar las actividades de tal manera que minimice el costo total ponderado de penalización por tardanza del proyecto, teniendo restricciones de precedencia y de recursos.

5.6.5.3 Algoritmo Genético (GA). Fue propuesto por Goldberg en el año de 1989, El GA se basa en la evolución de las especies de la naturaleza; en el proceso, las generaciones sucesivas de

poblaciones de una especie intentan mejorar sobre las anteriores a través de ciertos procesos genéticos y de supervivencia en las mejores condiciones. De acuerdo a lo anterior, es la Metaheurística más empleada para resolver los problemas de optimización combinatoria (Rostami, Moradinezhad y Soufipour, 2014), generando soluciones óptimas o casi óptimas. El autor Kim (2009) propuso un método del Algoritmo Genético mejorado para garantizar que el proceso de selección individual se realice correctamente al desarrollo del RCPSP.

5.6.5.4 Búsqueda de Vecindarios Variables (VNS). Es un método de búsqueda local, utiliza más de una estructura de vecindario, permitiendo de esta manera aumentar el espacio de exploración. El VNS empieza con un solo punto en el espacio (programa o cromosoma) y busca mejores puntos utilizando técnica de búsquedas locales, cuando se logra un punto que reemplaza al anterior, el proceso se repite hasta que ya no exista mejoría. Una ventaja de este enfoque es que requiere muy pocos parámetros y por ende es muy fácil de implementar (Joshi, Jain y Bilolikar, 2016). En la investigación realizada por (Fleszar e Hindi, 2004) presentaron un esquema de solución basado en esta Metaheurística, en la solución se utiliza la secuencia de actividad que son válidas en términos de restricción y precedencia; la búsqueda del espacio de solución se llevó a cabo mediante la generación de secuencias válidas utilizando dos tipos de estrategias de movimiento.

5.6.5.5 Colonia de Abejas Artificiales (ABC). Es una herramienta de optimización que se basa en el forraje inteligente mediante el comportamiento de enjambre de abejas; posee varias características: escalabilidad, falla, tolerancia, adaptación, velocidad, modularidad, autonomía y paralelismo, lo cual la convierte en una herramienta potencial para resolver problemas de

optimización. (Chen, Liang y Padilla, 2014) utilizaron esta Metaheurística efectiva para determinar un programa con vida útil mínima (p.235), para dar solución al RCPSP. En la colonia se consideran tres tipos de abejas: forrajeras, empleadas y desempleadas; las encargadas de buscar el alimento son las forrajeras y las empleadas, mientras que las desempleadas se quedan cerca de la colmena. En el modelo se define dos tipos de comportamiento: reclutamiento de forrajeros a fuentes de alimentos, dando como resultado una retroalimentación positiva y la otra es el abandono de las fuentes que tienen poco alimento como consecuencia de una retroalimentación negativa (Chen, Lian y Padilla, 2014).

5.6.5.6 Optimización de Colonia de Hormigas (ACO). Es una metaheurística probabilística para resolver problemas combinatorios, en esta se imita el mecanismo utilizado por las hormigas para encontrar el alimento e informar a la colonia donde se encuentra siguiendo la ruta más corta. Este algoritmo se clasifica como un procedimiento de construcción pseudoaleatoria, ya que inicialmente las hormigas se desplazan de forma aleatoria dejando por cada camino recorrido cierta cantidad de feromona, luego, las siguientes hormigas se desplazan de forma aleatoria, pero con mayor probabilidad de ir por los caminos que tienen mayor concentración de esta sustancia, Peng y Wang (2009) utilizaron este algoritmo para dar solución al RCPSP.

5.6.5.7 Procedimiento de Búsqueda Adaptable Aleatorizado y Codicioso (GRASP). Es una metaheurística muy útil en problemas en los cuales se deben tomar decisiones en las diferentes etapas del proyecto, este método consta de dos etapas: la primera es la fase constructiva en la que se generan soluciones factibles de forma iterativa. En cada iteración se incorpora un elemento de solución de manera aleatoria y en la fase de mejora se aplica una búsqueda local para mejorar la

solución encontrada en la primera, debido a que el punto de partida es diferente en cada iteración, se encuentran soluciones muy buenas en la búsqueda local. Anagnostopoulus y Koulinas (2013) muestra las ventajas de utilizar el enfoque metaheurístico para tratar los problemas de optimización en el campo de la construcción.

5.6.5.8 Recocido Simulado (SA). Es una metaheurística empleada para resolver problemas de optimización combinatoria. Este algoritmo es una variante de búsqueda local en el cual se realizan movimientos ascendentes con el objetivo de no quedar atrapado en un óptimo local. Este surgió en los años 50 y simula el enfriamiento del material en un proceso denominado "recocido" (Dowsland y Adenso, 2003). En la solución del RCPSP (Shukla, Son & Tiwari, 2008) utilizaron el SA mediante la generación de horarios en serie cumpliendo las restricciones de precedencia como de recursos y desarrollando artificialmente operadores de temperatura logrando de esta manera reducir el Makespan del proyecto.

5.6.6 Algoritmos Híbridos. La finalidad de "los algoritmos híbridos es combinar o unir varias metaheurísticas que se complementan entre sí, es decir aprovechar las cualidades de un algoritmo y compensar sus deficiencias con otro que se desempeñe mejor en ese aspecto" (Morillo et al. 2014, p.213). Estos son implementados en la solución de los problemas que hacen parte de la clasificación NP-Hard.

Principios básicos de los Algoritmos Híbridos (Morillo et al. 2014):

- Reemplazar un procedimiento no deseable de una Metaheurística por el de otra particular de mejor desempeño para este procedimiento.
- Implementar dos o más Metaheurísticas en secuencia, uno después de la otra.

- Programar dos o más Metaheurísticas en paralelo.

Para dar solución al RCPSP, Jia y Guo (2016) implementaron un Algoritmo Híbrido mediante la integración de los Algoritmos de Colonia de Abejas Artificiales (ABC) y Optimización de Enjambre de Partículas (PSO), dando como resultado que ABC mejora la capacidad de aprendizaje en las soluciones locales y globales en términos de PSO, mediante operadores de cruce e inserción.

6. Descripción del Recocido Simulado.

Es una Metaheurística basada en búsqueda local, se fundamenta en el proceso físico de enfriamiento de fluidos, fue incorporado este concepto en el año de 1983 por Kirkpatrick en la optimización combinatoria (Aarts, Korst y Michiels, 2005). El uso fue inspirado por una analogía entre el proceso de recocido sólido y el problema de resolver grandes optimizaciones combinatorias. En la física de la materia, el recocido se conoce como proceso térmico, en el cual, para obtener estados de baja energía de un sólido en un baño de calor se deben llevar a cabo dos pasos: el primero es aumentar la temperatura del baño de calor a un valor máximo en el que el sólido se derrite y el segundo consiste en disminuir paulatinamente la temperatura hasta que las partículas se organicen en su estado sólido fundamental, Tao y Dong (2017) el SA se caracteriza por una rápida convergencia y fácil implementación, lo cual permite ser implementado en problemas de tipo NP – Hard, especialmente para los problemas de programación de proyectos.

Por su parte, Cuberos Gallardo (2015) determinó la analogía entre el proceso de calentamiento usado en termodinámica y la adaptación del Algoritmo de Recocido Simulado empleado en problemas de optimización (pág. 61).

Tabla 3.

Analogía entre el algoritmo original y su adaptación a problemas de optimización

Proceso de Recocido	**Adaptación a problemas de optimización**
Configuración	Solución factible
Configuración de mínima energía interna	Solución óptima
Energía interna de la configuración	Costos asociados a la solución
Temperatura	Parámetro de control

Nota: Tomado de "Algoritmo de Recocido Simulado para la mejora de la eficiencia de una terminal multimodal, pág. 61" Dep. Organización Industrial y Gestión de Empresas II. Escuela Técnica Superior de Ingeniería. Universidad de Sevilla. Sevilla, 2015.

El Algoritmo de Recocido Simulado inicia con una solución *s*, luego mediante una estructura de vecindario genera una solución vecina *s'* de la solución actual, si la evaluación de la solución vecina *s'* es menor que la solución actual se cambia; si ocurre lo contrario, la evaluación de *s'* es mayor que *s*, se empeora eligiendo *s'* en lugar de s con una probabilidad de aceptación que depende de las funciones $\Delta f = f(s) - f(s')$ y de la temperatura T actual del sistema. La probabilidad permite salir de óptimos locales para llegar a óptimos globales; la probabilidad de aceptación se expresa de la siguiente manera $P(\Delta f, T) = e^{\Delta f / T}$

6.1 Parámetros del algoritmo de Recocido Simulado

6.1.1 Temperatura. Es el parámetro principal y determina la variación probabilística de aceptar una solución peor con el objetivo de no caer en óptimos locales, la temperatura inicial debe ser alta, posteriormente, paulatinamente disminuye y a su vez la probabilidad de aceptar una solución no factible.

6.1.2 Condición de parada. La principal condición de parada es determinar una temperatura mínima o mediante la fijación de un número máximo de iteraciones que realiza el algoritmo.

6.1.3 Longitud de la cadena. Este parámetro marca el número de iteraciones en cada uno de los valores establecidos de la temperatura, obteniendo como resultado búsquedas más exhaustivas a temperaturas bajas y de este modo se reduce la posibilidad de aceptar soluciones de peor calidad.

6.1.4 Tamaño de la vecindad. Se evalúa la solución obtenida en cada iteración formando un vecindario, luego se elige aleatoriamente una la solución que pertenece al mismo y se evalúan las condiciones, si cumple se toma como nueva solución.

En la generación de un programa (cronograma), es necesario tener en cuenta el orden de las siguientes variables:

- Como elegir el valor inicial de la temperatura To.
- Determinar la longitud de la secuencia para un solo nivel de temperatura, es decir, la iteración del algoritmo.

- Como ajustar la temperatura después de cada ejecución del algoritmo, para actualizar el esquema de la misma.
- Cuando detener el proceso de búsqueda.

Por tal razón, la efectividad y eficiencia depende del proceso del Recocido Simulado, para lograrlo es necesario que sea lento con el fin de generar un estado final factible, sin embargo, este no debe tener demasiado tiempo para alcanzar soluciones óptimas o casi óptimas.

Para comparar los diferentes programas se realiza mediante la experimentación. De acuerdo Cuberos Gallardo (2015) con la disminución de la temperatura, existen dos tipos de programas: el adaptativo que consiste en que la tasa de cambio se determina antes de que comience la búsqueda, permaneciendo fija y el otro tipo es el programa adaptativo, en el cual se ajusta la temperatura dependiendo de la información durante el proceso de búsqueda

7. Descripción del Algoritmo Genético

Los Algoritmos Genéticos son métodos adaptativos que se usan para resolver problemas de búsqueda y optimización; trabajan con una población de individuos, cada uno de los cuales representa una solución factible a un problema dado. Estos se basan en los postulados de Darwin (1859) los cuales se fundamentan en el proceso genético de los organismos vivos y se evidencian a lo largo de las generaciones, las poblaciones evolucionan en la naturaleza mediante los principios de selección natural y supervivencia de los más fuertes. Las soluciones se evalúan en términos de

su estado físico, el cual corresponde en su gran mayoría a la función objetivo (Martins y Ribeiro, 2014).

Mediante la implementación del GA no se garantiza encontrar la solución óptima al problema, sin embargo, encuentra soluciones de nivel aceptable en tiempo razonable con los demás algoritmos de optimización combinatoria.

Algunos de los conceptos más importantes del algoritmo son: Los individuos son las posibles soluciones al problema, los cuales se representan como un conjunto de parámetros (genes) y estos a su vez agrupados forman el cromosoma; el fenotipo representa el conjunto de parámetros con los cuales está constituido el cromosoma y contiene la información del organismo (genotipo), la adaptación al problema de un individuo depende de la evaluación del genotipo. Los operadores que intervienen en el GA son: selección, cruce y mutación.

7.1 Representación de la solución

En la figura 6 se evidencia la clasificación de la representación de la solución del RCPSP propuesta por Ballestín (2002).

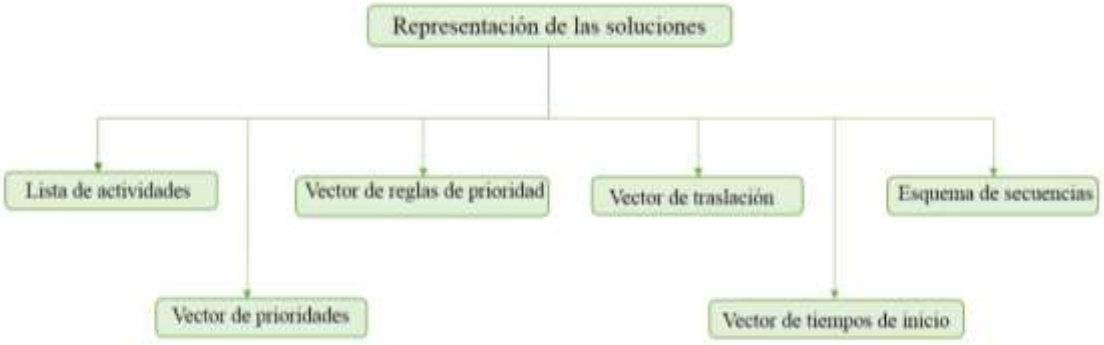

Figura 6. Representación de las soluciones. Adaptado de Ballestín (2002). Nuevos métodos de resolución del problema de secuenciación de proyectos con recursos limitados. Tesis doctoral. Universidad de Valencia. Facultad de ciencias matemáticas, Departamento de Estadística e Investigación Operativa. Valencia, España.

7.2 Codificación de soluciones para el RCPSP

Existen diferentes representaciones de las soluciones del RCPCP que han surgido a partir de técnicas metaheurísticas como alternativa de solución, a continuaciones describen algunas:

7.2.1 Lista de actividades. Es una permutación de las actividades $\lambda = (j_1,\ldots,j_i,\ldots,j_n)$ en la que se cumple las relaciones de precedencia, en el cual se define la posición y el orden creciente de las predecesoras, de esta forma se genera la lista de actividades. Los autores Bettemir y Sonmez (2014) utilizaron una representación de lista de prioridad para definir los cromosomas, en el que cada cromosoma en la población inicial representa una lista de prioridades factibles de prioridad seleccionada al azar.

7.2.2 Vector de prioridades (Random key). Es un vector en el cual se asigna un valor real a cada actividad y este depende de la prioridad (puede ser mayor o menor) asignada de acuerdo a

la regla empleada. Mendes, Goncalves y Resende (2009) desarrollaron la representación cromosómica del RCPSP al utilizar Random Key, el programa fue construido utilizando una regla de prioridad heurística generando programas activos parametrizados y en las que las prioridades de las actividades están definidas por el Algoritmo Genético.

7.2.3 Vector de reglas de prioridad. Es una lista de n reglas de prioridad, se usan como procedimientos de decodificación y se selecciona la actividad i-ésima a ser seleccionada de acuerdo la regla de prioridad empleada. Kolisch (1996) utilizó la heurística de planificación basada en reglas de prioridad, el cual estaba conformado por dos componentes: un esquema de planificación (serie o paralelo) y una regla de prioridad; el esquema formaba el conjunto de todas las actividades programables y posteriormente se empleaba una regla de prioridad específica para elegir una o más actividades que luego se programarían.

7.2.4 Vector de traslación (traslation vector). Consiste en una modificación hacia adelante, se utiliza para hallar el tiempo de inicio de la actividad j al calcular el máximo de tiempo de finalización de las predecesoras con su respectiva traslación. Hartmann (1998) implementó el Recocido Simulado (SA) basándose en el operador de traslación simple al seleccionar una actividad e insertarla inmediatamente después de otra(s) actividad(es), cumpliendo la restricción de precedencia.

7.2.5 Vector de tiempos de inicio. Se utiliza la codificación de enteros no negativos, en el cual representa el comienzo de cada actividad (se codifica la solución sobre la misma solución).

En otra codificación propuesta: este procedimiento está representada por cromosomas de extensión n genes el cual corresponde al número de actividades y cada gen tiene un valor binario ya sea uno o cero asignados de la siguiente manera:

$$gen_{(i,j,k,p,g)}=\begin{cases}1 \text{ si la actividad i se realiza en el periodo j}\\ 0 \text{ de lo contrario}\end{cases}$$

Donde p muestra el tamaño de la población y g presenta la generación, en la Figura 6.se muestra un ejemplo de codificación y decodificación de cadenas de cromosomas.

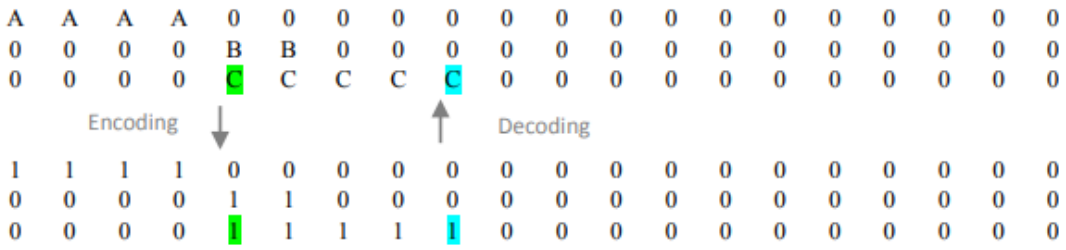

Figura 7. Ejemplo de codificación y decodificación de cadenas cromosómicas. Tomado de: Delgoshaei et al. (2015). Minimizing Makespan of a resource-constrained scheduling problem: A hybrid greedy and genetic algorithms. International Journal of Industrial Engineering Computations (p. 508). Serdang, Kuala Lumpur, Malaysia. doi: 10.5267/j.ijiec.2015.5.002

7.3 Criterio de selección

La selección se encarga de elegir los individuos que harán parte de la etapa de reproducción. En el proceso de revisión de literatura se evidenciaron distintos criterios de selección:

7.3.1 Selección por ruleta. Es un modelo de selección directa en donde los individuos con mejor valor de adaptabilidad se les asigna mayor proporción en la ruleta, obteniendo de esta

manera mayor probabilidad de ser seleccionados, este es un valor aleatorio comprendido de 0 a 1 y a su vez devuelve el individuo que corresponde a la posición.

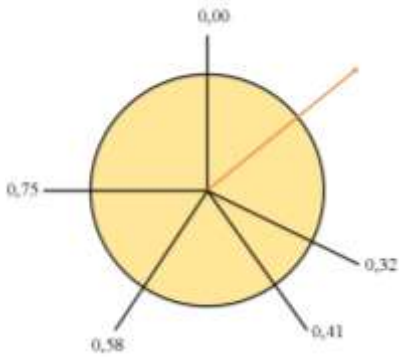

Figura 8. Ejemplo de selección por ruleta. Tomado de Gendreau y Potvin (2010). Handbook in Metaheuristics. International series in operations research and management science (pág. 121). Segunda Edición. doi: 10.1007/978-1-4419-1665-5

7.3.2 Selección por torneo de tamaño n. En este esquema de selección se elige aleatoriamente un subconjunto de soluciones de la población, para ello es necesario determinar el tamaño del torneo; luego se selecciona la mejor solución de este grupo para la fase de reproducción. La selección por torneo es muy utilizada en algoritmos evolutivos y funciona adecuadamente para una gran variedad de problemas.

7.3.3 Selección de rango lineal. En este esquema se utiliza el rango r, donde $r \in \{1, 2, \ldots, N\}$ asignado a cada individuo, donde N es el tamaño de la población de la solución y los individuos se clasifican respecto al valor de aptitud, el mejor de ellos obtiene el rango N y el peor el rango de 1.

$$p(i) = \frac{2ri}{N(N+1)}; \quad i = 1,2,\dots,N$$

$p(i)$ es la probabilidad de elegir el individuo i en la clasificación.

7.4 Operadores

7.4.1 Cruce. La operación de cruce selecciona los genes de los cromosomas progenitores y crea una nueva descendencia.

7.4.1.1 Operador de cruce de un punto. Es la técnica más sencilla de cruce, al seleccionar los dos padres se cortan sus dos cromosomas por un punto seleccionado aleatoriamente para generar dos segmentos diferenciados en cada uno de ellos, luego se intercambia cada cromosoma para generar los nuevos descendientes y de esta forma se obtiene información genética de cada padre.

En la figura 9 se puede ver el procedimiento, el primer hijo copia los genes del primer padre hasta el punto de cruce y los demás los adquiere del segundo padre; El segundo hijo toma los genes antes del punto de cruce del padre 2 y complementa con los del padre 1.

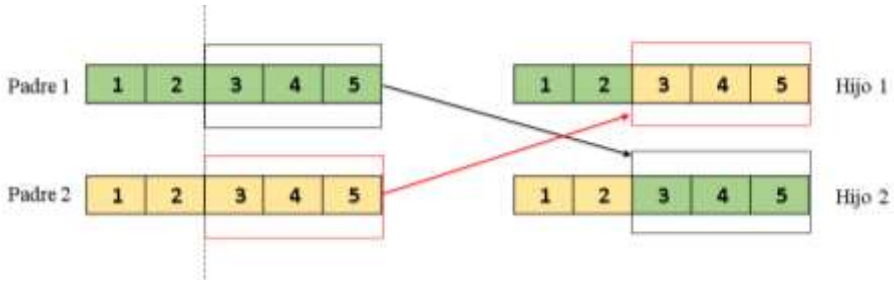

Figura 9. Cruce de un punto. Adaptado de: Gestal Pose. (2014). Introducción a los Algoritmos Genéticos, Depto. Tecnologías de la Informaci´on y las Comunicaciones. Universidad de Coruña, España. https://www.researchgate.net/publication/237812449

7.4.1.2 Operador de cruce de dos puntos. Es la generalización de cruce de un punto, consiste en cortar los cromosomas de los padres, se debe tener en cuenta que ninguno de los puntos de corte coincida con el extremo de los cromosomas para garantizar que se originen tres segmentos y finalmente se genera la descendencia eligiendo el segmento central de uno de los padres y los segmentos laterales del otro padre (ver figura 8.).

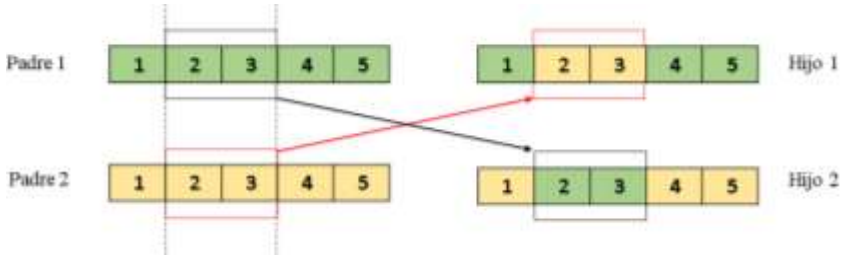

Figura 10. Cruce de dos puntos. Adaptado de: Gestal Pose. (2014). Introducción a los Algoritmos Genéticos, Depto. Tecnologías de la Informaci´on y las Comunicaciones. Universidad de Coruña, España. https://www.researchgate.net/publication/237812449

7.4.1.3 Operador de cruce uniforme. Cada gen de la descendencia tiene las mismas probabilidades de pertenecer a uno u otro padre, esta técnica implica la generación de una máscara de cruce (crossover mask) con valores binarios. Si en una de las posiciones de la máscara hay un 1, el gen situado en esa posición en uno de los descendientes se copie del primer padre. Si por el contrario hay 0 en el gen, para producir el segundo descendiente se intercambian los papeles de los padres o también se intercambia la interpretación de los 1 y los 0 en la máscara de cruce.

En la figura 11 se observa la mezcla de genes de cada uno de los padres, en este, el número de puntos de cruce es fijo pero esta determinado por tiempo medio L/2, siendo L la longitud del cromosoma.

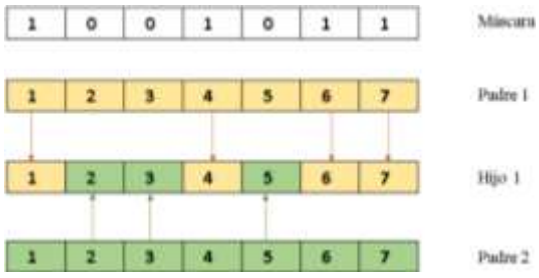

Figura 11. Cruce uniforme. Adaptado de: Gestal Pose. (2014). Introducción a los Algoritmos Genéticos, Depto. Tecnologías de la Informaci´on y las Comunicaciones. Universidad de Coruña, España. https://www.researchgate.net/publication/237812449

7.4.2 Mutación. La mutación genera diversidad en la población, dando como resultado variaciones aleatorias, Alcaraz y Maroto (2001) afirmaron que "la mutación se aplica a la población de descendientes, la cual consiste en alterar uno o más genes (posiciones) de un cromosoma (solución) seleccionado para introducir material genético perdido e introducir alguna variabilidad adicional en la población" (pág. 95) y en la investigación implementaron dos operadores de mutación, los cuales se mencionan a continuación:

- El primero fue una adaptación utilizado en el Algoritmo de Recocido Simulado para generar un vecino por Boctor en el año 2001; para determinar la secuencia de las actividades, elegían una posición al azar, sin embargo para generar las soluciones viables en cuanto a precedencia, esta posición debía ser más alta que cualquiera de sus predecesoras y a la vez más baja que cualquiera de sus sucesoras, posteriormente la actividad se inserta en esta posición (nueva) con una probabilidad de mutación (P_{mut}). Esta probabilidad es muy pequeña.
- Para el segundo operador tomaron como base el implementado en el Algoritmo Genético por Hartmann en el año de 1998, el cual parte de una solución, luego se intercambia cada

posición con la siguiente con una probabilidad de mutación (P_{mut}) teniendo en cuenta provenga de una solución factible.

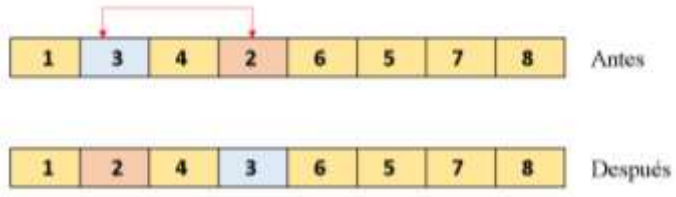

Figura 12. Operador de mutación. Adaptado de: Alcazar y Maroto (2001). A Robust Genetic Algorithm for Resource Allocation in Project Scheduling, Annals of Operations Research.

8. Problema de Programación de Proyectos con Restricción de Recursos

Se realizó una revisión de los modelos propuestos en la literatura para representar el problema y el que más se ajusta con el objetivo de minimizar el Makespan es el propuesto por Sprecher, Kolisch y Drexl (1995).

8.1 Descripción del problema

El Problema de Programación de Proyectos con Restricción de Recursos (RCPSP) es una extensión del Job Shop y se considera como el problema básico más importante dentro de la secuenciación con recursos limitados, el cual consiste en procesar un conjunto de actividades sujetas a restricción de precedencia y de recursos, siendo estos últimos compartidos por varias actividades, de esta

manera se busca la asignación adecuada de los mismos con el fin de reducir el tiempo de ejecución del proyecto (Makespan).

Las restricciones y consideraciones del problema son las siguientes:

- Un único proyecto
- Formado por un conjunto de actividades j = 1,2, 3, …, J
- Las actividades 1 y J son ficticias (no consumen recursos ni tiempo de procesamiento)
- Único modo de procesamiento de cada actividad.
- No se permite interrumpir el procesamiento de las actividades (los tiempos de alistamiento están incluidos en el tiempo de procesamiento).
- Los recursos son renovables de un periodo a otro (Una vez un recurso está ocupado, no se liberar hasta que finalice la actividad).
- Hay restricción de precedencia entre actividades (no inicia una actividad sin que haya finalizado su predecesora) y de recursos.
- Tipos de recursos r =1,2,3,…,R
- La función objetivo es minimizar el Makespan.
- Los parámetros son conocidos, determinísticos y enteros no negativos.

8.2 Formulación del problema

En la presente investigación del problema de programación de proyectos con restricción de recursos (RCPSP) se basa en el modelo propuesto por Sprecher, Kolisch y Drexl (1995). El RCPSP considerado está formado por un conjunto de j = 1,2,…, J actividades a procesar, no se acepta interrupción de las mismas, cada una tiene duración d_j, también entre ellas existe restricción de

precedencia (no se inicia una actividad sin haber finalizado la(s) predecesora(s)) y de recursos r = 1, 2, … R para la ejecución, cada uno de estos tiene una capacidad (disponibilidad) total de recursos K_r para cada periodo de programación *t* y a su vez se requiere k_{jr} unidades de recurso r para procesar la actividad j, siendo 1 y J actividades ficticias las cuales representan el inicio y finalización del proyecto, estas no consumen tiempo ni recursos; se asume que los tiempos de alistamiento están incluidos en el procesamiento, el objetivo es encontrar un programa que satisfaga las restricciones mencionadas anteriormente y que a su vez minimice la duración del proyecto (Makespan). Las cantidades d_j, k_{jr} y K_r son números enteros no negativos para ∀ j ∈ [1, J] y ∀ r ∈ [1, R].

Índices

j actividad $A= \{j \in Z^+ | 1 \leq j \leq J\}$

r Tipo de recurso $B= \{r \in Z^+ | 1 \leq r \leq R\}$

t periodo de tiempo $C= \{t \in Z | 1 \leq t \leq T\}$

Parámetros

J Número de actividades

R Tipos de recurso

T Fecha de vencimiento del proyecto

P_J Conjunto de actividades predecesoras i de la actividad j $i \in P_j$

d_j Tiempo de duración de la actividad j $D= \{(\forall_{j \in A})(\exists_{d_j \in Z}) | d_j \geq 0\}$

K_{rt} Capacidad total disponible del recurso r en el periodo t $E=\{(\forall_{r \in B})(\exists_{K_r \in Z}) | K_r \geq 0\}$

k_{jr} la actividad j consume cierta cantidad de recurso r. $F= \{(\forall_{r \in B})(\forall_{j \in A})(\exists_{k_{jr} \in Z}) | k_{jr} \geq 0\}$

8.2.1 Variables.

$$X_{jt} = \left\{ \begin{array}{c} \textbf{1 si la actividad j se termina en el periodo t} \\ \textbf{0 de lo contrario} \end{array} \right\}$$

t= EF_j,…, LF_j

EF_j Hora de finalización más Temprana de la actividad j

LF_j Hora de finalización más Tardía de la actividad j

8.2.2 Función de costo. La función de costo es el criterio empleado para medir la calidad de la solución obtenida. En el problema abordado se utiliza el Makespan (tiempo de finalización de la última actividad J).

Min C_{max}

$$minimizar \sum_{t=EF_J}^{LF_J} tx_{Jt}$$

8.2.3 Restricciones.

$$\sum_{t=EF_j}^{LF_j} x_{jt} = 1 \quad \forall\, j \in A \tag{1}$$

$$\sum_{t=EF_i}^{LF_i} tx_{it} \leq \sum_{t=EF_j}^{LF_j} (t - d_j) x_{jt} \qquad \forall\, j \in A,\ i \in P_j \tag{2}$$

$$\sum_{j=1}^{J} k_{jr} \sum_{\tau=t}^{t+d_j-1} X_{j\tau} \leq k_{rt} \qquad \forall\, r \in B;\ t=1,\ldots,T;\ \tau=t, t+1,\ldots,t+d_j-1 \tag{3}$$

$$X_{jt} \in \{0,1\} \quad j=1,\ldots,J \qquad t= EF_j, \ldots, LF_j \tag{4}$$

La restricción (1) representa la finalización de la actividad, la cual debe terminar exactamente en un periodo de tiempo dentro del intervalo (EF_j, LF_j), la ecuación (2) asegura la relación de precedencia entre las actividades (no se inicia una actividad sin haber finalizado sus predecesoras) y (3) asegura que todos los consumos de recurso k_{jr} para cada tipo de recurso r por parte de todas actividades j programadas en el periodo t no exceda el límite de la capacidad máxima de recurso disponible K_{rt} en el periodo t, τ es un intervalo de tiempo ($\tau \subseteq C$) de longitud d_j el cual está comprendido desde el periodo t en donde la actividad j comienza a ser ejecutada y termina en el periodo $t + d_j - 1$ en el cual finaliza la ejecución de dicha actividad.

9. Algoritmo Híbrido propuesto para el Problema de Programación de Proyectos con Restricción de Recursos (RCPSP)

En el Algoritmo Híbrido propuesto para dar solución al RCPSP se utiliza el Recocido Simulado como operador de mutación del Algoritmo Genético.

Para ilustrar el problema se utiliza el ejemplo descrito por Morillo, Moreno y Díaz (2014), el cual está conformado por 11 actividades incluidas las dummy, y tres tipos de recurso. Las diferentes soluciones para este proyecto y los programas relacionados para dichas soluciones se muestran en la figura 13.

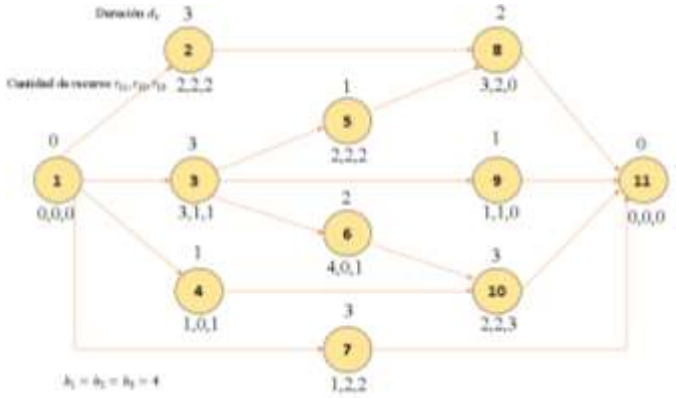

Figura 13. Ejemplo de un proyecto. Modificado de Morillo, Moreno y Díaz (2014). Metodologías Analíticas y Heurísticas para la Solución del Problema de Programación de Tareas con Recursos Restringidos (RCPSP): una revisión. Parte 1. Ingeniería y Ciencia. Pág. 250

9.1 Parámetros de entrada

- Número de actividades: Cantidad de actividades a realizar durante la ejecución del proyecto, incluyendo las dos actividades ficticias (inicial y final).
- Tipos de recursos: Recursos a utilizar en el desarrollo de la actividad.
- Capacidad del recurso: Cantidad de recurso máximo disponible para cada tipo de recurso.
- Duración de la actividad: Tiempo de ejecución de la actividad.
- Horizonte de planificación: Tiempo máximo de ejecución del proyecto.
- Predecesores de la actividad: Conjunto de actividades previamente finalizadas antes de comenzar a ejecutar una nueva actividad.

9.2 Representación de la solución

De acuerdo a Paraskevopoulos, Tarantilis e Ioannou (2012) en la literatura muchos de los investigadores han empleado la representación de los individuos mediante un vector de lista de actividades y el vector de Random Key, dado que Hartmann y Kolisch (2000) afirmaron que la Lista de Actividades es más eficiente de acuerdo a los resultados computacionales obtenidos en trabajos previamente realizados. En esta investigación se emplea la representación basado en Lista de Actividades la cual consiste en una secuencia aleatoria y factible de precedencia la cual consiste en una secuencia de las actividades a programar elegidas al azar pero se cumple la restricción de precedencia entre las actividades secuenciadas (ver *Figura 14*) en las cuales se presentan 3 Listas de Actividades (A,B,C.). Una vez obtenida la Lista de Actividades se procede a programarla usando un Esquema de Generación de Programación (SGS), para el caso de esta investigación se generó dicho esquema a través del Método en Serie el cual consiste en programar las actividades en el orden de la secuencia de la Lista de Actividades tan pronto como sea posible, este esquema construye un Cronograma Activo, un Cronograma Activo es un cronograma factible donde ninguna de las actividades j, 1 <j <J, puede ser desplazada a nivel local o global a la izquierda Sprecher, Kolish, Drexl (1995).

Figura 14. Lista de actividades para la ejecución del proyecto. Generado para el ejemplo de Morillo, Moreno y Díaz (2014). Metodologías Analíticas y Heurísticas para la Solución del Problema de Programación de Tareas con Recursos Restringidos (RCPSP): una revisión. Parte 1. Ingeniería y Ciencia. Pág. 250

En el ejemplo anterior para generar la Lista de Actividades A, se asigna en la primera posición la actividad número 1 por ser la única disponible ya que es la antecesora de todas, en la segunda posición de la Lista de Actividades se asigna una actividad sucesora de la actividad 1 las cuales son 2,3,4,7 ahora las actividades disponibles para asignar, como ya se dijo anteriormente se usó el Criterio Aleatorio para elegir una de las cuatro actividades posibles, se eligió así al azar la actividad 7 para asignar en la segunda posición, en la tercera posición se asigna una actividad disponible las cuales son las actividades sucesoras de las actividades asignadas (1,7) y que no han sido previamente asignadas, dichas actividades disponibles ahora son 2,3,4 asignando así al azar la actividad 3 para la posición tres, en la posición 4 se tiene como actividades disponibles 2,4,5,6,9 asignado así al azar la actividad 4 en la posición cuatro, se asignan así sucesivamente las siguientes actividades hasta culminar con la actividad 11, la cual es sucesora de todas las actividades del proyecto.

Para la generación de la Lista de Actividades B y C se sigue el mismo procedimiento el cual hace parte del Método de Generación de Esquema en Serie nombrado anteriormente.

9.3 Algoritmo Híbrido Genético

El Algoritmo Híbrido Genético está compuesto por un Algoritmo Genético con una modificación en el operador de mutación, esto se da con la probabilidad de mutación (si se da el caso de mutar) entonces se realiza mediante la implementación del Recocido Simulado.

9.3.1 Población inicial. La generación de la población inicial es vital para el Algoritmo Genético ya que influye directamente en la tasa de convergencia y en la calidad de las soluciones

óptimas. La población inicial representa un conjunto de Lista de Actividades generadas aleatoriamente, siendo cada una de estas factible de precedencia; Alcaraz y Maroto (2001) utilizaron el esquema de generación de programación en serie (Serial Schedule Generation Scheme, SGSS) para transformar dicha lista en una programación factible de recursos, procesando cada actividad una por una en el orden de la lista de actividades y programarlas en la precedencia más temprana y a su vez en la hora de inicio factible de recursos, se utiliza este tipo de esquema ya que Kolisch (1996) demostró que es adecuado para instancias que tienen recursos moderadamente limitados, mientras que el esquema de generación en paralelo busca en un espacio de soluciones más pequeño que el esquema en serie pero con un rendimiento de regular, además en el espacio de la solución puede no contener la solución factible y es aplicado en gran medida para problemas con recursos altamente restringidos.

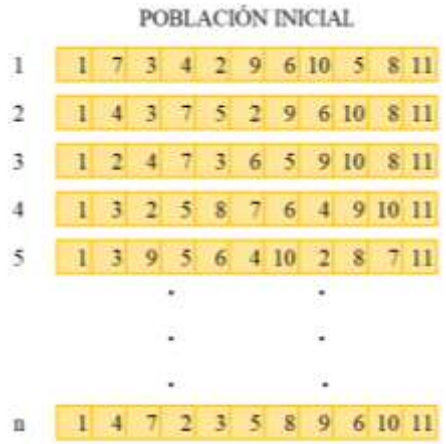

Figura 15. Población inicial para el ejemplo del proyecto. Generado para el ejemplo de Morillo, Moreno y Díaz (2014). Metodologías Analíticas y Heurísticas para la Solución del Problema de Programación de Tareas con Recursos Restringidos (RCPSP): una revisión. Parte 1. Ingeniería y Ciencia. Pág. 250

Cada individuo es una Lista de Actividades, la población inicial consiste en generar n Listas de Actividades y a cada una se le calcula su respectiva función de aptitud la cual es el Makespan correspondiente a cada Lista de actividades.

9.3.2 Criterio de selección. Se utilizó como criterio de selección el torneo determinístico de tamaño n para seleccionar los individuos mejores adaptados de la población inicial.

De la población inicial se eligen 4 individuos al azar que compiten entre sí bajo el criterio de Makespan como función fitness en el que se elige ganador del torneo al individuo con menor Makespan, en caso de empate se elige el primer individuo, se repite nuevamente el torneo para elegir al siguiente padre, se lleva a cabo tantos torneos como número de individuos hay en la población inicial (n), igualando así el número de padres al número de individuos de la población inicial.

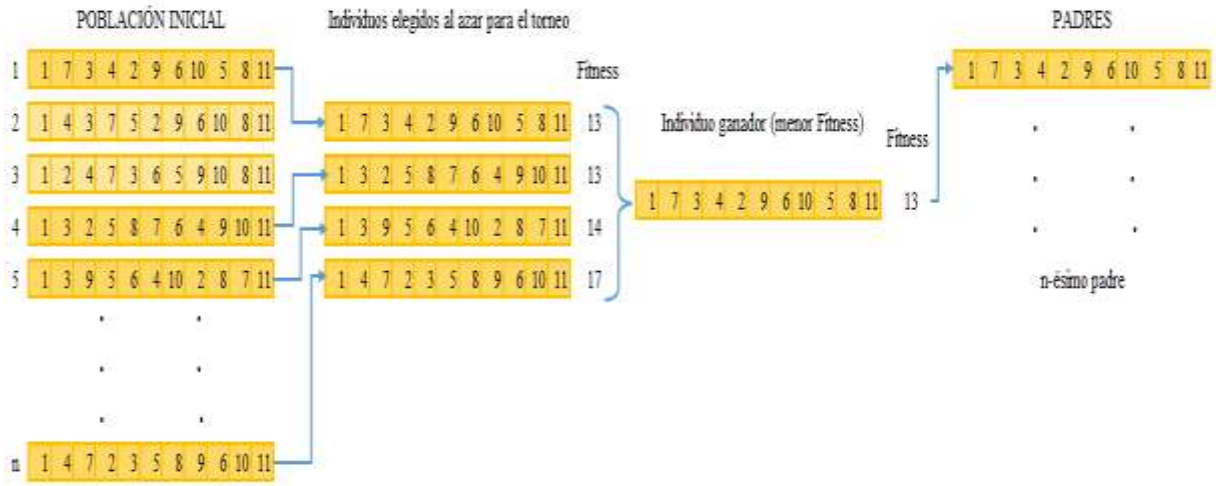

Figura 16. Operador de selección para el ejemplo del proyecto. Generado para el ejemplo de Morillo, Moreno y Díaz (2014). Metodologías Analíticas y Heurísticas para la Solución del Problema de Programación de Tareas con Recursos Restringidos (RCPSP): una revisión. Parte 1. Ingeniería y Ciencia. Pág. 250

9.3.3 Operadores de cruce. El mecanismo de generación de nuevos individuos es mediante el operador de cruce de un punto que, según Montoya, Gutierrez y Pirachicán (2010) obtiene buenos resultados en menor tiempo computacional en comparación con el de dos puntos y el de cruce uniforme; este se implementa a través de las generaciones del algoritmo.

La probabilidad de cruce empleada es del 80%, la cual fue determinada con base en la revisión de literatura ya que los autores Alcaraz y Maroto (2001) utilizaron esta probabilidad en el desarrollo del Algoritmo Genético para resolver el RCPSP obteniendo buenos resultados.

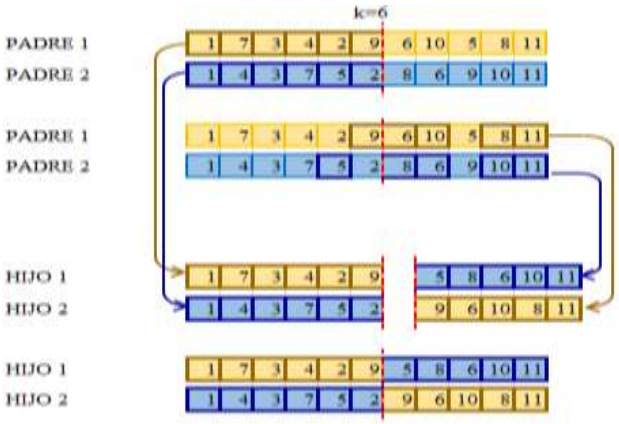

Figura 17. Operador de cruce para el ejemplo del proyecto. Generado a para el ejemplo de Morillo, Moreno y Díaz (2014). Metodologías Analíticas y Heurísticas para la Solución del Problema de Programación de Tareas con Recursos Restringidos (RCPSP): una revisión. Parte 1. Ingeniería y Ciencia. Pág. 250

En la matriz de Padres de tamaño n bajo una probabilidad del 80% se emparejan al azar los padres teniendo la precaución de no emparejar a un padre consigo mismo, los padres que no sean seleccionados para emparejarse pasan a ser ellos mismos los hijos para la siguiente generación. Para el cruce de dos padres se selecciona una posición al azar k=6 de la Lista de Actividades (ver Figura 17) en la cual se realiza el cruce por un punto, es necesario aclarar que no se selecciona ni

la primera ni la última posición de la Lista de Actividades para hacer el cruce. El Hijo 1 se genera tomando el segmento de la lista de actividades del Padre 1 que va desde la primera posición hasta la k-ésima posición, para el ejemplo hasta la sexta posición 1,7,3,4,2,9 y se completa asignando las actividades faltantes en orden de la secuencia del Padre 2 5,8,6,10,11(posiciones resaltadas, ver Figura 17) se unen la primera parte y la segunda para obtener al Hijo 1 1,7,3,4,2,9,5,8,6,10,11. El Hijo 2 se genera tomando las primeras 6 posiciones del Padre 2 1,4,3,7,5,2 y asignando las actividades faltantes en orden de secuencia que aparecen en el Padre 1 9,6,10,8,11(posiciones resaltadas, ver Figura 17) al unir ambos segmentos se obtiene el Hijo 2 1,4,3,7,5,2,9,6,10,8,11. Este método de cruce garantiza factibilidad de precedencia en los Hijos obtenidos.

9.3.4 Operadores de mutación. La finalidad de implementar este operador es garantizar la diversidad en el espacio de soluciones, en el Algoritmo Genético se emplea el Recocido Simulado como operador de mutación para refinar la búsqueda de soluciones con el objetivo de escapar de óptimos locales, en la revisión de literatura se encontró que Alcaraz y Maroto (2001) utilizaron en su trabajo de investigación una probabilidad de mutación de 1% y 5%, sin embargo en el presente trabajo se realiza con una probabilidad de mutación de 7% con el objetivo de tener mayor probabilidad de existencia de esta y verificar la incidencia del Recocido Simulado como operador.

9.3.4.1 Recocido Simulado. Para determinar los valores de los componentes principales del Recocido Simulado fue necesario plantear tres diferentes escenarios de acuerdo a la revisión de literatura (los cuales se decidieron llamar: HGA-1, HGA-P4 y HGA-P5), estos son combinaciones de los parámetros y se muestran en la tabla 4, para tal fin se elige la instancia J12048_10 (Dado que pertenece a la instancia de tamaño J120, la cual tiene mayor cantidad de actividades y a su vez

consume cierta cantidad de cada tipo de recurso. El número 48_10 se refiere a que hace parte de la instancia 48 en la clasificación 10 dentro de la librería PSPLIB), posteriormente se realiza una prueba de hipótesis de diferencia de medias para el Makespan y el Tiempo Computacional de los tres diferentes escenarios. (ver Apéndice C.). La librería PSPLIB, es una biblioteca virtual abierta en la cual están almacenadas las instancias con las mejores soluciones encontradas hasta el momento en la literatura para el RCPSP y sus extensiones.

Tabla 4.

Parámetros para hallar el Makespan

SA	**12048-10 HGA-1**	**12048-10 HGA-P4**	**12048-10 HGA-P5**
Temp. Inicial	8	20	400
Tamaño vecind.	10	50	100
Ciclos	3	10	30
α (Veloc. Enfriam)	25%	50%	70%
GA			
Población inicial	50	50	50
Grupo torneo	4	4	4
# Generaciones	20	20	20
Pc	0,8	0,8	0,8
Pm	0,05	0,05	0,05

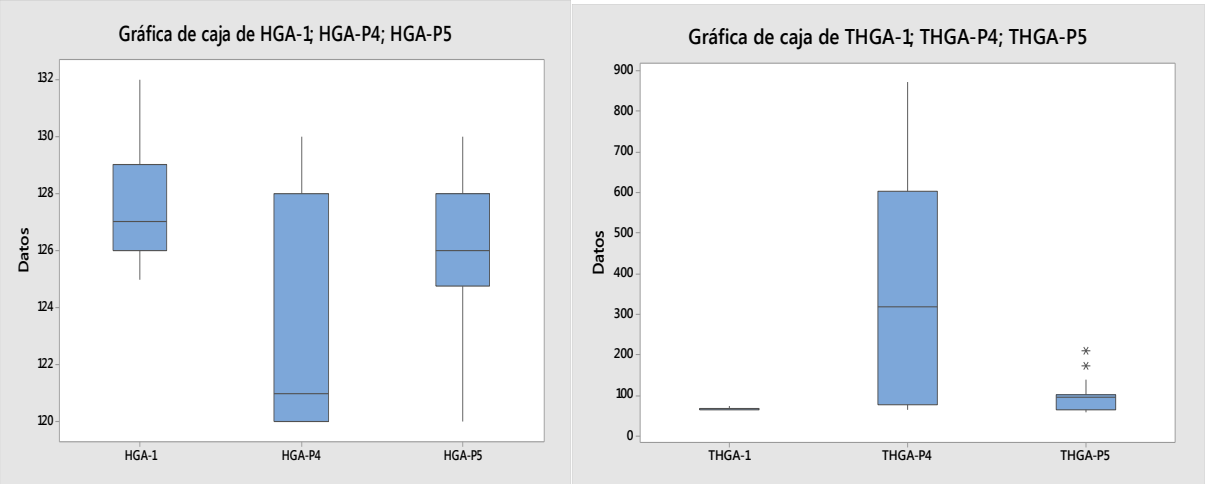

Figura 18. Gráfica de caja para Makespan y Tiempo Computacional de HGA-1, HGA-P4 y HGA-P5.

- Temperatura inicial: De acuerdo a la literatura los autores Alcaraz et al. (2001) utilizaron una temperatura inicial de 8, sin embargo, en esta investigación se realizaron 2 valores adicionales para este factor con la finalidad de determinar el valor que mejora el Makespan, posteriormente se probó con dos valores adicionales 20 y 400. a partir del análisis se decide como valor de temperatura 20, debido a que arroja menor Makespan.
- Función de decremento: se utilizó una velocidad de enfriamiento de 50%, basado en Alcaraz et al. (2001) la cual se halló de la misma forma que el valor de temperatura inicial, mediante tres escenarios, la cual arrojó buenos resultados para el Makespan y tiempo computacional.
- Criterio de parada: se determinó 10 ciclos como criterio de parada, ya que al aumentar el número de ciclos también aumenta el tiempo computacional y no se evidencia mejora en el valor del Makespan.
- Estructura de vecindario: para generar una solución vecina se parte de una Solución Inicial, se elige una posición, se toma dicha actividad y se cambia de posición conservando el

criterio de precedencia. En el ejemplo (ver Figura 19) la posición $k_1 = 5$ fue elegida al azar la cual contiene la actividad 5, se detecta su predecesora inmediata que es la actividad 3 y su sucesora inmediata que es la actividad 8, se identifican sus posiciones dentro de la Lista de Actividades que son la tercera y décima posición respectivamente, en el ejemplo aparecen sombreadas las actividades que se encuentran entre la 3 y la 8 sin incluirse ellas mismas, dentro de estas posiciones se eligió la posición $k_2 = 8$ al azar y es la posición que ocupará la actividad 5 seleccionada anteriormente desde la quinta posición, el segmento de Lista de actividades comprendida entre la posición $k_1 + 1 = 6$ hasta k_2=8 (2,9,6) es desplazada una posición hacia la izquierda de la Lista de Actividades. Generándose así la Solución vecina. Este método de estructura de vecindario garantiza factibilidad de precedencia en la solución generada.

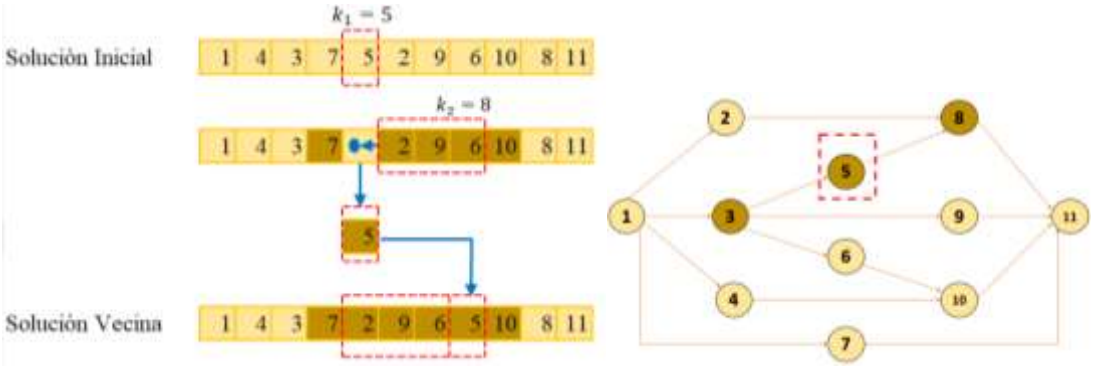

Figura 19. Operador de mutación para el ejemplo del proyecto. Generado a para el ejemplo de Morillo, Moreno y Díaz (2014). Metodologías Analíticas y Heurísticas para la Solución del Problema de Programación de Tareas con Recursos Restringidos (RCPSP): una revisión. Parte 1. Ingeniería y Ciencia. Pág. 250

9.4 Procedimiento para el cálculo del Makespan

Para el cálculo del Makespan para el Problema de Programación de Proyectos con Restricción de Recursos (RCPSP), la asignación de las actividades se realizó mediante un programa de secuenciación en serie activo, mencionado en la sección 7.1 Representación de la solución, dado que se programa cada actividad en su hora de inicio más temprana sin violar la restricción de precedencia y a su vez la de recursos, en este, se reorganizan las actividades y no se permiten realizar más cambios globales a la izquierda de manera que ese es el Makespan del proyecto, esto se visualiza en la siguiente figura 20.

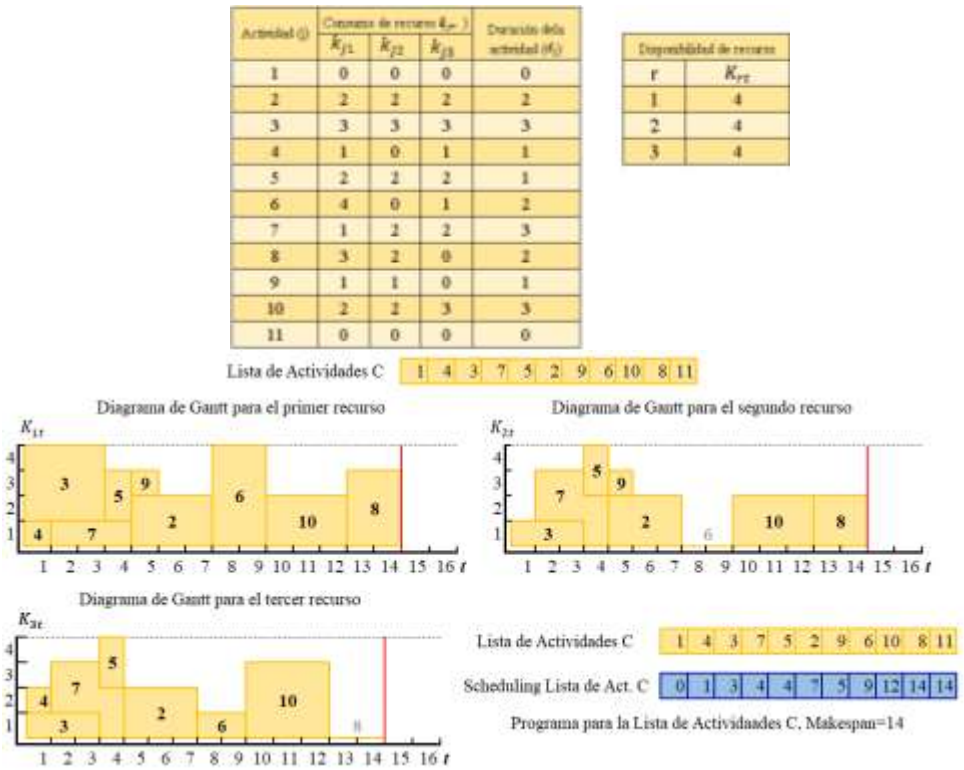

Figura 20. Diagrama de Gantt para el ejemplo del proyecto. Generado a partir del ejemplo de Morillo, Moreno y Díaz (2014). Metodologías Analíticas y Heurísticas para la Solución del Problema de Programación de Tareas con Recursos Restringidos (RCPSP): una revisión. Parte 1. Ingeniería y Ciencia. Pág. 250

En el Diagrama de Gantt se visualiza la programación de una lista de actividades factible de restricción de recursos, en el ejemplo (ver figura 20) se programa la Lista de Actividades C, para elaborar el diagrama de Gantt, en el eje horizontal se representa el horizonte de tiempo t y en el eje vertical el consumo de recurso k_{jr}(recurso tipo r para la actividad j), se dibuja una línea horizontal punteada a lo largo del horizonte de tiempo a la altura de $K_{rt} = 4$ para indicar la disponibilidad de recurso, el proyecto está conformado por tres tipos de recurso, por lo cual se realizan tres diagramas de Gantt, el primero corresponde al primer tipo de recurso ($K_{1t} = 4$), el segundo corresponde al segundo tipo de recurso ($K_{2t} = 4$), el tercero corresponde al tercer tipo de recurso ($K_{3t} = 4$). En la Figura 20 se observa el consumo de recurso para cada actividad y la duración de cada una de ellas, cada actividad se representa en el diagrama de Gantt como un bloque de ancho d_j y de alto k_{jr}.

Dada la Lista de Actividades C, se programa cada una en el orden de la secuencia de la Lista de Actividades tan pronto como sea posible, así para el Diagrama de Gantt del primer recurso, la actividad 1 no tiene duración ni consumo de recurso por lo cual no se visualiza en el diagrama, posteriormente se programa la actividad 4 la cual tiene duración $d_4 = 1$ y consumo del primer recurso $k_{11} = 1$ observándose así un bloque cuadrado en la parte inferior izquierda, a continuación se programa la actividad 3 la cual genera un bloque de ancho $d_3 = 3$ y alto $k_{31} = 3$ el cual se ubica lo más abajo y a la izquierda posible en el diagrama, Así sucesivamente se programa las demás en orden de la Lista de Actividades. Los tres Diagramas se realizan simultáneamente pues una actividad puede parecer factible para ser programada lo antes posible en un periodo de tiempo dado para el primer tipo de recurso pero en ese periodo de tiempo en el segundo recurso puede no ser factible de consumo del segundo recurso por lo cual se debe desplazar hacia la derecha dicha actividad hasta el mínimo periodo de tiempo donde sea factible de consumo de los tres tipos de

recurso simultáneamente, un ejemplo de lo anterior es la actividad 9 la cual de acuerdo al consumo del primer recurso puede aparentemente programarse en el periodo de tiempo t=4 ya que sería el más temprano posible, sin embargo se ha programado en t=5 dado que en el Diagrama de Gantt para el segundo recurso vemos que no alcanzaría éste tipo de recurso para ejecutar dicha actividad en t=4, caso contrario ocurre en t=5. La programación mostrada en el diagrama de Gantt también respeta la restricción de precedencia.

En la misma figura 20, se muestra el vector Scheduling de la Lista de Actividades C el cual corresponde al tiempo de finalización de cada actividad, obteniendo, así como Makespan la última posición de dicho vector, para el ejemplo Makespan = 14.

9.5 Diagrama de flujo del Algoritmo Híbrido Genético (AHG)

La siguiente es la estructura del Algoritmo Híbrido Genético desarrollada en esta investigación.

1. Codificación: El tipo de codificación utilizada es la representación de un vector que representa la lista aleatoria de actividades en la cual se cumple la restricción de precedencia.
2. Población inicial: La población inicial es obtenida aleatoriamente mediante el Algoritmo Genético.
3. Evaluación fitness: Se calcula el Makespan para cada cromosoma en cada generación, el procedimiento es descrito en la sección 9.4
4. Selección: En cada generación se selecciona los individuos por el método de selección por torneo (torneo de 4 individuos).

5. Cruce: Se obtiene la nueva generación cambiando la posición de las actividades, mediante la implementación del operador de cruce de un punto, respetando la precedencia entre actividades y la restricción de recursos.
6. Mutación: Con la finalidad de generar diversidad en la población, se utilizó el Recocido Simulado como operador de mutación.
7. Criterio de parada: se determinó como criterio de parada el número de generaciones.

En la Figura 21 se evidencia el diagrama de flujo para el Algoritmo Híbrido Genético (AHG) y en la Figura 22 el diagrama de flujo para el Recocido Simulado (SA).

Figura 21. Diagrama de flujo del Algoritmo Híbrido Genético

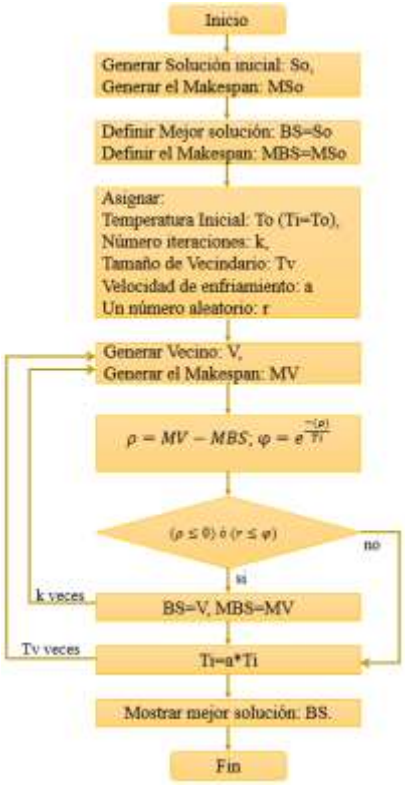

Figura 22. Diagrama de flujo del Algoritmo de Recocido Simulado

10. Validación del Algoritmo Propuesto

Para analizar el comportamiento del Algoritmo Híbrido Genético (AHG) en el Problema de Programación de Proyectos con Restricción de Recursos (RCPSP) se realizó un diseño de experimentos 2^k (ver Apéndice D). El algoritmo se desarrolló en Matlab versión R2019b y el diseño factorial se ejecutó en el software estadístico Minitab®18. Las instancias utilizadas fueron las encontradas en la biblioteca virtual PSPLIB(ver Apéndice E), la cual es una librería de benchmark para el problema RCPSP, creada por Kolisch y Sprecher en 1996, la cual contiene instancias de tamaño 30, 60, 90 y 120, denominadas J30, J60, J90 y J120 respectivamente (esto

quiere decir instancia con 30 actividades, 60 actividades, 90 actividades y 120 actividades; sin incluir las actividades dummy, las cuales no consumen tiempo ni recursos), las tres primeras instancias tienen un total de 48 instancias, pero estas a su vez están conformadas por 10 subconjuntos, por lo tanto estas tienen un total de 480 instancias cada una, mientras que J120 está conformada por 60 instancias, al igual que las anteriores también tiene un subconjunto de 10 para un total de 600 instancias; con 4 tipo de recursos cada una, en la librería están organizadas por cantidad de J y a su vez de menor a mayor consumo de recursos, las primeras consumen solo un tipo de recurso en cada actividad y a medida que se avanza en la lista de las instancias, aumenta el consumo de los recursos (es decir, las últimas consumen determinada cantidad de cada tipo de recurso). Los cuales se pueden observar en el Apéndice E.

Tabla 5.

Instancias del RCPSP

	Tamaño	**Nombre instancia**
Clase 1	J30	J301_1 - J3048_10
Clase 2	J60	J601_1 - J6048_10
Clase 3	J90	J901_1 - J9048_10
Clase 4	J120	J120_1 - J12060_10

El diseño experimental fue un 2^3 con 5 réplicas, los factores y niveles se muestran en la siguiente tabla:

Tabla 6.

Factores y niveles del Diseño Experimental

Factores	Nivel	Valores
	Bajo (-)	Alto (+)
Población inicial	50	100
Número de generaciones	20	100
Probabilidad de mutación	0,01	0,07

A continuación, se presenta el Análisis de Varianza de una instancia de cada clase, la cual representan el comportamiento en común de las instancias que pertenecen a cada tamaño. En el Apéndice D (archivo de Excel) se encuentra las soluciones del Diseño Factorial para cada una de las instancias que se describen en este capítulo.

10.1 Análisis de Varianza de Tiempo Computacional para la instancia J3048_10

Tabla 7.

Análisis de Varianza del Tiempo Computacional para la instancia J3048_10

Fuente	GL	SC Ajust.	MC Ajust.	Valor F	Valor p
Modelo	7	2686,23	383,75	88,10	0,000
Lineal	3	2511,30	837,10	192,17	0,000
Población	1	269,74	269,74	61,93	0,000
Generación	1	2138,10	2138,10	490,84	0,000
P. mutación	1	103,46	103,46	23,75	0,000
Interacciones de 2 términos	3	170,98	56,99	13,08	0,000
Población*Generación	1	114,07	114,07	26,19	0,000
Población*P. mutación	1	5,14	5,14	1,18	0,286
Generación*P. mutación	1	51,77	51,77	11,89	0,002
Interacciones de 3 términos	1	3,95	3,95	0,91	0,348

Fuente	GL	SC Ajust.	MC Ajust.	Valor F	Valor p
Población*Generación*P. mutación	1	3,95	3,95	0,91	0,348
Error	32	139,39	4,36		
Total	39	2825,63			

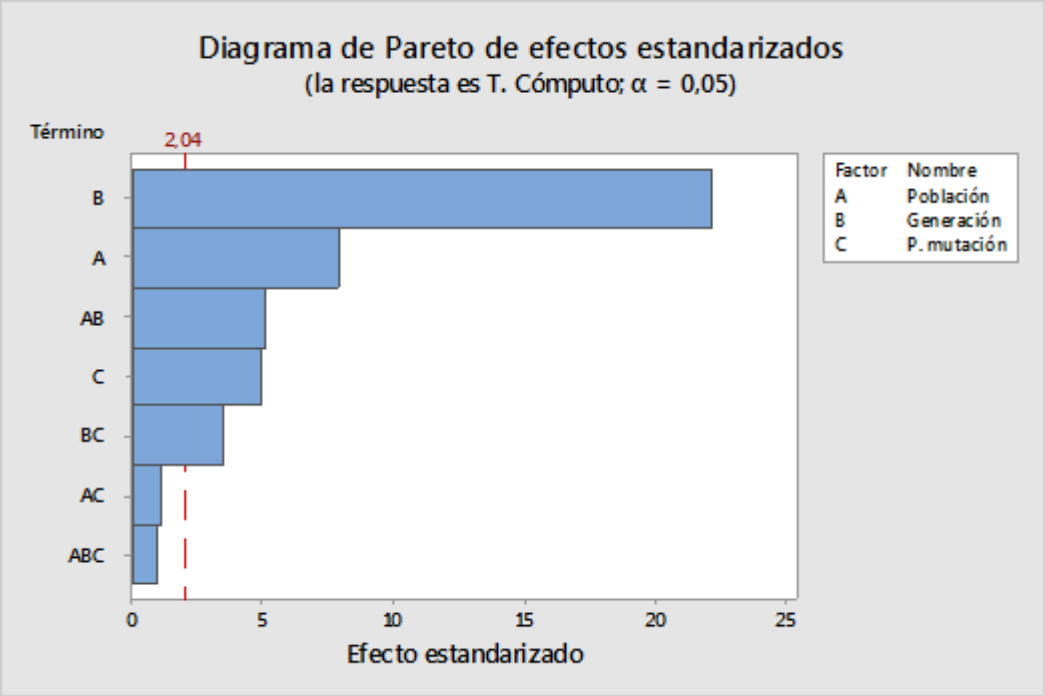

Figura 23. Diagrama de efectos de Pareto para la instancia J3048_10

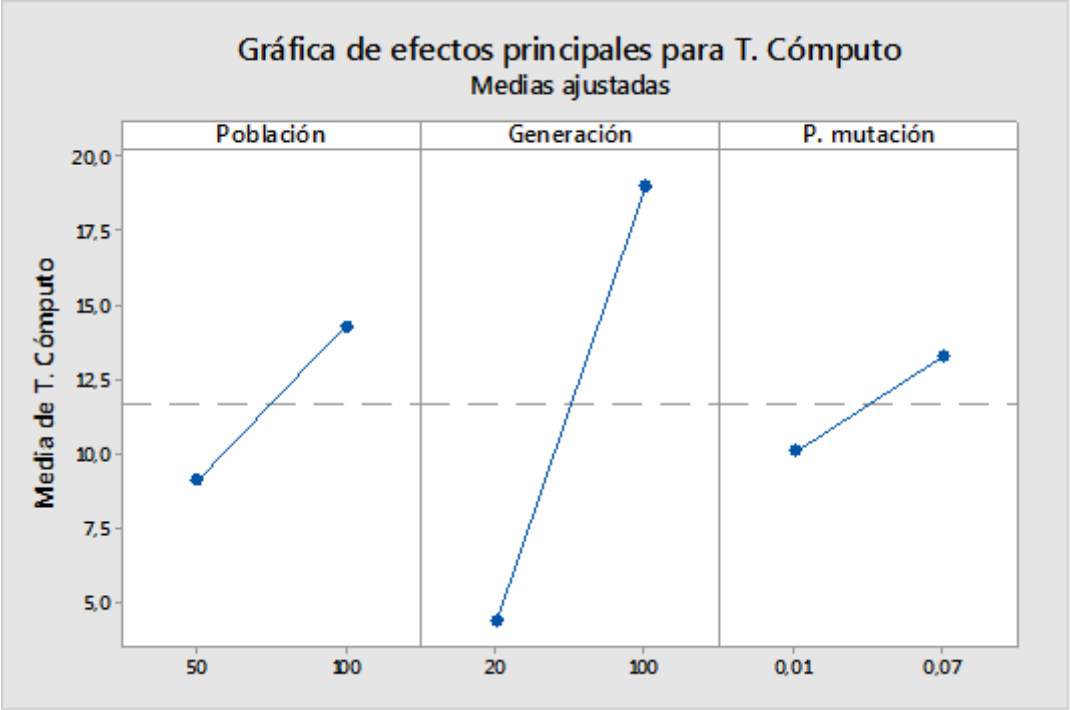

Figura 24. Gráfica de efectos principales para la instancia J3048_10

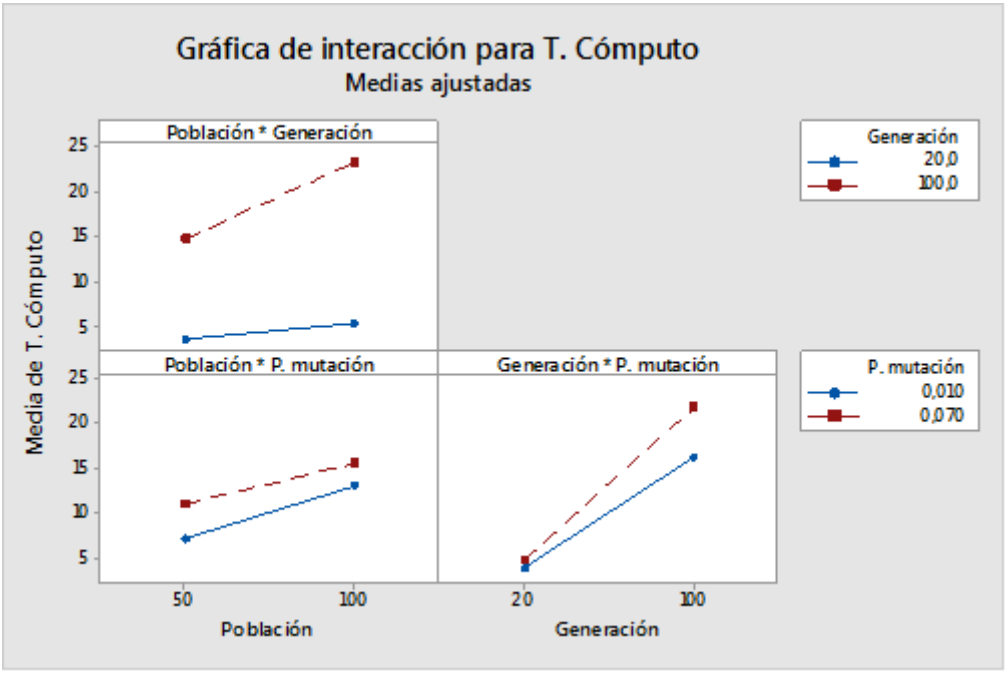

Figura 25. Gráfica de interacción del Tiempo Computacional de la instancia J3048_10

Dado que con las combinaciones de los tres factores alcanzó los resultados de Makespan de la mejor solución encontrada en la librería PSPLIB, se procede a realizar el Análisis de Varianza de la instancia J3048_10 respecto al tiempo computacional empleado para encontrar dicha solución, en la cual se evidencia que el principal factor que incide en el Tiempo Computacional es el número de generaciones con valor p = 0,000. También los factores generación y probabilidad de mutación con un valor p=0,000 inciden en esta instancia, igualmente, las interacciones: población-generación con valor p=0,000. Para verificar el Análisis de Varianza se muestra el Diagrama de Pareto de efectos estandarizados (ver figura 23).

Se puede evidenciar los niveles de los factores (ver figura 24) que inciden en la minimización del tiempo computacional son: número de generaciones: 20, tamaño de la población: 50 y probabilidad de mutación:1%.

10.2 Análisis de Varianza de Tiempo Computacional para la instancia J6048_10

Tabla 8.

Análisis de Varianza del Tiempo Computacional para la instancia J6048_10

Fuente	GL	SC Ajust.	MC Ajust.	Valor F	Valor p
Modelo	7	16747,0	2392,4	179,28	0,000
Lineal	3	15657,7	5219,2	391,12	0,000
Población	1	2041,1	2041,1	152,96	0,000
Generación	1	12754,1	12754,1	955,77	0,000
P. mutación	1	862,4	862,4	64,63	0,000
Interacciones de 2 términos	3	1087,3	362,4	27,16	0,000
Población*Generación	1	832,0	832,0	62,35	0,000
Población*P. mutación	1	0,0	0,0	0,00	0,968
Generación*P. mutación	1	255,4	255,4	19,14	0,000

Fuente	GL	SC Ajust.	MC Ajust.	Valor F	Valor p
Interacciones de 3 términos	1	1,9	1,9	0,14	0,707
Población*Generación*P. mutación	1	1,9	1,9	0,14	0,707
Error	32	427,0	13,3		
Total	39	17174,0			

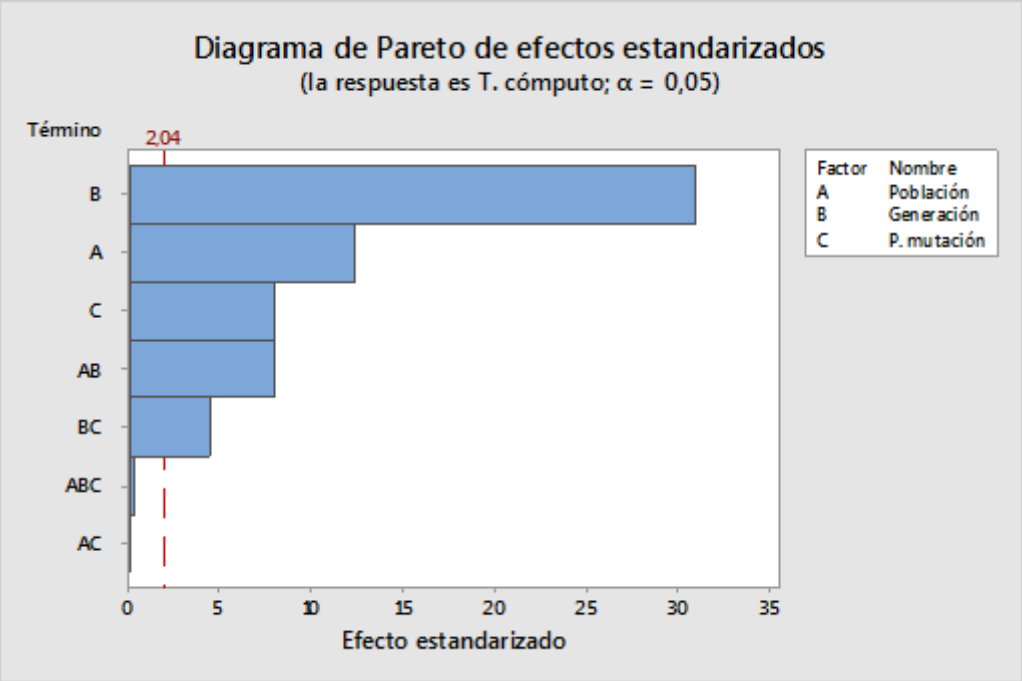

Figura 26. Diagrama de Pareto de efectos estandarizados de la instancia J6048_10

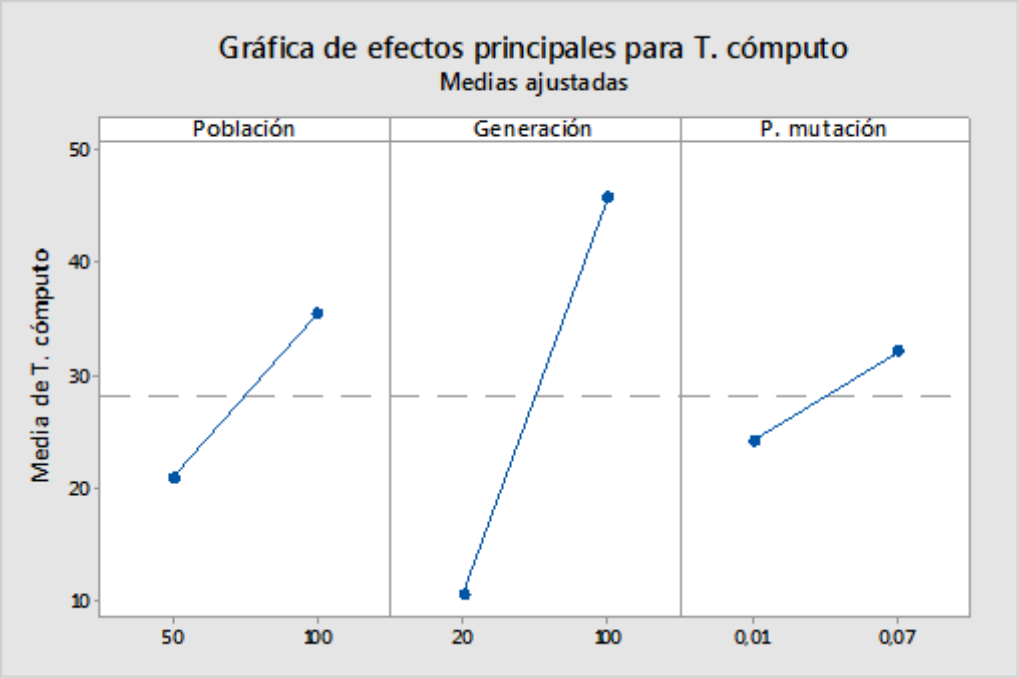

Figura 27. Gráfica de efectos principales de la instancia J6048_10

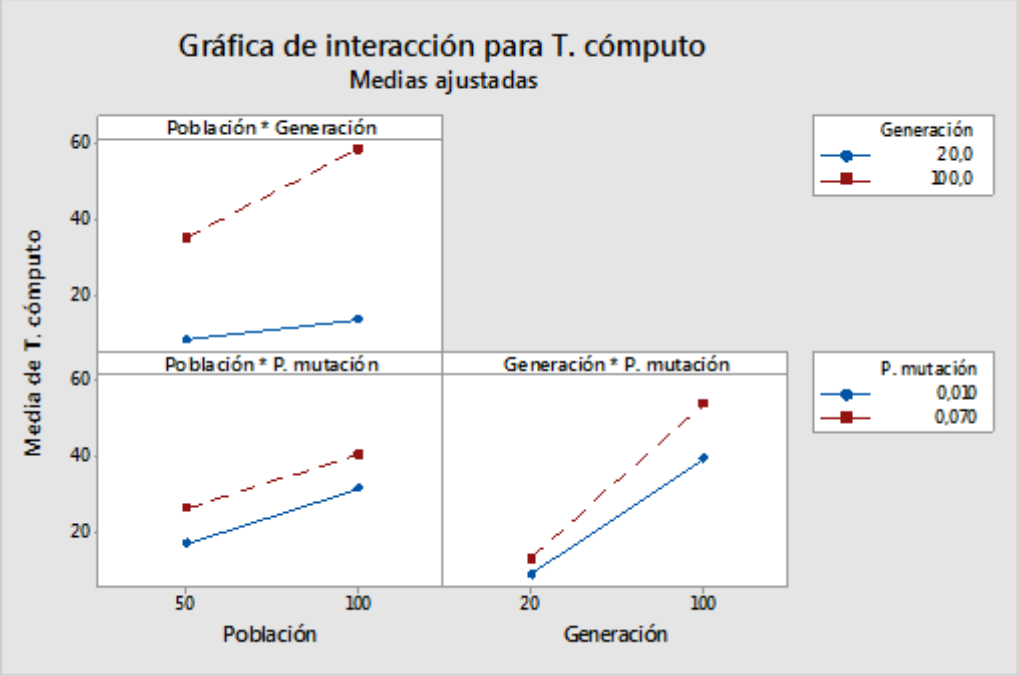

Figura 28. Gráfica de interacción para el Tiempo Computacional de la instancia J6048_10

Dado que con las combinaciones de los tres factores alcanzó los resultados de Makespan de la mejor solución encontrada en la librería PSPLIB, se procede a realizar el Análisis de Varianza de la instancia J6048_10 respecto al Tiempo Computacional empleado para encontrar dicha solución, en la cual se evidencia que el factor número de generaciones incide en este con valor p = 0,000; también la población inicial y la probabilidad de mutación con valor p = 0,000; las interacciones: población-generación con valor p =0,000 y población inicial – generación y número de generaciones – probabilidad de mutación con valor p = 0,000 inciden en el mismo. Para verificar el Análisis de Varianza se muestra el Diagrama de Pareto de efectos estandarizados (ver Figura 26)

Analizando la gráfica de efectos principales para el tiempo computacional (ver figura 27), se evidencia que los niveles de factores que inciden en la minimización del tiempo computacional son: número de generaciones: 20, tamaño de la población: 50 y probabilidad de mutación:1%.

10.3 Análisis de Varianza de Tiempo de Computacional para la instancia J9048_10

Tabla 9.

Análisis de Varianza del Tiempo Computacional para la instancia J9048_10

Fuente	GL	SC Ajust.	MC Ajust.	Valor F	Valor p
Modelo	7	102140	14591,5	254,82	0,000
Lineal	3	94301	31433,5	548,95	0,000
Población	1	12161	12161,0	212,38	0,000
Generación	1	77600	77600,0	1355,20	0,000
P. mutación	1	4540	4539,6	79,28	0,000
Interacciones de 2 términos	3	7838	2612,7	45,63	0,000
Población*Generación	1	5262	5261,6	91,89	0,000
Población*P. mutación	1	29	29,5	0,51	0,478

Fuente	GL	SC Ajust.	MC Ajust.	Valor F	Valor p
Generación*P. mutación	1	2547	2546,9	44,48	0,000
Interacciones de 3 términos	1	2	1,8	0,03	0,860
Población*Generación*P. mutación	1	2	1,8	0,03	0,860
Error	32	1832	57,3		
Total	39	103973			

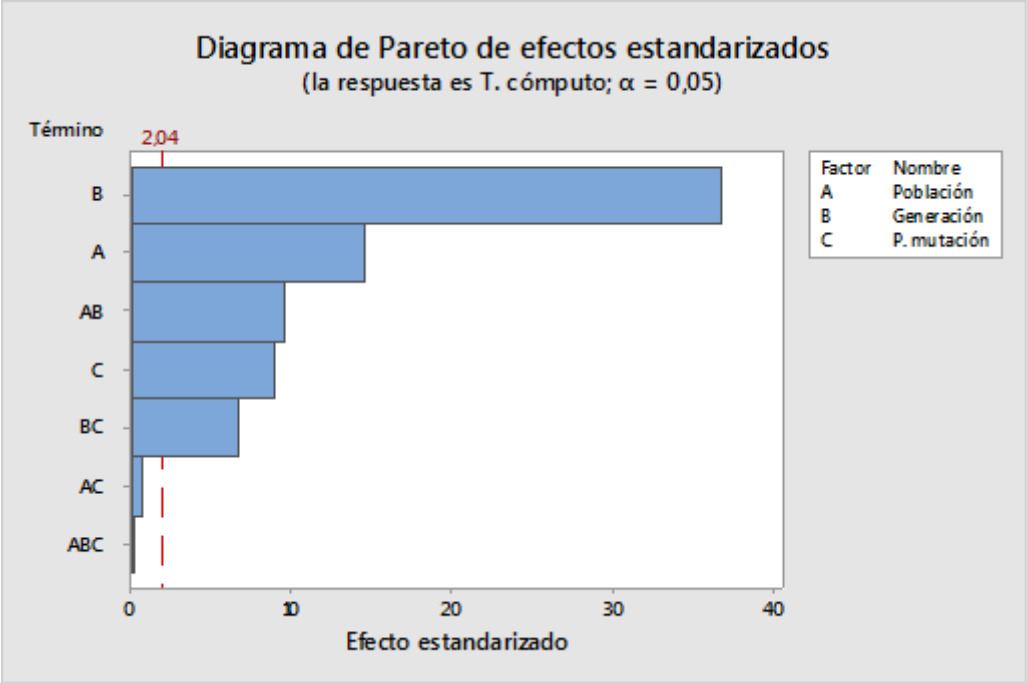

Figura 29. Diagrama de Pareto de efectos estandarizados de la instancia J9048_10

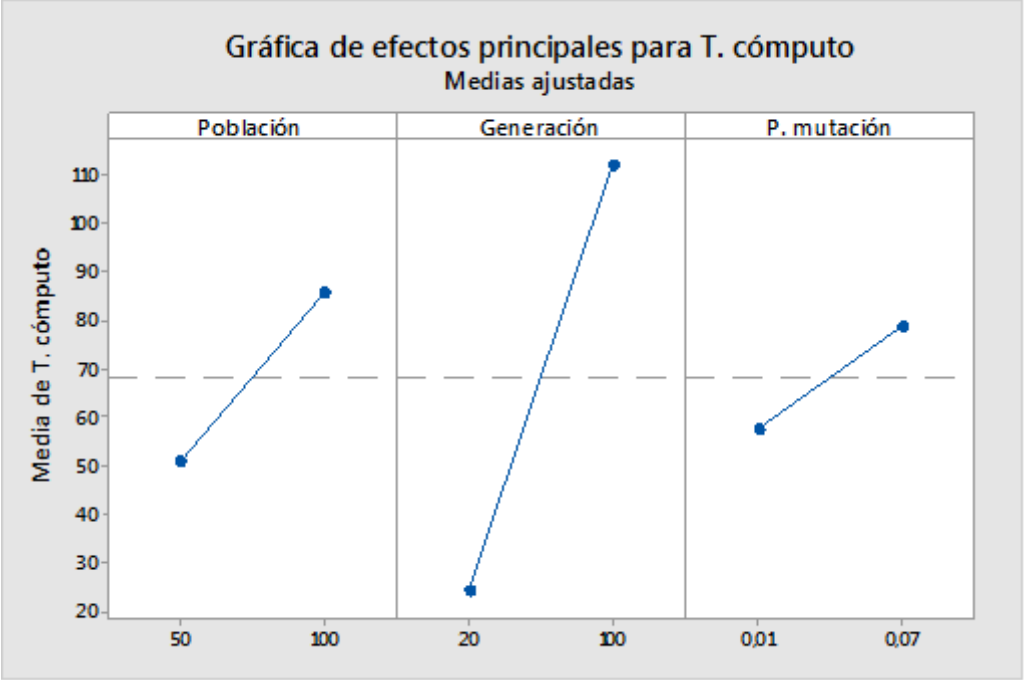

Figura 30. Gráfica de efectos principales para la instancia J9048_10

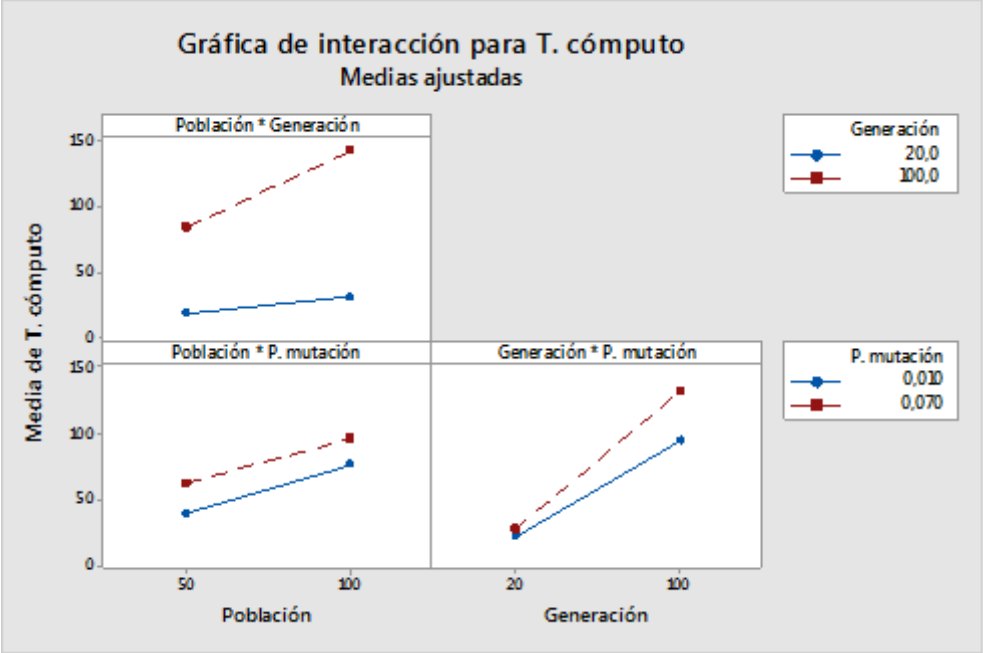

Figura 31. Gráfica de interacción para el Tiempo Computacional de la instancia J9048_10

Dado que con las combinaciones de los tres factores alcanzó los resultados de Makespan de la mejor solución encontrada en la librería PSPLIB, se procede a realizar el Análisis de Varianza de la instancia J9048_10 respecto al tiempo computacional empleado para encontrar dicha solución, en la cual se evidencia que el factor principal es el número de generaciones con valor p = 0,000; la población inicial y probabilidad de mutación con valor p = 0,000 también inciden en el mismo; las interacciones: población inicial – número de generaciones y número de generaciones y probabilidad de mutación con valor p = 0,000 inciden en el Tiempo Computacional. Para verificar el Análisis de Varianza se muestra el Diagrama de Pareto de efectos estandarizados (ver figura 29).

Analizando la gráfica de efectos principales para el tiempo computacional (ver Figura 30), se evidencia que los niveles de factores que inciden en la minimización del tiempo computacional son: número de generaciones: 20, tamaño de la población: 50 y probabilidad de mutación:1%.

10.4 Análisis de Varianza de Makespan para la instancia J12048_10

Tabla 10.

Análisis de Varianza para la instancia J12048_10

Fuente	GL	SC Ajust.	MC Ajust.	Valor F	Valor p
Modelo	7	87,375	12,4821	2,74	0,024
Lineal	3	72,075	24,0250	5,27	0,005
Tam. poblac	1	9,025	9,0250	1,98	0,169
No. Generac	1	3,025	3,0250	0,66	0,422
Prob. mutac	1	60,025	60,0250	13,16	0,001
Interacciones de 2 términos	3	12,275	4,0917	0,90	0,453
Tam. poblac*No. Generac	1	0,625	0,6250	0,14	0,714
Tam. poblac*Prob. mutac	1	0,625	0,6250	0,14	0,714

Fuente	GL	SC Ajust.	MC Ajust.	Valor F	Valor p
No. Generac*Prob. mutac	1	11,025	11,0250	2,42	0,130
Interacciones de 3 términos	1	3,025	3,0250	0,66	0,422
Tam. poblac*No. Generac*Prob. mutac	1	3,025	3,0250	0,66	0,422
Error	32	146,000	4,5625		
Total	39	233,375			

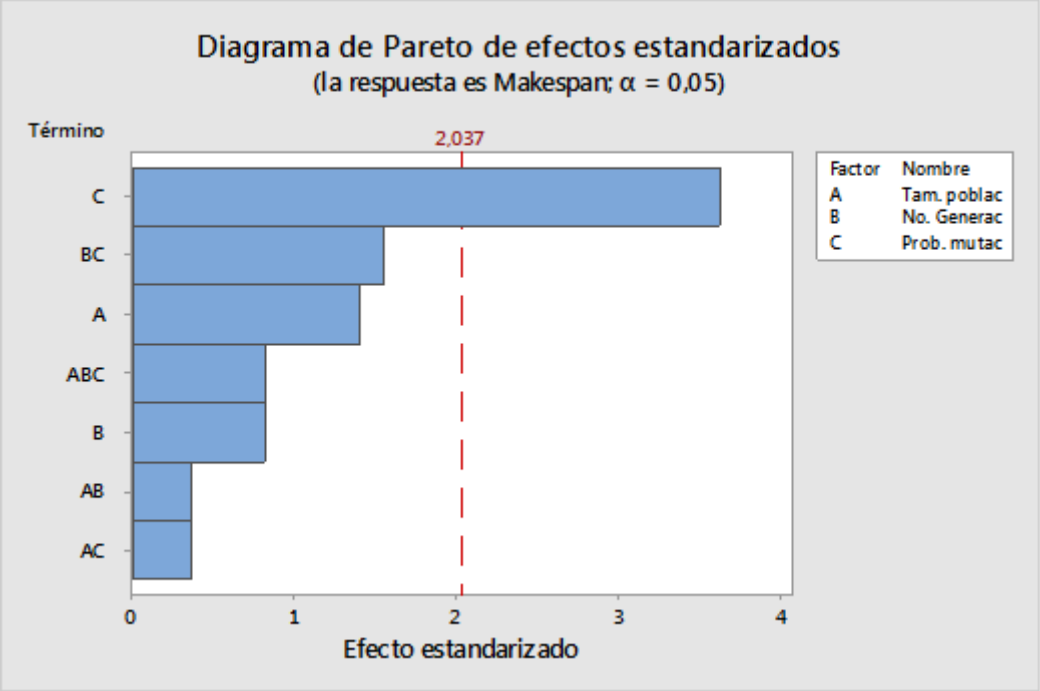

Figura 32. Diagrama de Pareto de efectos estandarizados de la instancia J12048_10

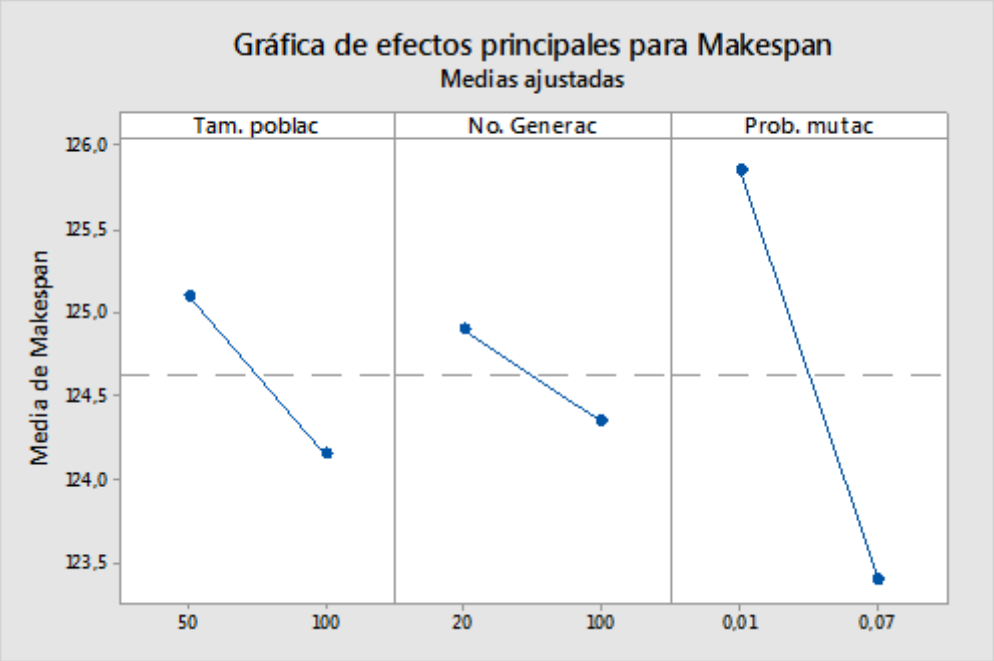

Figura 33. Gráfica de efectos principales de la instancia J12048_10

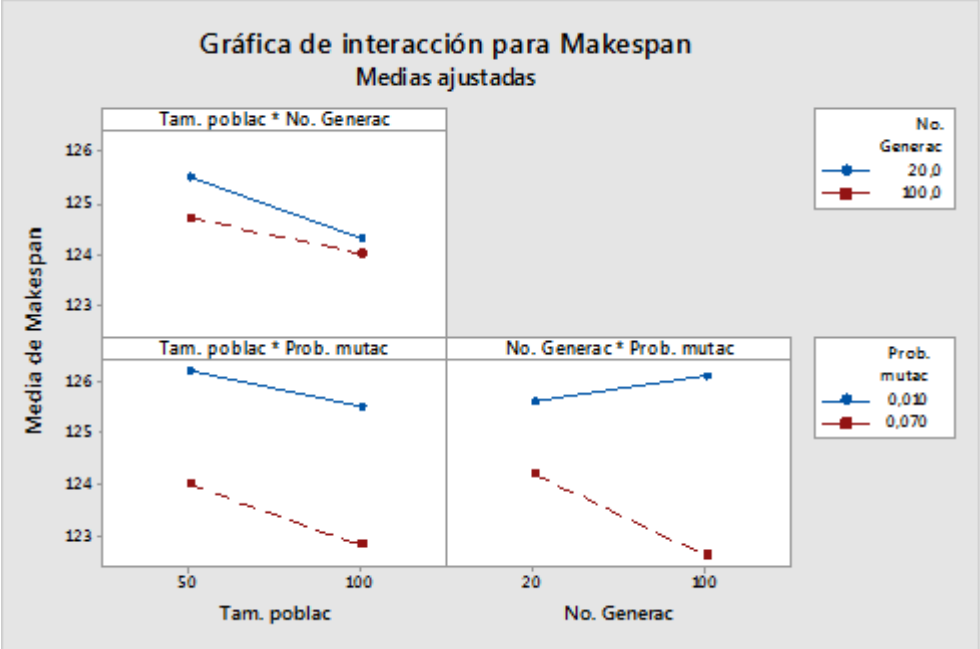

Figura 34. Gráfica de interacción para el Makespan de la instancia J12048_10

Para la instancia J12048_10 el efecto principal es la probabilidad de mutación con valor p= 0,001; mientras que el número de generaciones con valor p= 0,422 y tamaño de la población con un valor p= 0,169 no influyen significativamente en el Makespan. Las interacciones población inicial – número de generación con valor p= 0,714; población inicial- probabilidad de mutación con valor p= 0,714 y número de generaciones - probabilidad de mutación con valor p= 0,130 no influyen significativamente en el Makespan. Para verificar el Análisis de Varianza se muestra el Diagrama de Pareto de efectos estandarizados (ver Figura 32).

Analizando la gráfica de efectos principales para el Makespan (ver Figura 33), se evidencia que los niveles de factores que inciden en la minimización de este son: número de generaciones: 20, tamaño de la población: 100 y probabilidad de mutación:7%.

10.5 Análisis de Varianza de Tiempo Computacional para la instancia J12048_10

Tabla 11.

Análisis de Varianza del Tiempo Computacional para la instancia J12048_10

Fuente	GL	SC Ajust.	MC Ajust.	Valor F	Valor p
Modelo	7	2234185	319169	120,16	0,000
Lineal	3	2008277	669426	252,02	0,000
Población	1	218384	218384	82,21	0,000
Generación	1	1605366	1605366	604,37	0,000
P. mutación	1	184526	184526	69,47	0,000
Interacciones de 2 términos	3	220190	73397	27,63	0,000
Población*Generación	1	122727	122727	46,20	0,000
Población*P. mutación	1	2606	2606	0,98	0,329
Generación*P. mutación	1	94857	94857	35,71	0,000
Interacciones de 3 términos	1	5718	5718	2,15	0,152

Fuente	GL	SC Ajust.	MC Ajust.	Valor F	Valor p
Población*Generación*P. mutación	1	5718	5718	2,15	0,152
Error	32	85001	2656		
Total	39	2319185			

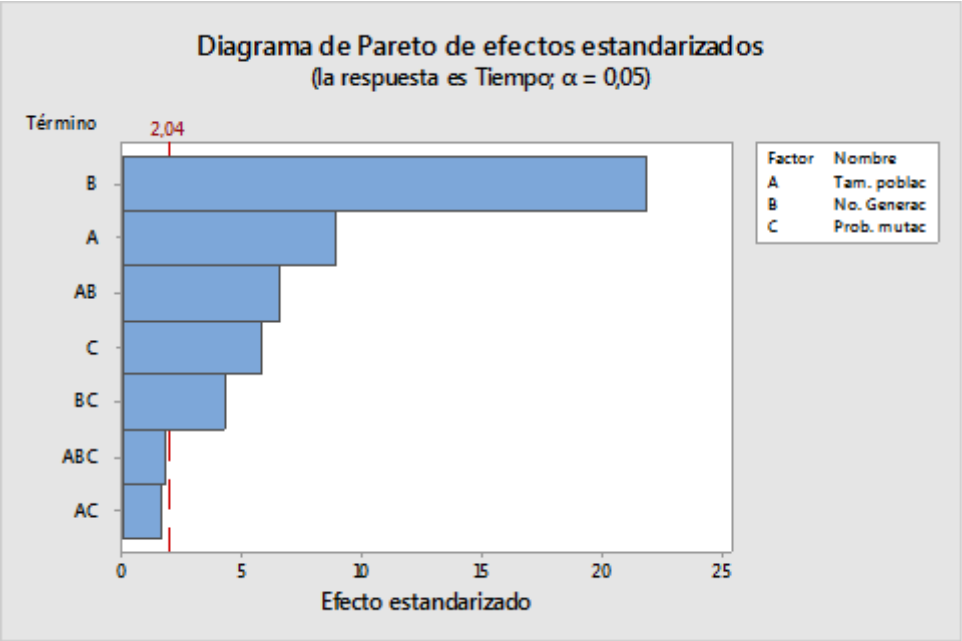

Figura 35. Diagrama de Pareto de efectos estandarizados de la instancia J12048_10

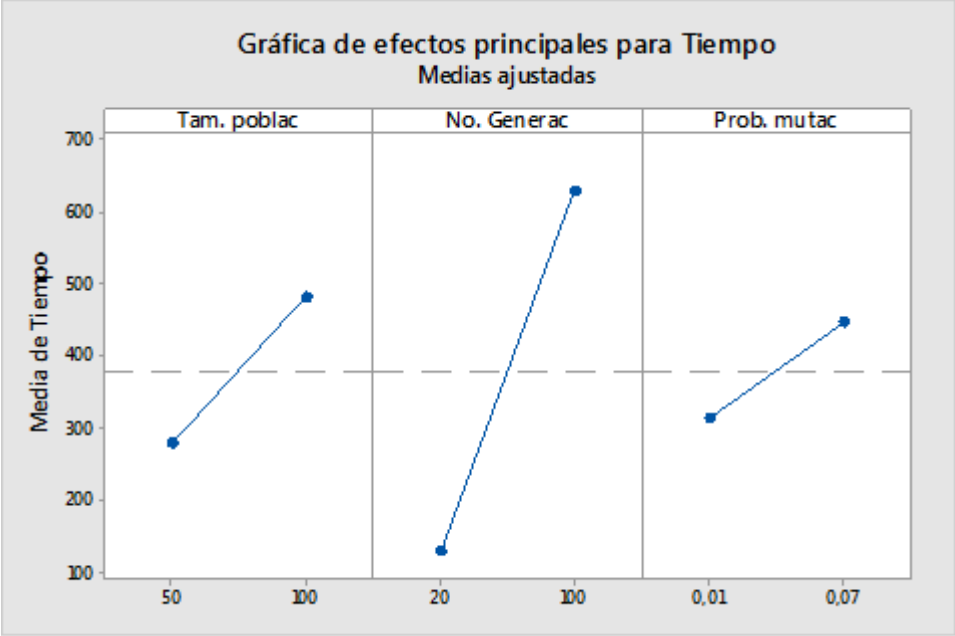

Figura 36. Gráfica de efectos principales de la instancia J12048_10

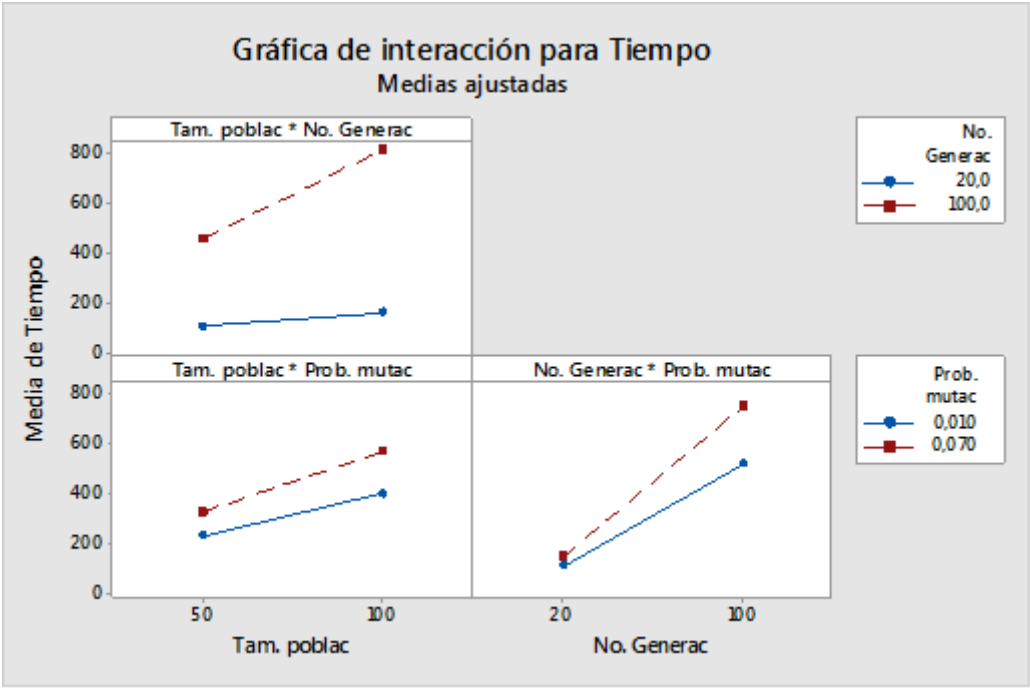

Figura 37. Gráfica de interacción para el Makespan de la instancia J12048_10

Para la instancia J12048_10 el efecto principal que incide en el Tiempo Computacional es el número de generaciones con valor p= 0,000; también el tamaño de la población con valor p= 0,000 y probabilidad de mutación con un valor p= 0,000 influyen significativamente; así como las interacciones Tamaño de la población- Número de generación con valor p = 0,000 y número de generación – probabilidad de mutación con valor p = 0,000 influyen significativamente en el Tiempo Computacional. Para verificar el Análisis de Varianza se muestra el Diagrama de Pareto de efectos estandarizados (ver Figura 35).

Analizando la gráfica de efectos principales para el Tiempo Computacional (ver Figura 36), se evidencia que los niveles de factores que inciden en la minimización de este son: número de generaciones: 20, tamaño de la población: 50 y probabilidad de mutación: 1%.

Los parámetros utilizados en el Algoritmo Híbrido Genético para cada instancia, se muestran en la siguiente tabla.

Tabla 12.

Parámetros utilizados en el Algoritmo Híbrido Genético

Instancia	Parámetro SA					Parámetro GA			
	T_0	Vec	Ciclos	α	P_0	Torn	Gen	Pc	Pm
J30	20	50	10	0,5	50	4	20	80%	1%
J60	20	50	10	0,5	50	4	20	80%	1%
J90	20	50	10	0,5	50	4	20	80%	1%
J120	20	50	10	0,5	100	4	20	80%	7%

Nota: T_0: Temperatura inicial, Vec: Tamaño de vecindario, Ciclos, α: Factor de enfriamiento, P_o: Población inicial, Torn: Grupo torneo, Gen: Número de generaciones, P_c: Probabilidad de cruce, P_m: Probabilidad de mutación.

11. Resultados

Los resultados de los algoritmos desarrollados: Algoritmo Híbrido Genético (AHG), Recocido Simulado (SA) y Algoritmo Genético (GA), se obtuvieron mediante un procesador Intel(R) Core (TM) i7-8700 CPU@ 3.20GHz 3.19GHz, Memoria (RAM) 8,00 GB y Sistema Operativo de 64 bits, procesador x64, el cual fue proporcionado por el grupo de investigación OPALO de la Escuela de Estudios Industriales y Empresariales de la Universidad Industrial de Santander. En el Apéndice G se encuentran los valores obtenidos (mejor Makespan y Tiempo Computacional obtenido en las 10 ejecuciones de cada algoritmo) por cada algoritmo con su respectivo GAP. Sin embargo, en la figura 38 Se resume el GAP de Makespan de cada uno de estos.

Tamaño	Instancia	MAKESPAN			
		PSPLIB	GAP		
			AHG	GA	SA
J30	j301_2.sm	47	0,00	0,00	0,00
	j306_10.sm	61	1,64	3,28	0,00
	j3012_10.sm	57	0,00	0,00	0,00
	j3018_10.sm	49	0,00	0,00	0,00
	j3024_10.sm	53	0,00	0,00	0,00
	j3030_10.sm	53	0,00	1,89	0,00
	j3036_10.sm	59	0,00	0,00	0,00
	j3042_10.sm	75	0,00	0,00	0,00
	j3048_10.sm	54	0,00	0,00	0,00
	j3025_6.sm	55	3,45	3,45	3,45
J60	j601_1.sm	77	0,00	0,00	0,00
	j601_10.sm	80	0,00	0,00	0,00
	j606_10.sm	74	0,00	0,00	0,00
	j6012_10.sm	79	0,00	0,00	0,00
	j6018_10.sm	97	0,00	0,00	0,00
	j6030_10.sm	86	8,14	5,81	6,98
	j6036_10.sm	77	0,00	0,00	0,00
	j6042_10.sm	87	0,00	0,00	0,00
	j6048_10.sm	70	0,00	0,00	0,00
	j6024_5.sm	76	0,00	0,00	0,00
J90	j901_1.sm	73	10,96	10,96	9,59
	j901_10.sm	90	5,56	6,67	6,67
	j906_10.sm	94	0,00	0,00	0,00
	j9012_10.sm	86	0,00	0,00	0,00
	j9018_10.sm	94	0,00	0,00	0,00
	j9024_10.sm	89	0,00	0,00	0,00
	j9030_10.sm	90	5,56	3,33	1,11
	j9036_10.sm	109	0,00	0,00	0,00
	j9042_10.sm	90	7,78	8,89	6,67
	j9048_10.sm	93	0,00	0,00	0,00
J120	j12048_10.sm	111	9,91	12,61	13,51
	j1201_10.sm	108	12,04	16,67	17,59
	j12024_10.sm	91	0,00	0,00	0,00
	j12030_10.sm	86	1,16	3,49	1,16
	j12060_1.sm	101	1,98	3,96	2,97
	j12054_2.sm	134	0,00	0,75	0,00
	j1209_10.sm	84	1,19	1,19	1,19
	j12028_5.sm	102	0,00	3,92	4,90
	j1203_10.sm	103	0,00	0,00	0,00
	j12042_7.sm	123	3,25	3,25	3,25

Figura 38. Comparación de los resultados obtenidos respecto a Makespan de los algoritmos: AHG, GA y SA.

Para las instancias de tamaño J30, en 8 de las 10 instancias evaluadas el AHG, GA y SA alcanzan el valor de la mejor solución de Makespan encontrada en la librería PSPLIB (celdas en color verde).

En las instancias de tamaño J60, en 9 de las 10 instancias evaluadas el AHG, GA y SA alcanzan el valor de la mejor solución de Makespan encontrada en la librería PSPLIB (celdas en color verde).

En las instancias de tamaño J90, en 6 de las 10 instancias evaluadas el AHG, GA y SA alcanzan el valor de la mejor solución de Makespan encontrada en la librería PSPLIB (celdas en color verde).

Finalmente, para las instancias de tamaño J120, en 4 de las 10 instancias evaluadas el AHG, GA y SA alcanzan el valor de la mejor solución de Makespan de la librería PSPLIB (celdas en color verde), también se observa que en 3 instancias el AHG presenta menor GAP comparado con los otros dos algoritmos (celdas en color azul) y en las 3 instancias restante obtiene el mismo GAP que la del Algoritmo de Recocido Simulado (color amarillo).

En la figura 39. se resume el GAP de Tiempo Computacional empleado en hallar las soluciones de Makespan, bajo las consideraciones anteriormente mencionadas.

Tamaño	Instancia	TIEMPO COMPUTACIONAL			
		PSPLIB	GAP		
			AHG	GA	SA
J30	j301_2.sm	0,11	3,7627	10,4394	2,354
	j306_10.sm	0,57	4,7919	8,5998	2,481
	j3012_10.sm	0,01	2,9273	5,7329	1,523
	j3018_10.sm	0,03	3,8663	7,6697	1,995
	j3024_10.sm	0,01	3,1009	5,8965	1,577
	j3030_10.sm	2,32	8,5175	9,0652	3,655
	j3036_10.sm	0,02	3,0764	6,1004	1,647
	j3042_10.sm	0,01	4,0638	8,0378	2,252
	j3048_10.sm	0,01	3,0023	5,9080	1,612
	j3025_6.sm	0,02	3,3129	6,5929	1,902
J60	j601_1.sm	0,01	14,2296	28,8120	7,058
	j601_10.sm	0,01	14,7696	25,9914	6,459
	j606_10.sm	0,01	12,6476	26,0496	6,402
	j6012_10.sm	0,04	7,2039	13,7468	3,409
	j6018_10.sm	0,04	9,0549	17,8085	4,829
	j6030_10.sm	0,02	16,5903	34,4106	8,375
	j6036_10.sm	124,2	7,2462	14,0424	3,487
	j6042_10.sm	0,02	15,9821	29,5126	7,799
	j6048_10.sm	0,06	7,0618	13,6412	3,324
	j6024_5.sm	0,03	7,1243	13,8744	3,336
J90	j901_1.sm	0,04	57,2245	91,1757	25,441
	j901_10.sm	0,04	64,5705	83,3193	22,083
	j906_10.sm	0,06	38,1172	64,8448	18,646
	j9012_10.sm	0,02	12,9748	24,7727	5,821
	j9018_10.sm	0,01	19,5359	37,2654	8,851
	j9024_10.sm	0,02	12,8005	25,1298	5,922
	j9030_10.sm	0,01	48,8646	66,8458	17,371
	j9036_10.sm	0,01	13,8853	26,6404	6,319
	j9042_10.sm	203,7	48,2985	58,8201	16,596
	j9048_10.sm	0,04	13,4498	25,5356	6,013
J120	j12048_10.sm	61,72	142,9676	142,9692	34,733
	j1201_10.sm	3,78	277,4467	213,6260	50,153
	j12024_10.sm	0,07	207,3468	116,2871	24,708
	j12030_10.sm	0,14	92,5177	114,3315	32,136
	j12060_1.sm	0,19	154,7562	136,2724	40,199
	j12054_2.sm	0,23	104,8864	131,1683	29,223
	j1209_10.sm	1,23	230,9276	224,1039	53,321
	j12028_5.sm	0,3	201,9193	146,3601	38,900
	j1203_10.sm	0,12	56,0292	114,4960	26,577
	j12042_7.sm	0,2	128,9848	101,6519	24,992

Figura 39. Comparación de los resultados obtenidos respecto a Tiempo Computacional de los algoritmos AHG, GA y SA.

En las instancias de tamaño J30, J60, J90 y J120 el Algoritmo de Recocido Simulado obtuvo menor GAP, el segundo valor de GAP con menor variación fue el del Algoritmo Híbrido Genético para todas las instancias de tamaño J30, J60 y J90, mientras que en la instancia de tamaño J120 obtuvo menor GAP en 3 instancias (J12030_10, J12054_2 y J1203_10), comparado con el GA.

El Algoritmo Genético es el que presenta mayor Tiempo Computacional comparado con los otros dos algoritmos. Sin embargo, al ejecutar el Algoritmo Híbrido se observa una disminución, esto se debe a que el Algoritmo de Recocido Simulado aporta en la disminución del Tiempo Computacional al ser utilizado como operador de mutación.

12. Conclusiones

Según la revisión de literatura, en las últimas tres décadas han surgido numerosas investigaciones con la finalidad de solucionar el Problema de Programación de Proyectos con Restricción de Recursos (RCPSP) mediante la implementación de Metaheurísticas Híbridas, como se observa, el algoritmo propuesto en la presente investigación en algunos casos encuentra el mejor resultado registrado en la librería PSPLIB y en otros se está muy cerca, por lo cual se recomienda diseñar nuevas instancias con tamaños mayores para comprobar el alcance del algoritmo propuesto.

Con los resultados obtenidos en el Apéndice G se observó que el Recocido Simulado obtiene el 67,5% de las veces ejecutadas el valor del Makespan de las instancias registrado en la librería PSPLIB, mientras que el Algoritmo Genético obtiene el 60% de las veces ejecutadas y el Algoritmo Híbrido Genético el 67,5% de las veces ejecutadas, sin embargo el Tiempo

Computacional empleado en cada algoritmo difiere en gran medida respecto al de la librería como consecuencia del tipo de procesador y la arquitectura de la programación empleada. Al comparar el tiempo computacional de los tres algoritmos diseñados, se observa que el Algoritmo de Recocido Simulado es el de menor valor y el más alto es el del Algoritmo Genético, pero en el Algoritmo Hibrido disminuye el Tiempo Computacional como consecuencia al operador de mutación el cual es el Recocido Simulado.

El valor de Makespan, para las instancias de tamaño J120, el Algoritmo Híbrido Genético, obtiene el 40% de las veces ejecutadas el valor de la librería, mientras que el Algoritmo Genético obtiene el 20% de las veces ejecutadas y el Algoritmo de Recocido Simulado obtiene el 30% de las veces ejecutadas la mejor solución encontrada en la librería PSPLIB.

Para las instancias de tamaño J90, tanto el Algoritmo Híbrido Genético como el Algoritmo Genético y el Recocido Simulado alcanzan el 60% de las veces ejecutadas el Valor del Makespan, para las instancias de tamaño J60 el Híbrido, GA y SA en un 90% de las veces ejecutadas, Para las instancias de tamaño J30 el Híbrido alcanza 90%, GA 80% y SA 90% de las veces ejecutadas.

13. Recomendaciones

Para futuras investigaciones considerar restricciones de interrupción, penalización por actividades tardías, ventanas de procesamiento e incluso recursos no renovables en la ejecución de las actividades, con la finalidad de acercarse más al entorno real.

Se recomienda ejecutar el algoritmo híbrido en nuevos escenarios variando el valor de sus parámetros con la finalidad de verificar las soluciones generadas y a su vez el tiempo computacional empleado en encontrar la solución.

Motivar a los estudiantes de Ingeniería Industrial para que adquieran conocimientos en distintos lenguajes de programación con la finalidad de fortalecer las habilidades propias de la academia y así puedan desarrollar proyectos bajo esta modalidad de investigación.

Profundizar en el estudio de Heurísticas y Metaheurísticas con el objetivo de ampliar las herramientas que permitan desarrollar y contribuir en esta nueva línea de investigación en el Programa de Ingeniería Industrial de la Universidad Industrial de Santander.

Referencias Bibliográficas

Aarts, E., Korst, J., & Michiels, W. (2005). Chapter 7 SIMULATED ANNEALING. *Search Methodologies*, 187-210. doi:10.1007/0-387-28356-0_7

Abbasi, B., Shadrokh, S., & Arkat, J. (2006). Bi-objective resource-constrained project scheduling with robustness and makespan criteria. *APPLIED MATHEMATICS AND COMPUTATION*, 146-152.

Alcazar, J., & Maroto, C. (2001). A Robust Genetic Algorithm for Resource Allocation in Project Scheduling. *Annals of Operations Research*, 83 - 109.

Anagnostopoulos, k., & Koulinas, G. (2012). Resource-Constrained Critical Path Scheduling by a GRASP-Based Hyperheuristic. *JOURNAL OF COMPUTING IN CIVIL ENGINEERING*, 204-213.

Argawal, R., Tiwari, M., & Mukherjee, S. (2007). Artificial immune system based approach for solving resource constraint project scheduling problem. *International Journal of Advanced Manufacturing Technology*, 584-593.

Atli, O. (2011). Tabu search and an exact algorithm for the solutions of resource-constrained project scheduling problems. *International Journal of Computational Intelligence Systems*, 255-267.

Ballestín, F. (2002). Nuevos métodos de solución del problema de secuenciación de proyectos con recursos limitados. 1-272.

Baradaran, S., Fatemi, G., Mobini, M., & Hashemin, S. (2010). A hybrid scatter search approach for resource-constrained project scheduling problem in PERT-type networks. *Advances in Engineering Software*, 966-975.

Ben-Daya, M., Duffuaa, S., Raouf, A., Knezevic, J., & Ait-Kadi, D. (2009). *Handbook of Maintenance Management.* London New York: Springer Dordrecht Heidelberg.

Bettemir, H., & Sonmez, R. (2014). Hybrid Genetic Algorithm with Simulated Annealing for Resource-Constrained Project Scheduling. *American Society of Civil Engineers.*

Bianco, L., & Caramia, M. (2012). Minimizing the completion time of a project under. *Springer-Verlag*, 361-377.

Blazewicz, J., Lenstra, J., & Rinnooy, A. (1983). SCHEDULING SUBJECT TO RESOURCE CONSTRAINTS: CLASSIFICATION AND COMPLEXITY. *Discrete Applied Mathematics*, 11-24.

Chen, A., Liang, Y., & Padilla, J. (2014). An entropy-based upper bound methodology for robust predictive multi-mode RCPSP schedules. *Entropy*, 5032-5067.

Chen, W., Shi, Y., Teng, H., Lan, X., & Hu, L. (2010). An efficient hybrid algorithm for resource-constrained project scheduling. *Information Sciences, 180*, 1031-1039. doi:10.1016/j.ins.2009.11.044

Coelho, J., & Vanhoucke, M. (2018). An exact composite lower bound strategy for the resource-constrained. *Computers and Operations Research*, 135-150.

Cthourou, H., & Haouari, M. (2008). A two-stage-priority-rule-based algorithm for robust resource-constrained project scheduling. *Computers and Industrial Engineering*, 183-194.

Cuberos Gallardo, M. (2015). Algoritmo de Recocido Simulado para la mejora de la eficiencia de una terminal multimodal. *Grupo Ingeniería de Organización*, 1-147.

De Ryck, B., & Herroelen, W. (1999). Multi-mode resource-constrained project scheduling problem with generalized precedence relations. *European Journal of Operational Research*, 538-556.

Debels, D., & Vanhoucke, M. (2007). A decomposition-based genetic algorithm for the resource-constrained project-scheduling problem. *Operations Research*, 457-469.

Delgoshaei, A., Mohd Ariffin, K., Bin Baharudin, H., & Leman, Z. (2015). Minimizing makespan of a resource-constrained scheduling problem: A hybrid. *International Journal of Industrial Engineering Computations*, 503-520.

Demeulemeester, K., & Herroelen, W. (2002). Project Scheduling: A Research Handbook. En K. Demeulemeester, & W. Herroelen, *Project Scheduling: A Research Handbook* (págs. 203 - 339). Lovania, Bélgica.: Frederick S. Hillier, Series Editor .

Dowsland, K., & Adenso, D. (2003). Heuristic design and fundamentals of the Simulated Annealing. *Revista Iberoamericana de Inteligencia Artificial*, 93-102.

Eshraghi, A. (2016). International Journal of Industrial Engineering Computations. *International Journal of Industrial Engineering Computations, 7*, 205-216. doi:10.5267/j.ijiec.2015.11.001

Fahmy, A., Hassan, T., & Bassioni, H. (2014). Improving RCPSP solutions quality with Stacking Justification – Application with particle swarm optimization. *Expert Systems with Applications*, 5870 - 5881.

Ferland, J., Amaya, J., & Djuimo, M. (2007). Application of a particle swarm algorithm to the capacited open pit mining problem. *Studies in Computational Intelligence*, 127-133.

Fleszar, K., & Hindi, K. (2004). Solving the resource-constrained project scheduling problem by a variable neighbourhood search. *European Journal of Operational Research*, 402-413.

Gestal Pose, M. (2014). Introducción a los Algoritmos Genéticos. En *Depto. Tecnolog´ıas de la Informaci´on y las Comunicaciones* (págs. 1-16). España: Universidade da Coruña.

Giran, O., Temur, R., & Bekdas, G. (2017). Resource constrained project scheduling by harmony search algorithm. *KSCE Journal of Civil Engineering*, 479-487.

Hartmann, S. (1997). A Competitive Genetic Algorithm for Resource-Constrained Project Scheduling. *Naval Research Logistics*, 733-750. Obtenido de http://www.bwl.uni-kiel.de/bwlinstitute/Prod/

Hartmann, S., & Briskorn, D. (2010). A survey of variants and extensions of the resource-constrained project. *European Journal of Operational Research*, 1-14.

Hartmann, S., & Kolisch, R. (2000). Experimental evaluation of state-of-the-art heuristics for the resource-constrained project scheduling problem. *European Journal of Operational Research*, 394-407.

Herroelen, W., De Ryck, B., & Demeulemeester, E. (1998). Resource-constrained project scheduling: A survey of recent developments. *Computers and Operations Research*, 279-302.

Hong, Z., & Heng, L. T. (2006). Particle swarm optimization for resource-constrained project scheduling. *International Journal of Project Managemen*, 83-92.

Jia, Q., & Guo, Y. (2016). Hybridization of ABC and PSO algorithms for improved solutions of RCPSP. *JOURNAL OF THE CHINESE INSTITUTE OF ENGINEERS*, 727-734.

Joshi, K., Jain, K., & Bilolikar, V. (2016). A VNS-GA-based hybrid metaheuristics for resource constrained project scheduling problem. *International Journal of Operational Research*, 437-449.

Kadri, R., & Boctor, F. (2018). An efficient genetic algorithm to solve the resource-constrained project scheduling problem with transfer times: The single mode case. *European Journal of Operational Research*, 454-462.

Kawan, M., Mitsuo, G., & Genji, Y. (2003). Hybrid genetic algorithm with fuzzy logic for. *Applied Soft Computing*, 174-188.

Kellenbrink, C., & Helber, S. (2015). Scheduling resource-constrained projects with a flexible project. *European Journal of Operational Research*, 379-391.

Kim, J. (2009). Improved genetic algorithm for resource-constrained scheduling of large projects. *Canadian Journal of Civil Engineering*, 1016-1027.

Kim, K., Gen, M., & Yamazaki, k. (2003). Hybrid genetic algorithm with fuzzy logic for resource-constrained project scheduling. *Applied Soft Computing*, 174-188.

Kolisch, R. (1995). Serial and parallel resource-constrained project scheduling methods revisited: Theory and computation. *European Journal of Operational Research*, 320-333. doi:10.1016/0377-2217(95)00357-6

Kolisch, R., & Hartmann, S. (2006). Experimental investigation of heuristics for resource-constrained project scheduling: An update. *European Journal of Operational Research*, 23-37.

Kuri, A. (2000). Algoritmos Genéticos. *Centro de Investigación en computación.*, 22.

Liu, S., Yung, K., & Ip, W. (2007). Genetic local search for resource constrained project scheduling under uncertainty. *International Journal of Information and Management Sciences*, 347-363.

Mahdi, M., Rabbani, M., Amalnik, M., Razmi, J., & Rahimi, A. (2009). Using an enhanced scatter search algorithm for a resource-constrained project scheduling problem. *Soft Computing*, 597-610.

Martí, R. (2003). Procedimientos Metaheurísticos en Optimización Combinatoria. *Journal of Heuristics*, 1 - 60.

Martins, S., & Ribeiro, C. (2014). METAHEURISTICS AND APPLICATIONS TO OPTIMIZATION PROBLEMS IN TELECOMMUNICATIONS. *Journal of the Operational Research Society*, 103-128.

Mckendall, J., Noble, A., & Klein, C. (2008). Scheduling maintenance activities during planned outagest at nuclear power plants. *International Journal of Industrial Engineering: Theory Applications and Practice*, 53-61.

Mendes, J., Gongalves, J., & Resende, M. (2009). A random key based genetic algorithm for the resource constrained. *Computers & Operations Research*, 92-109.

Montoya, J., Gutierres, E., & Pirachicán, C. (2010). Project scheduling with limited resources using a genetic algorithm. *International Journal of Project Management*, 619-628.

Moreno Díaz, P., Huecas Frenandez, G., Sánchez Allendes, J., & Almuneda García, M. (2007). METAHEURÍSTICAS DE OPTIMIZACIÓN COMBINATORIA:. *Revista de Ciencia, Tecnología y Medio Ambiente*.

Morillo, D., Moreno, L., & Díaz, J. (2014). Metodologías Analíticas y Heurísticas para. *Ingeniería y Ciencia*, 247-271.

Moumene, K., & Ferland, J. (2009). Activity list representation for a generalization of the resource-constrained project scheduling problem. *European Journal of Operational Research*, 46-54.

Neetesh, K., & Deo Prakash, V. (2015). A model for resource-constrained project scheduling using. *Springer-Verlag Berlin Heidelberg*.

Palacio, J., & Larrea, O. (2017). A lexicographic approach to the robust resource-constrained project scheduling problem. *International Transactions in Operational Research*, 143-157.

Paraskevopoulos, D., Tarantilis, C., & Ioannou, G. (2012). Solving project scheduling problems with resource constraints via an event list-based evolutionary algorithm. *Expert Systems with Applications*, 3983-3994. doi:10.1016/j.eswa.2011.09.062

Pastor, R. (1999). METALGORITMO DE OPTIMIZACIÓN COMBINATORIA. *Tésis Doctoral*.

Peng, W., & Wang, C. (2009). ACO for solving resource-constrained project scheduling problem. *Xitong Fangzhen Xuebao/Acta Simulata Systematica Sinica*, 1974-1978.

Pinedo, M. L. (2012). *Scheduling:Theory, algorithms y systems.* Londres: Springer New York Dordrecht Heidelberg London. .

Potgieter, I. (Diciembre de 2015). Resource-Constrained Scheduling in the Engineering-Planning Phase of Civil Engineering Projects. Stellenbosch University, Sudafrica.

Pritsker, A., Watters, L., & Wolfe, P. (2016). Multiproject Scheduling with Limited Resources: A Zero-One Programming Approach. *Management Science,*, 93-108.

Proon, S., & Jin, M. (2011). A genetic algorithm with neighborhood search for the resource-constrained project scheduling problem. *Naval Research Logistics*, 73-82.

Rajeev, S., Kurian, S., & and Brijesh, P. (2015). A modified serial scheduling scheme for resource constrained project scheduling weighted earliness tardiness problem. *International Journal of Information and Decision Sciences, 7*, 241-254. Obtenido de http://www.inderscience.com/link.php?id=71373

Ranjbar, M., & Kianfar, F. (2010). Resource-constrained project scheduling problem with flexible work profiles: A genetic algorithm approach. *Scientia Iranica*, 25-35.

Rosenfeld, R., & Irazábla, J. (2013). *Computabilidad, complejidad computacional y verificación de programas.* Buenos aires, Argentina: Editorial de la Universidad de La Plata.

Rostami, M., Moradinezhad, D., & Soufipour, A. (2014). Improved and Competitive Algorithms for Large Scale Multiple. *KSCE Journal of Civil Engineering*, 1261-1269.

Salewski, F., Schirmer, A., & Drexl, A. (1997). Project scheduling under resource and mode identity constraints: Model, complexity, methods, and application. *European Journal of Operational Research*, 88-110.

Schwindt, C., & Zimmermann, J. (2015). *Handbook on Project Management and Scheduling Vol.1*. New York Dordrecht London: Springer Cham Heidelberg.

Shao, T., & Zhijie, S. (2018). Multi-mode resource-constrained project scheduling problem with. *Computers & Industrial Engineering*, 333-347.

Shou, Y., & Wang, W. (2012). Robust optimization-based genetic algorithm for project scheduling with stochastic activity durations. *Scopus coverage*, 4049-4064.

Shukla, S., Son, Y., & Tiwari, M. (2008). Fuzzy-based adaptive sample-sort simulated annealing for resource-constrained project scheduling. *International Journal of Advanced Manufacturing Technology*, 982-995.

Šišejkovic, D. (9 de Junio de 2016). EVOLUTION OF SCHEDULING. Zagreb, Groacia.

Sprecher, A., Kolisch, R., & Drexl, R. (1995). Semi-active, active, and non-delay schedules for the resource-constrained project scheduling problem. *European Journal of Operational Research*, 94-102. doi:10.1016/0377-2217(93)E0294-8

Stanimirovic, I., Petkovic, M., Stanimirovic, M., & Ciric, M. (2009). Heuristic algorithm for single resource constrained project scheduling problem based on the dynamic programming. *Yugoslav Journal of Operations Research*, 281 - 298.

Suresh, M., Dutta, P., & Jain, k. (2011). resource constrained project scheduling problem using genetic algorithms. *International Journal of Industrial and Systems Engineering*, 251-269.

Tao, S., & Dong, Z. (2017). Scheduling resource-constrained project problem with alternative activity chains. *Computers and Industrial Engineering*, 288-296. doi:10.1016/j.cie.2017.10.027

Tesch, A. (Junio de 2015). Compact MIP Models for the Resource-Constrained Project Scheduling Problem. Berlín, Alemania.

Tseng, L., & Chen, S. (2006). A hybrid metaheuristic for the resource-constrained. *European Journal of Operational Research*, 707-721.

Valls, V., Ballestín, F., & Quintanilla, S. (2008). A hybrid genetic algorithm for the resource-constrained project scheduling problem. *European Journal of Operational Research*, 495-508.

Vanhoucke, M. (2008). Setup times and fast tracking in resource-constrained project scheduling. *Computers and Industrial Engineering*, 1062.1070.

Viveros, R., & Rivera, J. (2017). Formulaciones matemáticas y heurísticos simples para solucionar problemas de programación de proyectos con recursos limitados. 1 - 23.

Yamashita, D., & Morabito, R. (2007). Um algoritmo branch-and-bound para o problema de programação de projetos com custo de disponibilidade de recursos e múltiplos modos. *Gest. Prod.*, 545 - 555.

Zamani, R. (2004). An efficient time-windowing procedure for scheduling projects under multiple resource constraints. *Operations-Research-Spektrum*, 423-440.

Zäpfel, G., Braune, R., & Bögl, M. (2010). Metaheuristics Based on Solution Recombination. *Metaheuristic Search Concepts: A Tutorial with Applications to Production and Logistics*, 1-316.

Zeighami, V., Akbari, R., & Ziarati, K. (2013). Development of a method based on particle swarm optimization to solve resource constrained project scheduling problem. *Scientia Iranica*, 2123-2137.

Zhang, H., Li, H., & Tam, C. (2006). Permutation Based Particle Swarm Optimization. *Journal of Computing in Civil Engineering*, 141-149.

Ziarati, K., Akbari, R., & Zeighami, V. (2011). On the performance of bee algorithms for resource-constrained project scheduling problem. *Applied Soft Computing*, 3720-3733.