Desarrollo de un algoritmo genético para resolver el Problema de Programación de Proyectos con Recursos Restringidos (RCPSP) con duración de actividades aleatorias soportado en un método basado en duraciones redundantes.

Kandessa Stefany Fuentes Luna y Danna Geraldine Tautiva Quinche

Trabajo de Grado para Optar al Título de Ingeniería Industrial

Director

Néstor Raúl Ortiz Pimiento

PhD. En Ingeniería

Universidad Industrial de Santander

Facultad de Ingenierías Físico-Mecánicas

Escuela de Estudios Industriales y Empresariales

Grupo de Investigación ÓPALO

Bucaramanga

2021

AGRADECIMIENTOS

A Dios por ser mi guía, mi apoyo y el cimiento en mi vida.

A mi madre Otilia Luna López quien ha sido mi ejemplo, mi apoyo, mi amiga, mi mejor consejera en el mundo y por quien siento una gran admiración.

A mi padre Jairo Luis Fuentes Pérez que siempre me ha brindado todo el amor, ha tenido palabras de aliento y apoyo incondicional en todo mi camino

A mi hermano Luis Rafael Fuentes Luna, siempre juntos contra el mundo, el que siempre me inspira cada día a ser una mejor persona y que me enseñó que nada es imposible cuando se quiere de verdad.

A la familia Bautista Jerez que vivieron conmigo este proceso y me impulsaron a seguir en este camino, celebrando mis triunfos y apoyándome incondicionalmente

A Eduard Fabián Bautista Jerez, quien con su amor y su apoyo son la muestra de lo afortunada que soy de tenerte y de compartir mi vida contigo, gracias por ser quien me llena de alegría todos los días.

A toda mi familia en Valledupar, pues sus palabras siempre me llenaron de amor.

A mi amiga y compañera Danna Tautiva quien vivió conmigo esta etapa y la culmino a mi lado con este proyecto, gracias por todos esos momentos que pudimos compartir.

A mis apreciados amigos, Viviana Jerez, Fernando Vesga y Gourangui Silva gracias por escuchar, por amor y su apoyo.

Kandessa Stefany Fuentes Luna

A mi amor más grande, mi creador justo, fiel y verdadero que me permitió vivir este proceso lleno de aprendizaje con mucha fortaleza.

A mi familia por siempre creer en mí, apoyarme, y ser el motivo de luchar cada día.

A los compañeros y profesores que conocí en cada semestre y pasé momentos inolvidables en este viaje la universidad.

Y, por último, a Kandessa por ser una excelente persona, compañera y amiga con la que superamos cada obstáculo por medio del trabajo y apoyo en equipo.

Danna Geraldine Tautiva Quinche

Contenido

Pág.

Lista de Tablas

Pág.

Lista de Figuras

Pág.

Lista de Apéndices

Pág.

Desarrollo de un Algoritmo Genético para Resolver el Problema de Programación de Proyectos con Recursos Restringidos (RCPSP) con Duración de Actividades Aleatorias Soportado en un Método Basado en Duraciones Redundantes

---

Development of a Genetic Algorithm to Solve Programming Problems of Projects with Restricted Resources (RCPSP) With the Duration of Random Activities Supported in a Method Based on Redundant Duration.

Kandessa Stefany Fuentes Luna[1], Danna Geraldine Tautiva Quinche[2].

[1]Grupo de investigación OPALO, Escuela de Estudios Industriales y Empresariales, Universidad Industrial de Santander, Colombia. Orcid: https://orcid.org/0000-0001-6001-4314. correo electrónico: KANDESSA.FUENTES@correo.uis.edu.co

[2]Grupo de investigación OPALO, Escuela de Estudios Industriales y Empresariales, Universidad Industrial de Santander, Colombia. Orcid: https://orcid.org/0000-0002-5206-7499. correo electrónico: DANNA.TAUTIVA@correo.uis.edu.co

Resumen

Título: Desarrollo de un algoritmo genético para resolver el Problema de Programación de Proyectos con Recursos Restringidos (RCPSP) con duración de actividades aleatorias soportado en un método basado en duraciones redundantes*

Autor(es): Kandessa Stefany Fuentes Luna y Danna Geraldine Tautiva Quinche**

Palabras clave: Problema de Programación de Proyectos con Recursos Restringidos RCPSP no determinístico, Algoritmo Genético, Makespan, Duraciones Redundantes, Robustez de la Calidad, Robustez de la Solución, Valor Esperado.

Descripción:

Al abordar problemas de secuenciación de actividades, se encuentra que uno de los más estudiados es el problema de programación de proyectos con recursos restringidos (Resource Constrained Project Scheduling Problem: RCPSP), el cual consiste en secuenciar y dar orden a un conjunto de actividades sujetas a restricciones de recursos y precedencias con el objetivo de minimizar el tiempo de ejecución del proyecto (makespan). Dado que este problema está presente en cualquier tipo de industria, surge la necesidad de su investigación. Los procedimientos para abordar el RCPSP no determinístico se diseñan con base en estrategias predictivas, reactivas y proactivas. En suma, el presente trabajo resume la investigación que tuvo como objetivo desarrollar un algoritmo genético que pertenece a la fase proactiva; del cual se obtiene la mejor lista de actividades (mejor makespan), y la línea base asociada.

Posteriormente se lleva a cabo como estrategia reactiva una simulación donde se prueba la robustez de las líneas bases obtenidas. Las instancias utilizadas se encuentran en la librería PSPLIB, la cual es una librería de benchmark para el problema RCPSP, aplicando la modificación de las duraciones redundantes, el cual consiste en adicionar tiempo extra a la duración original, esto para enfrentar las eventualidades que durante la ejecución del proyecto pueden suceder. Para analizar el comportamiento de los parámetros en el Algoritmo genético propuesto se realizó un diseño de experimentos mediante un diseño cuasi experimental, es decir, que no se realizarán experimentos aleatorios, debido a que existe una correlación entre el aumento numérico de los parámetros y el porcentaje de exactitud con respecto a los resultados esperados.

Finalmente se compararon los resultados en términos de indicadores de robustez obtenidos de la programación reactiva del algoritmo genético con el método exacto utilizando los mismos parámetros.

---

* Trabajo de grado

** Escuela de Estudios Industriales y Empresariales. Director Néstor Raúl Ortiz Pimiento, PhD. En Ingeniería.

Abstract

Title: Development of a Genetic Algorithm to Solve Programming Problems of Projects with Restricted Resources (RCPSP) With the Duration of Random Activities Supported in a Method Based on Redundant Duration*

Author(s): Kandessa Stefany Fuentes Luna y Danna Geraldine Tautiva Quinche**

Key words: Non deterministic Resource Constrained Project Scheduling Problem (RCPSP), genetic algorithm, makespan, redundant duration, quality strength, solution strength, expected value.

Description:

When solving problems with sequenced activities, it is found that one of the most studied is the problem of the Resource-Constrained Project Scheduling Problem (RCPSP). This sequence and orders a group of activities subject to resources and precedence restrictions to minimize project execution time (makespan). Because this problem is present in any industry, the necessity of researching it arises. The procedure to take on the non-deterministic RCPSP is a design based on predictive, reactive, and proactive strategies. To sum up, this article condenses the investigation that had as objective the development of a genetic algorithm that is part of the proactive phase, from which the best activities list is obtained (better makespan) and the associated based line.

After that, a reactive strategy is done, this is a simulation where the strength of the obtained baselines is tested. The used instances are in the library PSPLIB, a benchmark library for the RCPSP problem, applying the modification of the redundant duration; this consists of adding extra time to the original normal time, this is to confront the problems that during the project execution may occur. To analyze the performance of the parameters in the proposed genetic algorithm, a experiment design was done with a semi experimental design, meaning, that random experiments are not done, this because exist a correlation between the numeric increase of the parameters and the percentage of accuracy according to the expected results.

Finally, the results are compared in terms of indicators of the obtained strength of the reactive programming of the genetic algorithm with the exact method using the same parameters.

---

* Bachelor Thesis

** Escuela de Estudios Industriales y Empresariales. Director Néstor Raúl Ortiz Pimiento, PhD. En Ingeniería.

Introducción

Actualmente el mundo se encuentra en constante cambio, avanza con rapidez ante la globalización que nos permite estar hoy más conectados, donde las barreras geográficas e idiomáticas han pasado a un segundo plano gracias a los avances tecnológicos que brinda la posibilidad de mejorar cada vez más la calidad humana. En pro de esta mejora se encuentra la necesidad de la utilización de los recursos de una forma más eficiente, es ahí donde entra la gestión de proyectos, siendo en la actualidad una de las disciplinas que más atención e interés ha despertado y goza de una gran acogida y difusión. Es en el entorno complejo y rápidamente cambiante de hoy donde la efectividad es el eje de la vida contemporánea, y el análisis y la toma de decisiones deben realizarse de manera ágil y oportuna. La gestión de proyectos ha demostrado ser una disciplina fundamental frente a los desafíos que se presentan en las organizaciones.

Los gerentes de proyectos buscan crear una planificación acertada de sus actividades en el momento en el que se determina realizar un proyecto, y para ello es de crucial importancia la asignación de forma eficiente del uso de los recursos finitos de cada proyecto con base a sus necesidades. Sin embargo, para el desarrollo adecuado de un proyecto es preciso aplicar correctamente una secuenciación de sus actividades, es decir, definir el orden de cada tarea que se va a ejecutar.

En consecuencia, la programación de un proyecto comienza con su planificación, luego se realiza la asignación de los recursos finitos a cada actividad con su respectiva fecha, que permite estimar la culminación del proyecto.

Es allí donde surge el problema de programación de proyectos con recursos restringidos (*Resource Constrained Project Scheduling Problem:* RCPSP), como respuesta a ciertas restricciones como los recursos limitados. Este es un problema combinatorio con tiempo de

solución *NP-hard*, que tiene como propósito ulterior, identificar los tiempos de inicio de las actividades relacionadas en un proyecto, para minimizar el tiempo total de ejecución (*makespan*), toda vez que, existen ciertas restricciones en las relaciones de precedencia y de capacidad de recursos finitos (Ballestín, Schwindt, & Zimmermann, 2007). El RCPSP puede definirse según Mingozzi et al. (1998) de la siguiente manera:

Sea un proyecto constituido por un conjunto $X = (1, .., n)$ de actividades y un conjunto de $S = (1, .., m)$ de recursos, donde cada recurso $k \in S$ tiene una disponibilidad total b_k en cada intervalo de tiempo. Cada actividad $i \in X$ tiene un tiempo de procesamiento constante d_i, cuya ejecución requiere de una cantidad constante r_{ik} del recurso k. Las interrupciones de las actividades no son permitidas y los tiempos de preparación se asumen dentro de los tiempos de procesamiento e independientes de las actividades. Las actividades ficticias 1 y n representan el inicio y la finalización del proyecto, con duración y consumo de recursos iguales a cero (Mingozzi, Maniezzo, Ricciardelli, & Bianco, 1998).

Existen varios métodos de solución para el RCPSP y particularmente se destacan las técnicas metaheurísticas, que se conocen por ser algoritmos de optimización muy generales basados en procesos iterativos para diseñar o mejorar heurísticas. Como los algoritmos genéticos (AG) que permiten obtener una solución en un esfuerzo computacional bajo.

En esta investigación se pretende aplicar un AG al RCPSP con duración aleatoria de las actividades, donde se esperan mejores soluciones. La presente investigación está organizada por capítulos de la siguiente forma.

En el capítulo 1 se encuentra la descripción del Problema de Programación de Proyectos con Recursos Restringidos (*RCPSP*), en el capítulo siguiente, capítulo 2 se exponen el marco teórico, el capítulo 3 se compone de las generalidades del proyecto que son, planteamiento del

problema, justificación del proyecto, objetivos, luego en el capítulo 4 se encuentra la revisión de literatura del RCPSP enfocado en algoritmos genéticos como técnicas de solución, para el capítulo 5 se muestra el Algoritmo Genético propuesto (fase proactiva), y posterior a este, en el capítulo 6 está la simulación (fase reactiva) y por último en los capítulos 7, 8 y 9 se encuentran los resultados, conclusiones y recomendaciones respectivamente.

Tabla 1.

Cumplimiento de objetivos del proyecto

Objetivo	**Cumplimiento**
1. Realizar una revisión de literatura sobre el Problema de programación de proyectos con Recursos restringidos (RCPSP) con duración de actividades aleatorias.	Capítulo 4
2. Desarrollar un algoritmo genético para el Problema de programación de proyectos con Recursos restringidos (RCPSP) que permita obtener una línea base para el proyecto.	Capítulo 5
3. Evaluar la **robustez** de la(s) línea base generada por el algoritmo genético propuesto con la **robustez** de una línea base obtenida a partir de programación lineal entera.	Capítulo 6
4. Analizar los resultados obtenidos, como mínimo en dos redes de proyecto diferentes.	Capítulo 7, 8 y 9
5. Elaboración de un artículo de carácter publicable que contenga los resultados del proyecto de investigación	Apéndice A

1. Generalidades del Proyecto

1.1 Planteamiento del problema

Los problemas de secuenciación de actividades se han diseñado para lograr una asignación óptima, tanto en términos de tiempo como sobre los recursos limitados. Específicamente cuando se habla de secuenciación, se trata un caso concreto de la programación, a saber, aquella que controla el orden de una serie de actividades que guardan alguna conexión; la programación, también puede darse para asuntos generales, asignando tiempos para la realización de las acciones y/o actividades y para asignar el uso de otros recursos limitados.

Por su importancia en la gestión de proyectos de diferentes tipos como proyectos de construcción, programación de producción industrial, prestación de servicios entre otras, la programación de proyectos con recursos restringidos (*Resource Constrained Project Scheduling Problem: RCPSP*) es uno de los problemas más estudiados por su gran aplicabilidad en la programación de tareas como a nivel industrial y empresarial, dando cabida a su uso en proyectos diversos (Morillo, Moreno, & Díaz, 2014).

Los investigadores utilizan para resolver el problema Procedimientos Especiales (*Special Procedures: SP*) los cuales generan soluciones de calidad con esfuerzos computacionales bajos, también usan métodos exactos especialmente la técnica *Branch and Bound (B&B)* y la Programación dinámica; sin embargo, los avances en el tema van siempre direccionados en la búsqueda de soluciones mejores en términos de robustez.

Resolver este tipo de problemas por métodos exactos puede ser complicado porque al usar un gran número de actividades requiere esfuerzos computacionales grandes, entonces los

procedimientos diseñados específicamente por los investigadores y las metaheurísticas son las técnicas de solución más aplicadas.

Al no poder resolver el problema de una manera exacta debido al esfuerzo computacional, al ser no determinista, se requiere buscar métodos alternativos y encontrar cada vez mejores soluciones para resolverlo. La problemática se basa en la dificultad para encontrar soluciones altamente robustas y que al mismo tiempo provean un *makespan* mínimo esperado para el RCPSP no determinístico. Utilizar metaheurísticas como los algoritmos genéticos (AG) permite obtener muy buenas soluciones en un tiempo computacional bajo y se podría decir que es una buena solución, es decir, una solución más robusta (Morillo, Moreno, & Díaz, 2014).

El objeto de investigación sobre el cual se trabaja es el RCPSP no determinista, donde se busca aplicar nuevas técnicas de solución que podrían dar mejores resultados para problemas mediante el uso de procedimientos proactivos y reactivos.

1.2 Justificación del Proyecto

El interés sobre los problemas de secuenciación de actividades se debe a su gran aplicabilidad en la programación de tareas tanto a nivel industrial como empresarial, dando cabida a su uso en proyectos de diferente tipo como proyectos de construcción, programación de producción industrial, prestación de servicios entre otras. El objetivo de los problemas de secuenciación es la asignación óptima de las actividades teniendo en cuenta los recursos. Cuando se habla de secuenciación es dar un orden a una serie de actividades que se relacionan entre sí, en otras palabras, hace referencia a programación que se puede desarrollar en casos donde se tengan en cuenta más de un recurso.

Al abordar este tipo de problemas de secuenciación de actividades, se encuentra uno de los más estudiados el problema de programación de proyectos con recursos restringidos (*Resource Constrained Project Scheduling Problem: 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 planeado por las organizaciones, debido a esto, surge la necesidad de seguir investigando.

Por lo tanto, en el proceso de revisión de literatura, los investigadores han hecho diversas contribuciones para resolver este problema y han realizado investigaciones en este campo, donde inicialmente se aplicaron métodos exactos para su solución; sin embargo, a medida que aumenta el número de actividades de la red del proyecto, los recursos y su disponibilidad, se crea la necesidad de hacer uso de métodos de aproximación como lo son las distintas metaheurísticas con la finalidad de encontrar soluciones a situaciones más complejas.

En suma, la presente investigación tiene como objetivo el desarrollo de un algoritmo genético que permita dar solución al RCPSP analizándolo dentro de un contexto probabilístico, ya que representa la duración de las actividades por medio de distribuciones de probabilidad, donde por programación reactiva el AG genera una línea-base y una lista de actividad asociada a esta. Esa lista de actividades definitiva (el resultado del AG) se compara con la línea-base de Otríz & Díaz (2020) la cual tiene su lista de actividades asociada (programación proactiva), se evalúa la eficiencia del resultado dependiendo del valor esperado del *makespan*.

1.3 Objetivos

1.3.1 Objetivo General

Desarrollar un algoritmo genético con el fin de resolver el Problema de programación de proyectos con Recursos restringidos (RCPSP) con una duración de actividades aleatorias, tomando como parámetros de entrada las precedencias entre actividades, los recursos y la duración obtenida a partir de un método basado en duraciones redundantes, logrando una línea base robusta para el proyecto.

1.3.2 Objetivos Específicos

1. Realizar una revisión de literatura sobre el Problema de programación de proyectos con Recursos restringidos (RCPSP) con duración de actividades aleatorias.
2. Desarrollar un algoritmo genético para el Problema de programación de proyectos con Recursos restringidos (RCPSP) que permita obtener una línea base para el proyecto.
3. Evaluar la robustez de la(s) línea base generada por el algoritmo genético propuesto con la robustez de una línea base obtenida a partir de programación lineal entera.
4. Analizar los resultados obtenidos, como mínimo en dos redes de proyecto diferentes.
5. Elaboración de un artículo de carácter publicable que contenga los resultados del proyecto de investigación.

2. Marco Teórico

2.1 Revisión de la literatura

A la fecha, se ha hecho múltiples extensiones del problema RCPSP. En el presente proyecto de los autores Palpant, Artigues, & Michelon (2004) se considerará el problema base como un programa lineal entero mixto el cual se modela a continuación:

Subíndices:

i: Son las actividades

j: Son las actividades

t: Periodos de tiempo

k: Recursos

Parámetros de entrada:

di: Duración estimada de la actividad i (incluyendo las nuevas actividades que se insertan a la red del proyecto)

p_{ij}: Parámetro binario que indica la relación de precedencia entre actividades. (p_{ij} es igual a 1 si la actividad i precede la actividad j, e igual a cero en caso contrario.

El horizonte de tiempo es dividido en periodos parciales representados por el subíndice (t), el cual toma valores entre 1 y T. T es el límite superior para la duración total del proyecto.

Variables de decisión:

Si: Tiempo de inicio planeado de la actividad i

Ait: Variable binaria que indica si la actividad i se ejecuta en el periodo t. (A_{it} es igual a 1 si la actividad i se encuentra activa en el periodo t, e igual a cero en caso contrario).

El modelo de optimización para resolver el RCPSP con duración de actividades aleatoria e inserción de tareas, es expresado como un programa lineal entero:

$FO: Minimizar\ Sn$ (3.1)

Sujeto a:

$$S_j \geq P_{ij} * [S_i + d_i] \qquad \forall i, \forall j \qquad (3.2)$$

$$S_i + d_i - 1 \geq t * A_{it} \qquad \forall i, \forall t \qquad (3.3)$$

$$S_i \leq t + (1 - A_{it}) * M \qquad \forall i, \forall t \qquad (3.4)$$

$$\sum_{t=1}^{T} A_{it} = d_i \qquad \forall i \qquad (3.5)$$

$$\sum_{i=1}^{n} (r_{ik} * A_{it}) \leq RW_{kt} \qquad \forall k, \forall t \qquad (3.6)$$

$$S_i \geq 0 \qquad \forall\ i \qquad (3.7)$$

$$A_{it} \in \{0, 1\} \qquad \forall\ i, \forall t \qquad (3.8)$$

$$P_{ij} \in \{0, 1\} \qquad \forall\ i, \forall j \qquad (3.9)$$

La función objetivo (3.1) del modelo busca minimizar el Makespan o duración del proyecto, donde Sn señala el tiempo de inicio de la actividad ficticia n. La ecuación (3.2) garantiza que las actividades inician solo cuando aquellas que la preceden han finalizado. Las ecuaciones (3.3 – 3.5) aseguran que Ait sea igual a 1 cuando la actividad i se encuentra activa en el momento t, de lo contrario, Ait será igual a cero. A continuación se expone el aporte de cada ecuación: la ecuación (3.3) permite que Ait sea 1 si y solo si la actividad i no ha finalizado en el momento t, la ecuación (3.4) permite que Ait sea 1 solo si la actividad i inició después del momento i, donde M representa un número muy grande, y la ecuación (3.5) asegura que Ait sea 1 para el intervalo $\{S_i, S_i + d_i\}$.

De otra parte, la ecuación (3.6) permite programar de manera simultánea varias actividades siempre y cuando no sobrepase los recursos disponibles. Finalmente, las ecuaciones (3.7 – 3.8 y 3.9) indican las condiciones de no negatividad y el tipo de variable requerido.

En la literatura se pueden encontrar diversos estudios acerca *RCPSP*, encaminados a la reducción del *makespan* ya que desempeña un gran papel 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, & Díaz, 2014, pág. 209).

Los problemas de planificación de proyectos con recursos limitados no se resuelven fácilmente. Debido a la complejidad computacional, las heurísticas de programación utilizadas se emplean para grandes problemas, por lo general se combinan para tratar de lograr un mejor resultado en tiempos computacionales.

Para el problema de programación de proyectos con recursos restringidos (RCPSP), de clase *NP-hard*, llamado así por su difícil solución al momento de optimizarlos debido a la complejidad de su naturaleza combinatoria, resulta de gran utilidad el uso de metaheurísticas para resolver problemas con numerosas actividades, además de la incertidumbre en el proceso de estimación de la duración de la actividad. Las metaheurísticas por lo general se combinan para intentar obtener un mejor rendimiento, buscando así todas las combinaciones posibles.

Liu, Yung & Ip (2007) propusieron un algoritmo específico de búsqueda genética local (GLS) basado en una teoría de conjuntos difusos, esta teoría provee herramientas de modelado para tratar las duraciones de actividades aleatorias.

Más adelante Zhao, You & Lv (2008) desarrollaron una programación de cadena crítica, donde intentaron resolver el RCPSP por medio de un algoritmo genético mejorado, para lograr

determinar el tamaño de un búfer en el proyecto. Por otra parte, Tahooneh & Ziarati (2011) usan otro método para resolver el RCPSP estocástico, usando una metaheurísticas llamada colonia de abejas (ABC) basada en el comportamiento inteligente de los enjambres de abejas domésticas.

Con la suposición de que el cronograma de referencia de un proyecto se realizara según lo planeado, Shou & Wang (2012) propusieron un algoritmo genético el cual genera secuencias estables que no sean susceptibles a las actividades con duración estocástica para generar una mejor respuesta del RCPSP. Por otra parte, Zhong, Yu & Jia (2014) desarrollaron un algoritmo genético para resolver el RCPSP con duraciones estocásticas, utilizando un método de dos puntos para simular una propiedad incierta, donde compararon el resultado de esta metaheurística con la simulación de Monte Carlo, obteniendo como resultado que el método propuesto es más efectivo y eficiente.

Más adelante, Mogaadi & Chaar (2016) investigaron dos modelos robustos, el modelo min-max que se enfoca en la minimización del objetivo de robustez absoluta y el modelo de arrepentimiento min-max que tiene por objeto minimizar el arrepentimiento absoluto, proponiendo así un algoritmo genético para encontrar una mejor solución. En este mismo año Huang, Dai & Du (2016) tomaron en cuenta que la cantidad de incertidumbres y dinámicas afectan la programación de proyectos y además a esto aumentan la complejidad de este, por ende, proponen un algoritmo genético de Pareto.

Considerando el RCPSP incierto, Ma et al. (2016) propusieron un modelo de programación de proyectos basado en la teoría de la incertidumbre, para dar solución se utilizó un algoritmo genético que integra una simulación incierta basada en 99 métodos.

Finalmente Ortiz & Diaz (2020) realizan un análisis de literatura para el problema de programación del proyecto con la duración de las actividades no deterministas de los artículos

publicados desde el año 1996 hasta el 2018, donde identifican que los Algoritmos Genéticos (AG) son la técnica de solución principal empleada por los investigadores, y el Problema de Programación de Proyectos con Recursos Restringidos (RCPSP) es el tipo de problema más estudiado. Se destacan Roel Leus y Hua Ke debido al número de publicaciones realizadas en el tema. Es por esto por lo que se invita al lector a ahondar más en la lectura del análisis preliminar del artículo.

En este punto se hace necesario complementar la revisión de literatura y realizar su respectivo análisis en la base de datos de Web of Science, haciendo una inspección de los artículos que no se encontraron en la base de datos de Scopus, además de exponer otras versiones del RCPSP durante el periodo de tiempo que permite esta base de datos, desde 2001 hasta 2020, con la ecuación de búsqueda del presente proyecto:

("Resource constrained project scheduling" OR "Resource-Constrained Project Scheduling" OR RCPSP) AND (uncertain OR uncertainty OR stochastic OR random) AND duration.

Inicialmente para el año 2012 los autores Baradaran et al. (2012) presentaron un artículo con un algoritmo metaheurístico híbrido (HMA) para el problema de programación de proyectos con restricciones de recursos multimodo (MRCPSP) en redes PERT. Con el objetivo de aprobar el rendimiento del algoritmo metaheurístico híbrido, las soluciones se comparan con soluciones óptimas para redes pequeñas.

Entre otra de estas variaciones, se encuentra el estudio del RCPSP con incidencia en la duración de las actividades, Xiong et al. (2012) consideraron la robustez y la estabilidad del cronograma del proyecto, modelando el problema con un problema de optimización multiobjetivo.

Proponiendo un Algoritmo evolutivo multiobjetivo híbrido (H-MOEA), los resultados obtenidos exponen que el enfoque propuesto es factible y efectivo.

Según Masmoudi & Hait (2013) en el mundo real, los proyectos están expuestos a incertidumbres en los diferentes niveles de planificación de un proyecto, es por esto que proponen una técnica en la que permanece la incertidumbre en todo el procedimiento del modelo, considerándolo un modelo difuso inspirado en el enfoque difuso/probabilista.

En la planificación de proyectos una parte importante que debe tenerse en cuenta es la disminución del tiempo de una actividad al agregar más recursos, como pueden ser trabajadores y horas extra. Donde Glisovic (2014) propone se compara el recocido simulado difuso y el algoritmo genético basado en el método de falla, para evaluar las redes del proyecto y determinar la forma de minimización del costo promedio del proyecto, que son causados por retrasos y otros. Como resultados de la evaluación el método se puede aplicar de manera confiable a proyectos de ingeniería.

Wein et al. (2014), abordaron el *multi-project scheduling problem* BMPSP, considerando factores de incertidumbre y de diferentes objetivos para lograr la finalización de todos los proyectos a tiempo, proponiendo un algoritmo genético diseñado para generar la prioridad de las actividades y obtener un plan de programación, teniendo como resultado final que el algoritmo genético supera significativamente a algoritmos multiobjetivos anteriores. Años más adelante Chan et al. (2019) desarrollaron para el RCPSP dos algoritmos genéticos NSGA-II considerando las duraciones de las actividades y NSGA-III también teniendo en cuenta las duraciones de las actividades como incertidumbres y además introduciendo amortiguadores dentro de estas, como resultados obtuvieron que el NSGA-III obtuvo mejores soluciones.

En el año 2020 Khalilzadeh, Hosseini, & Ghaeli (2020) abordaron el RCPSP Multi-Mode (MRCPSP), desarrollaron un algoritmo hibrido basado en un algoritmo genético, el cual compararon con un modelo exacto, obtenido para el algoritmo propuesto un mejor resultado en un tiempo computacional más corto. Además, Kucuksayacigil & Ulusoy (2020) estudiaron el RCPSP considerando a multimodo, multi-objetivo y multi-proyecto, donde desarrollaron un algoritmo genético híbrido (híbrido-NSGA-II) como método de solución y un procedimiento de inyección en NSGA-II. Realizaron un extenso estudio computacional obteniendo como resultados que la NSGA-II híbrida superó a la NSGA-II en términos de rendimiento, dispersión máxima.

2.2 Problema de Programación de Proyectos con Recursos Restringidos

El problema de programación de proyectos con recursos restringidos (*RCPSP*), es uno de los problemas de programación de optimización combinatoria más desafiantes, ya que ha sido el foco de un gran número de investigaciones, tal y como lo indican Lacomme et al. (2019), lo que resulta en numerosas publicaciones en la última década. Publicaciones centradas en *RCPSP*, incluyendo varias ampliaciones con diferentes objetivos que deben minimizarse y las limitaciones que deben comprobarse.

El RCPSP no solo tiene en cuenta las restricciones de precedencia entre las actividades, sino también el número de recursos que gasta cada una de ellas durante el proyecto. De acuerdo con Muñoz et al. (2013) existen recursos limitados que deben ser tenidos en cuenta a la hora de generar la planeación de los proyectos para que estos sean viables desde un punto de vista operacional. Es por esto que Kolish and Padman (2001) han clasificado estos recursos de la siguiente manera:

Recursos Renovables. Donde se definen como aquellos recursos que se encuentran disponibles durante la totalidad del horizonte de planeación, y cuya cantidad disponible en un periodo de tiempo dado no depende de la cantidad disponible en periodos anteriores. Para este punto se habla de horas-hombre, tiempo disponible de maquinaria o equipos, entre otros, los cuales están disponibles cada vez que inicia un nuevo periodo de tiempo (Kolish & Padman, 2001).

Recursos no Renovables. Se entienden como aquellos cuyo uso se produce una única vez y su cantidad en un periodo es estrictamente dependiente de la cantidad utilizada en el periodo anterior; como lo son los materiales requeridos o dinero, ya que están aptos para el proyecto en forma global y pueden consumirse en cualquier momento siempre y cuando no sobrepase el límite máximo (Kolish & Padman, 2001).

Recurso Doblemente Restringido. Cumplen simultáneamente las dos consideraciones, tanto el recurso renovable como el recurso no renovable. A modo de ejemplo, el dinero cumple bien esta condición si además de la disponibilidad presupuestal total se le asigna un presupuesto máximo por periodo de tiempo (Kolish & Padman, 2001).

Recurso Parcialmente Renovable. Se define como la disponibilidad que tiene un recurso para un subconjunto de periodos, se entiende que por medio de este tipo de recursos se puede llegar a representar a las otras tres categorías (Kolish & Padman, 2001).

En consecuencia, se aborda información más detallada sobre el RCPSP, donde se clasifica en un enfoque determinístico y no determinístico.

2.2.1 RCPSP Determinístico

El Problema de Programación de Proyectos con Recursos Limitados RCPSP según Viveros & Rivera (2017) es un problema proveniente del área de programación de producción y de

construcción, aunque sus aplicaciones se extienden a diversas áreas del conocimiento. Donde los autores Moreno et al. (2007) exponen como objetivo del RCPSP asignar tiempos de inicio a cada una de las actividades de manera que se satisfagan todas las restricciones, de precedencias y recursos del sistema, y se minimice la duración total del proyecto o *makespan*.

En la formulación del modelo de optimización son definidas n actividades o tareas, las cuales son identificadas por un subíndice $(i\ o\ j)$ con valores entre 1 y n. Donde la primera y la última actividad se representan como actividades ficticias con duración cero por temas de practicidad en la programación. La planeación se divide en períodos parciales representados por el subíndice (t) con valores entre 1 y T. T es el límite superior para la duración total del proyecto, En cada periodo de tiempo, cada actividad i gasta r_{ik} unidades del recurso k. La cantidad máxima del recurso k disponible para cada periodo de tiempo t se expresa como RW_{kt}.

FO: $Minimizar\ Sn$ (1.1)

Sujeto a:

$$S_j \geq S_i + d_i \qquad \forall (i,j)\ \in E \qquad (1.2)$$

$$\sum_{i \in A(t)} r_{ik} \leq RW_{kt} \qquad \forall k, \forall t \qquad (1.3)$$

$$S_i\ \geq 0 \qquad \forall\ i \qquad (1.4)$$

Donde S_i representa el periodo de inicio de cada actividad i, y d_i su duración. Las restricciones indican lo siguiente:

La restricción (1.2) da los tiempos de inicio para cada actividad j teniendo en cuenta los tiempos de inicio y la duración de las actividades precedentes. Por otra parte, E representa el conjunto de actividades y sus precedencias.

La restricción (1.3) hace referencia a la restricción de recursos renovables, ya que no permite que las actividades activas en un periodo t, superen la cantidad disponible de recursos tipo

k, a su vez, A(t) hace referencia a diferentes conjuntos de actividades que comparten recursos y pueden ser programadas simultaneamente en cada periodo t.

La restricción (1.4) indica la no negatividad ya que los tiempos de inicio son variables continuas.

Por otro lado, es importante nombrar algunas variaciones del problema de programación de proyectos con recursos restringidos (RCPSP). Las más conocidas son:

El problema multi-objetivo de programación de proyectos con recursos restringidos, donde así mismo las restricciones de recursos y de precedencias, la generación de la línea-base tiene dos o más objetivos los cuales deben optimizarse simultáneamente.

El problema multi-modo de programación de proyectos con recursos restringidos, da la posibilidad de hacer de forma diferente cada una de las actividades del proyecto, el problema de optimización genera una línea-base e indica la forma más conveniente para ejecutar cada actividad.

Cuando se genera una línea-base que tenga en cuenta simultáneamente los recursos que deben ser compartidos por todas las actividades de los proyectos que se están analizando se le llama el problema de programación con recursos restringidos y múltiples proyectos (*Resource Constrained Multi-Project Scheduling Problem:* RCMPSP).

El problema de programación de proyectos con recursos restringidos e inserción de tareas (*Resource Constrained Project Scheduling Problem- task insertion*: RCPSP-TI), estudia la posibilidad de incorporar nuevas actividades o tareas a un proyecto en curso. Esta situación aparece cuando existen imprevistos o eventualidades que llevan al gerente de proyectos a modificar el cronograma inicial.

2.2.2 RCPSP no determinístico

Los modelos construidos para resolver el RCPSP generalmente se asocian a un enfoque determinístico, en donde los parámetros como por ejemplo los recursos a utilizar, los costos involucrados o la duración de las actividades son tomados como valores fijos, no obstante, cada vez es más usual la construcción de modelos considerando comportamientos aleatorios en los parámetros de entrada del modelo, es decir, considerando un enfoque no determinista.

Es más apropiado analizar los proyectos bajo un contexto probabilístico o de incertidumbre, debido a que las actividades suelen desarrollarse en un entorno dinámico, cambiante, y sujeto a riesgos. Suponiendo que un gerente de proyecto dispone de información histórica, cuenta con la asesoría de personal de amplia experiencia, lo cual le permite identificar el tipo de distribución de probabilidad más adecuado para representar cada parámetro del modelo matemático; sin embargo, en caso de no contar con dicha información es necesario trabajar bajo un contexto de incertidumbre.

El desarrollo del algoritmo genético para solucionar el RCPSP presentado en este proyecto de grado ha sido analizado dentro de un contexto probabilístico, ya que representa la duración de las actividades por medio de distribuciones de probabilidad.

En el enfoque no determinístico se analiza el RCPSP basado en las estrategias de programación predictiva, proactiva y reactiva.

2.2.2.1 Programación predictiva, proactiva y reactiva. Los procedimientos para abordar el RCPSP no determinístico se diseñan con base en estrategias predictivas, reactivas o proactivas.

Una estrategia predictiva no tiene en cuenta el riesgo que afecta la duración de las actividades, y establece una línea-base tomando como referente parámetros determinísticos como

el valor esperado de la duración de las actividades. En todo caso, el problema es resuelto como un problema determinístico (Brčić, Kalpic, & Fertalj, 2012).

Un método de solución proactivo tiene en cuenta el riesgo dentro del modelo matemático para generar una línea-base robusta para un proyecto. Una línea-base puede considerarse robusta cuando los tiempos de inicio resultantes para cada actividad requieren pocos ajustes al aparecer las disrupciones que afectan su duración.

Dentro del procedimiento predictivo existen tres tipos de solución: los métodos basados en duraciones redundantes *(redundancy based methods*), los métodos de programación robusta (*robust scheduling methods*), y los métodos de programación contingente (*contingent scheduling*) (Brčić, Kalpic, & Fertalj, 2012).

Los métodos basados en duraciones redundantes (*redundancy based methods*), buscan proveer tiempo adicional a las actividades, de forma que para enfrentar las eventualidades que puedan aparecer sirvan de soporte, evitando de esta manera el incumplimiento en la fecha de finalización del proyecto. El tiempo extra o protección adicional se incorpora de dos formas, una es extendiendo la duración original de cada actividad y la otra insertando amortiguadores de tiempo (buffers) en sitios estratégicos de la red del proyecto.

Los métodos de programación robusta (*robust scheduling methods*) presentan una línea-base para el proyecto que ha sido previamente evaluada por medio de una función objetivo que busca optimizar alguna medida de robustez.

los métodos de programación contingente (*contingent scheduling*) generan más de una línea-base para el proyecto, las cuales han sido creadas fundamentada en las diferentes alternativas de riesgo, donde los cambios son analizados previamente y se crea una línea-base para cada

posibilidad, de tal manera que se contará con planes de acción alternativos durante la ejecución del proyecto.

En la literatura el enfoque proactivo y reactivo, han despertado un gran interés entre los investigadores. Verfaillie & Jussien (2005) lo definen de la siguiente manera:

Un método de solución proactivo tiene en cuenta el riesgo dentro del modelo matemático para generar una línea-base robusta para un proyecto. Una línea-base puede considerarse robusta cuando los tiempos de inicio resultantes para cada actividad requieren pocos ajustes al aparecer las disrupciones que afectan su duración. Por otra parte, la programación reactiva es aquella que permite secuenciar las actividades justo en el momento en que se ejecuta el proyecto, esto indica que la generación de una línea-base no es obligatoria para llevarlo a cabo. Existen dos tipos de programación reactiva la programación on-line y la programación reparadora de la línea-base. La primera programación reactiva on-line se refiriere a algoritmos que no tienen en cuenta una línea-base generada con anterioridad, ordenando las actividades del proyecto a partir de un proceso de toma de decisiones por etapas soportado en políticas que permiten seleccionar las actividades a programar en cada una de las etapas del horizonte de planeación. La segunda programación se refiere algoritmos que guían el proceso de ajuste de la línea-base generada por algún procedimiento predictivo o proactivo; dependiendo del origen de la línea-base en consecuencia estos procedimientos se conocen como programaciones predictivas-reactivas o programaciones proactivas-reactivas.

2.2.2.2 Complejidad del RCPSP. El RCPSP es uno de los problemas más comunes en literatura científica relacionada con Scheduling y se considera como un problema *NP-hard*

(Hartmann, 1999; Klein, 2000; Artigues, Demassey, & Néron, 2008), lo que implica que un gran esfuerzo computacional es necesario para dar solución a este tipo de problemas.

La complejidad del RCPSP se eleva a medida que aumenta el número de actividades de la red del proyecto, no obstante, este no es el único factor que interviene. Algunos de estos factores adicionales son los recursos analizados, disponibilidad de los recursos y la complejidad de la red del proyecto.

Al incrementarse el número de los recursos analizados, se eleva la posibilidad de generar conflicto entre actividades que se programan inicialmente de forma simultánea. En la disponibilidad de los recursos donde un recurso renovable por ser escaso conlleva a limitar el número de actividades programadas en el mismo periodo de tiempo. Al igual que en los recursos analizados, el espacio de soluciones factibles se reduce.

No solo la cantidad de relaciones de precedencia que exista entre las actividades, sino también la ubicación de dichas relaciones influye en el tiempo requerido para encontrar una solución al problema y esto es la complejidad de la red del proyecto. Los indicadores de complejidad más conocidos relacionados a los factores descritos anteriormente son la Complejidad de la red (NC), el índice de complejidad (CI), el factor de Recursos (RF) y la Intensidad de los recursos.

Complejidad de la red (NC). Este indicador compara las redes de proyecto considerando la relación entre el número de actividades y el número de precedencias. Los autores Browning & Yassine (2010) presentan la formulación para calcular este indicador.

Índice de complejidad (CI). Es el número mínimo de métodos de reducción usados para simplificar una red de proyecto cíclica. Los métodos de reducción pueden ser de nodo, reducción en serie y reducción en paralelo (De Reyck & Herroelen, 1996).

Factor de Recursos (RF). Es el nivel de dependencia entre actividades y tipo de recursos; Los autores Kolisch, Sprecher, & Drexl (1995) presentan la ecuación para calcular este valor.

Intensidad de los recursos. Evalúa cada tipo de recurso estableciendo la relación entre el promedio la cantidad requerida por actividad y el total de recurso disponible (Kolisch, Sprecher, & Drexl, 1995)

La complejidad del RCPSP no determinístico está sujeta por los mismos factores asociados al RCPSP determinístico. No obstante, ya que los parámetros del problema pueden asumir cualquier valor dentro de un rango previamente definido, el proceso de solución debe considerar el análisis de múltiples escenarios, donde cada uno de ellos tiene su propia solución determinística. Esto implica que a medida que se aumenta el número de escenarios analizados es más complejo encontrar una solución adecuada al problema.

Otros factores, como la localización de las actividades y la correlación entre los riesgos que afectan un proyecto, pueden aumentar la complejidad del RCPSP no determinístico, como por ejemplo:

Localización de las actividades. Cuando las actividades con alto nivel de incertidumbre hacen parte de la ruta crítica del proyecto la búsqueda de una solución adecuada al problema se dificulta.

Correlación entre riesgos que afectan las actividades. Existe una relación directa entre el número de riesgos correlacionados entre sí y la complejidad del problema ya que la aparición de un riesgo puede afectar la duración de una o más actividades, y además a esto cuando un riesgo se encuentra correlacionado con otra abra un impacto adicional en la duración de un nuevo grupo de actividades.

Hasta la fecha no se encuentran reportado en la literatura indicadores de complejidad exclusiva para el RCPSP no determinístico.

2.3 Schedule Generation Schemes, SGS

Los esquemas de generación de secuencias (SGS) son la base de la mayoría de los programas de soluciones heurísticas para el RCPSP (Mika, Waligóra, & Węglarz, 2005). El SGS comienza desde cero y crea un cronograma viable que se lleva a cabo 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 *makespan*. Sin embargo, existen tipos de esquemas de generación lo cuales son el Esquema de generación de programas en paralelo *(Parallel Schedule Generation Scheme)* el cual está dirigido al tiempo, cada iteración posee un único momento t, en donde hay muchas actividades de precedencia y cierta cantidad de recursos en cada algoritmo, este esquema es factible siempre que haya suficientes recursos para que se puedan ejecutar actividades al mismo tiempo y sin sobrepasar el tiempo total. También se pueden distinguir con el incremento de actividad y tiempo el esquema generador de secuencias en serie (*Serial Schedule Generation Scheme*) donde se llevan a cabo n iteraciones y es posible programar una actividad en cada una, es decir, de forma lineal, por último, el SGS en serie estocástico, sigue trabajando como el SGS en serie, es decir, que toma las actividades una por una en el orden de la lista y las programa tan pronto sea posible, pero sin adelantar las actividades que están ya programadas.

2.3.1 Reglas de prioridad (Priority Rules)

Estas reglas consisten en la generacion de listas de actividades organizadas de acuerdo a su prioridad, pero sujetas a las restricciones de precedencia, estableciendo la actividad que se va a seleccionar durante el proceso del esquema generador de secuencias. Demeulemeester y Herroelen (2002) clasificaron las reglas en categorias ver Tabla 2.

Tabla 2.

Clasificación de las Reglas de Prioridad

Clasificación	Descripción	Reglas de prioridad
Basada en la actividad	Relaciona información con la actividad y no con el resto del proyecto	SPT: Shortest Procesing Time (Tiempo de procesamiento más corto) LPT:Longest Processing Time (tiempo de procesamiento más largo) RND: Random (Aleatorio)
Basada en la red	Relaciona la información de precedencia de las actividades	MIS: Most Total Successors(Sucesores más inmediatos). MTS:Most Total Successors (Mayoria de los sucesosres totales) LNRJ: Least Non-related Jobs (Trabajos menos relacionados) GRPW: Gretest Rank Positional Weight (Rango de mayor peso posicional)

Clasificación	Descripción	Reglas de prioridad
Basada en ruta critica	Se basa en los cálculos de la ruta critica	EST: Earliest Start Time (tiempo de inicio más temprano). EFT:Earliest Finish Time (Tiempo de inicio Tardío) LST: Latest Stard Time(Ultima hora de inicio). LFT: Latest Finish Time (Tiempo de finalizacion mas tardio). MSLK: Minimum Slack (Holgura mínima) RSM: Resource Scheduling Method (Metodo de planificacion de recursos) ESTD: Dynamic Earliest Start Time (Tiempo de inicio más temprano dinámico) EFTD: Dynamic Earliest Finish Time (Tiempo de finalización más temprano dinamico). MSLKD: Dynamic Minimum Slack (Holgura minima dinamica)
Basada en el recurso	Se fundamenta en el uso del recurso de la actividad que se está realizando 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).

Clasificación	Descripción	Reglas de prioridad
Compuestas	Consiste en la combinacion de reglas anterioes	WRUP: Weighted Resource Utilisation and Precedence (Utilizacion de recursos ponderados y precedencia) IRSM: Improved Resource Scheduling Method(Metodo mejorado de planificacion de recursosos). WCS:Worst Case Slack (Holgura en el peor de los casos)

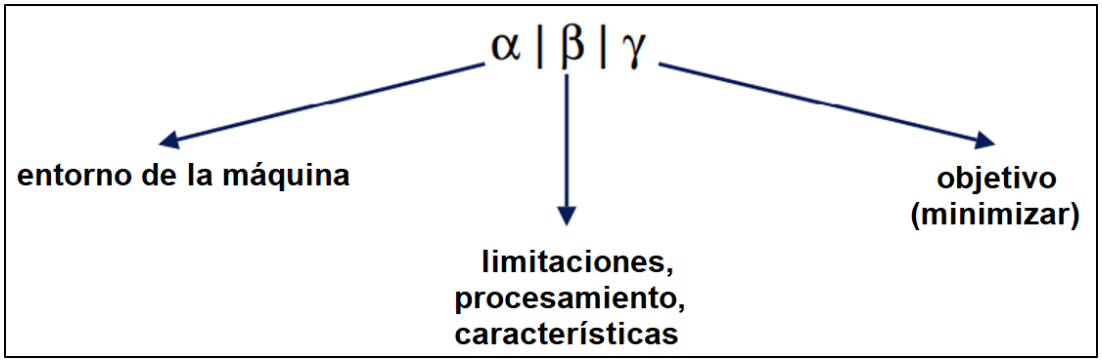
Nota: (Demeulemeester & Herroelen, 2002).

2.4 Problema de Planificación (*Scheduling problem*)

El *Scheduling problem*, es el problema de determinar el orden en que las operaciones se ejecutarán. En los problemas de programación, los trabajos deben procesarse en máquinas de capacidad limitada, para ello es necesario tener en cuenta la notación de los siguientes tres campos: $\alpha \mid \beta \mid \gamma$ ver figura 1. Donde, α es el entorno de la máquina, β representa las características del trabajo a procesar y γ es el objetivo (óptimo) para dar solución al problema de programación.

Figura 1.

Descripción de Scheduling Problem



Nota. (Schwiegelshohn, 2004)

Scheduling problem abarca tres áreas, la fabricación y producción, el transporte y distribución y por último la Información - procesamiento. En el área de fabricación y producción es aplicado a proyectos usando esquemas de generación de secuencias (*Schedule Generation Schemes, SGS*) para generar secuencias óptimas siendo una solución heurística para el RCPSP.

2.5 Indicadores de robustez

Para resolver el RCPSP con duración de actividades no determinística, hasta el momento se han creado diferentes indicadores de robustez que tienen como fin evaluar el rendimiento de los algoritmos que se han propuesto. Los dos indicadores más habituales se llaman la robustez de calidad *(quality robustness)* y la robustez de la solución *(solution robustness).*

2.5.1 Robustez de calidad (quality robustness)

Hace referencia a la capacidad que tiene la línea-base para soportar las transformaciones en torno a la duración total del proyecto (Van De Vonder, Demeulemeester, Herroelen, & Leus, 2005).

2.5.2 Robustez de la solución (solution robustness)

Analiza la capacidad de la línea-base para soportar las transformaciones que afectan los tiempos de inicio de cada una de las actividades (Van De Vonder, Demeulemeester, Herroelen, & Leus, 2005).

2.6 Métodos de solución

Para dar solución al RCPSP a lo largo de los años se han realizado investigaciones, aplicando diferentes métodos.

2.6.1 Métodos exactos

Estos métodos se basan en supuestos y características específicas como la continuidad, diferenciabilidad, espacio de búsqueda pequeño o linealidad, entre otros. Con relación a la clase de los métodos exactos se encuentran los algoritmos tales con particularidades como el empleo de técnicas analíticas o matemáticas, las cuales aseguran una solución óptima siempre y cuando ésta exista. No obstante, la solución generada no siempre es la más adecuada debido a sus desventajas en problemas aplicados, ya que posee alta complejidad computacional. Los autores Bianco & Caramia (2012) utilizaron la técnica de solución *Branch and Bound* para minimizar el *makespan* y Palacio & Larrea (2017) presentaron dos enfoques de Programación Lineal de Entera Mixta (*Mixed-Integer Linear Programming, MILP*) para maximizar la robustez de la secuencia obtenida.

2.6.2 Métodos Iterativos de Aproximación

Estos métodos se conocen como heurísticas primitivas, encontrándose categorías como aquellas que incorporan procesos iterativos, en la que para cada iteración es posible generar búsquedas que ayuden en la construcción o mejora de la solución actual. En algunos casos este razonamiento es simple. Sin embargo, cuando se trata de problemas complejos, los resultados que ofrece este método no son convincentes. Mientras que en otras ocasiones, cuando el razonamiento es complejo, se da origen a las metaheurísticas, demostrando una gran variabilidad en problemas

complejos como el RCPSP ya que no es posible ofrecer una solución óptima (aunque a veces llegan a obtenerla) (Morillo, Moreno, & Díaz, 2014).

2.6.3 Metaheurísticas

Las técnicas metaheurísticas son algoritmos de optimización muy generales basados en procesos iterativos en su mayoría por la observación de procesos naturales, para diseñar o mejorar heurísticas, capaces de proporcionar muy buenas soluciones en tiempo y con recursos razonables. Morillo, Moreno & 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).

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

2.6.3.1 Método Tabu Search (TS). Es una técnica de optimización combinatoria introducida por Fred Glover, la cual se origina de la inteligencia artificial usando conceptos de memoria adaptativa y su principal característica es el uso de memoria adaptativa que permite explorar diferentes regiones en el espacio de búsqueda (memoria de corto plazo) (Melián & Glover, 2003).

2.6.3.2 Metaheurísticas basadas en población. Las operaciones para estas técnicas son realizadas con cada una de las posibles soluciones de un conjunto de individuos los cuales se denominan población. Según Villagra et al. (2013) la eficacia de esta técnica se da

fundamentalmente por el tratamiento o manipulación de los individuos de la población en cada una de las iteraciones.

Los principales tipos de técnicas metaheurísticas basadas en población son los siguientes:

- Los Algoritmos Evolutivos
- Metaheurísticas híbridas: GRASP
-Los sistemas basados en Colonias de Hormigas, ("*Ant Colony Optimization*")
- Algoritmos Basados en Nubes de Partículas (*ó Particle Swarm Optimization*)
- La Búsqueda Dispersa (*SS ó "Scatter Searh*").

2.6.3.3 Optimización de la Colonia de hormigas (ACO). Es un algoritmo donde su estrategia de búsqueda está fundamentada mediante múltiples agentes que cooperan en la búsqueda, transmitiendo e intercambiando información al resto. El método nace de la observación de las hormigas, el cual intenta reproducir el mecanismo utilizado por las hormigas cuando requieren ubicar los alimentos informando al resto de individios de la población, el lugar donde se ubica la comida y la ruta, con la característica de tener menor distancia desde su nido hasta el alimento. Merkle & Middendorf (2002) aplicaron las colonias de hormigas al problema del RCPSP.

2.6.3.4 Algoritmo genético. Es un método basado en la población, siendo un procedimiento adaptativo que utiliza operadores de selección y recombinación para generar nuevas soluciones de problemas de búsqueda y optimización, imitando el proceso de selección natural y evolución de especies basado en el proceso genético de los organismos vivos (Rostami, Moradinezhad, & Soufipour, 2014).

Los Algoritmos Genéticos tienen la capacidad de ir creando soluciones para problemas que suceden en la cotidianidad, el resultado de estas soluciones se dirige hacia valores óptimos del problema dependiendo de una adecuada codificación.

2.7 Descripción del algoritmo genético (AG)

Según la definición bastante completa de un algoritmo genético propuesta por Koza (1994), un algoritmo genético es:

> "…un algoritmo matemático altamente paralelo que transforma un conjunto de objetos matemáticos individuales con respecto al tiempo usando operaciones modeladas de acuerdo al principio Darwiniano de reproducción y supervivencia del más apto, y tras haberse presentado de forma natural una serie de operaciones genéticas de entre las que destaca la recombinación sexual. Cada uno de estos objetos matemáticos suele ser una cadena de caracteres (letras o números) de longitud fija que se ajusta al modelo de las cadenas de cromosomas, y se les asocia con una cierta función matemática que refleja su aptitud"(p.89).

2.7.1 Nomenclatura

Genotipo: Puede ser vista como cromosomas donde está contenida toda la información siendo una cadena de caracteres, siendo la naturaleza propia de cada uno que lo distingue de los demás, siendo este quien determina el fenotipo del individuo.

Fenotipo: Es lo que identifica a cada individuo, se habla de sus características propias que lo distinguen a uno de otro, donde contiene la información, también puede ser vista como la solución que resulta de la descodificación de la cadena de cromosomas. El fenotipo es la

consecuencia de cómo se desarrolla el individuo debido al medio y la herencia de sus antecesores que recibe dicho individuo.

Cromosoma: Un cromosoma (a veces llamado genoma) es un conjunto de parámetros que definen una solución propuesta al problema que el algoritmo genético está tratando de resolver.

Individuo: Un conjunto de individuos forman una población, donde cada individuo es una cadena de caracteres que representan un conjunto de soluciones.

Población: Un conjunto de individuos (cromosomas) que viven en el mismo ambiente forman una población.

Padres: Tienen la labor (cromosomas) de crear nuevos individuos en la población.

Descendientes (hijos): Son los nuevos cromosomas, también son el resultado de la reproducción de los padres, que tienen como tarea transmitir los genes.

Selección: Son el conjunto de reglas usadas para elegir a los progenitores de la siguiente generación, que se reproducen por medio del cruzamiento genético y estos generan la descendencia.

Viabilidad: Es la probabilidad que tiene un cromosoma de sobrevivir.

Fertilidad: Es el número de descendientes los cuales son generados por los cromosomas durante su vida.

Generación: Es una estructura en un instante de tiempo dentro de la Población.

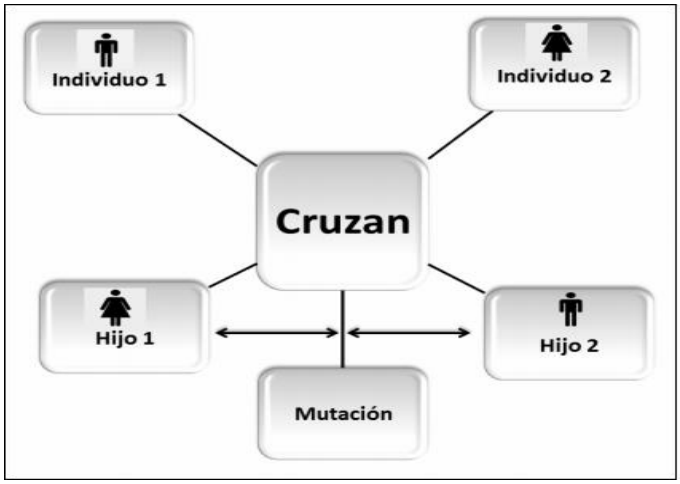
2.7.2 Esquema general de los algoritmos genéticos

Los autores Narváez & Saltos (2014) definen el siguiente esquema para los algoritmos genéticos.

> "Primero, existe una población inicial de individuos con determinadas características, luego los individuos se cruzan entre sí dando origen a los habitantes de la siguiente generación. Estos habitantes heredarán las características de sus padres y si éstas son beneficiosas se irán propagando a través de las generaciones futuras, caso contrario, desaparecerán por el proceso de selección natural que indica que sólo las especies más fuertes sobreviven. Por último, durante el proceso de cruce puede darse la mutación del nuevo individuo, si esta mutación es mala seguramente el nuevo hijo morirá al nacer, pero si es beneficiosa se irá transmitiendo en las generaciones futuras. La mutación es un proceso mediante el cual se altera de forma significativa el código genético del individuo otorgándole nuevas características que nadie posee en la población, pero al mismo tiempo ocurre con muy poca frecuencia. Posiblemente en una especie determinada uno de cada millón de habitantes ha mutado, el proceso se representa en la figura 2 y 3" (Narváez & Saltos, 2014, pág. 30).

Figura 2.

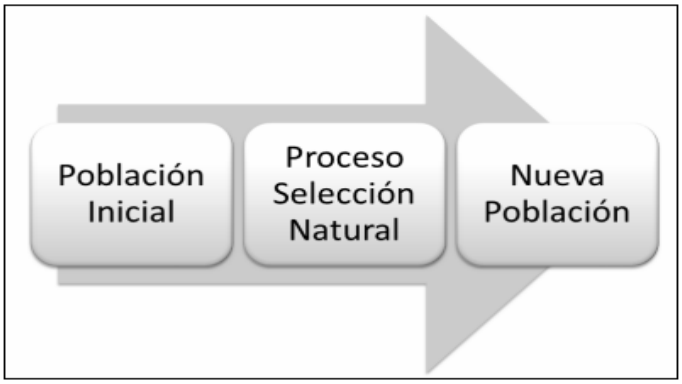
Esquema de una Algoritmo Genético



Nota. (Narváez & Saltos, 2014).

Figura 3.

Proceso del Algoritmo Genético



Nota. (Narváez & Saltos, 2014)

2.7.3 Componentes del Algoritmo Genético

Inicialización. Esta etapa es de vital importancia ya que se crea la población inicial, donde existen distintos métodos, donde cada uno de éstos debe estar en la capacidad de dar como resultado soluciones diferentes, ya que, por esta cualidad los métodos de muestreo aleatorio resultan ser muy conocidos; sin embargo, no suelen ser aplicables siempre puesto que en algunos problemas se presentan limitaciones, es decir, las soluciones factibles no siempre se consiguen.

Evaluación. En esta etapa se determina la calidad de la solución candidata. En este tipo de Algoritmos Genéticos se les llama *fitness*, el cual es un valor necesario para la selección y reemplazo, de acuerdo con la ejecución del Algoritmo Genético y del problema, este puede o no corresponder a la función objetivo.

Selección. En esta etapa el operador se encarga de seleccionar las soluciones que se utilizan para la recombinación. Por lo general, la toma de decisión se fundamenta en ajustar dichas soluciones. En el transcurso de selección por lo general opta por los mejores individuos para

producir diversos esquemas de detección de soluciones dentro de regiones en el espacio de búsqueda. A continuación, se mostrarán los esquemas utilizados.

Selección por rueda de ruleta: También llamado método de selección directa. Las soluciones se producen acorde a su *fitness* proporcional.

Selección de rango lineal: en vez de emplear el valor *fitness* de primera mano, se utiliza el rango de la solución para confrontarlo con otras soluciones.

Selección elitista: Permite que los mejores organismos de la generación actual se trasladen a la siguiente sin que estos se lleguen a alterar, donde selecciona el mejor individuo y se copia en la nueva población, los demás se eligen usando una de las formas mencionadas anteriormente lo cual garantiza que la calidad de la solución obtenida por el AG no sea disminuida de una generación en generación.

Selección por torneo: Se escogen de forma aleatoria de la población, un subconjunto de soluciones donde más adelante se escoge la mejor solución para ejecutar la recombinación.

Selección jerárquica: los individuos de la población participan en diferentes vueltas de selección, esto para cada generación donde las evaluaciones iniciales se llevan a cabo de manera rápidas, en cambio las últimas rondas son más estrictas.

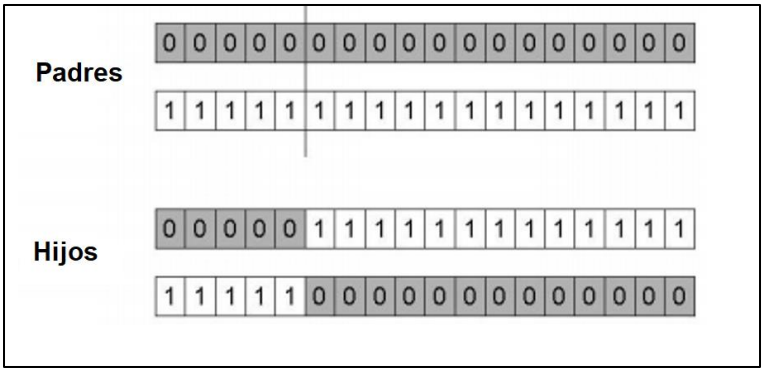
Cruce. Cuando se tienen seleccionados los individuos éstos son de nuevo combinados para producir descendencia que se introducirá en la siguiente generación. Combinando la información de los padres de manera estocástica para generar esta nueva descendencia, en donde si la combinación de dos partes es la correcta se podría obtener la nueva solución de mejor calidad que la solución anterior, para lograr esto, el operador debe estar diseñado y codificado metódicamente adaptado al problema. Existen variedad de algoritmos de cruce, pero los más utilizados son los siguientes:

Cruce de un punto: Este método es conocido también por ser el más sencillo de cruce, ya que consiste en seleccionar los dos individuos, donde se cortan sus cromosomas por un punto el cual es seleccionado de forma aleatoriamente para así originar dos segmentos que se diferencian en cada uno de ellos, después se intercambia cada cromosoma para producir los nuevos descendientes y así poder obtener la información genética de los padres.

En conclusión, se escoge un punto de cruce aleatoriamente, luego se dividen los padres en ese punto y por último se crean hijos intercambiando partes de los cromosomas. (Ver Figura 4.).

Figura 4.

Cruce de un punto

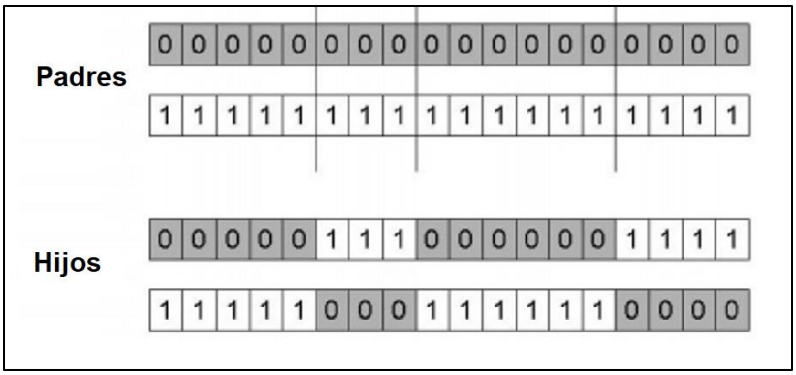


Nota. (Berzal, 2015).

Cruce por n puntos: Se eligen n puntos de cruce aleatoriamente donde se dividen los cromosomas en esos puntos. (Ver Figura 5.) Al añadir un número n de puntos de cruce que puede ser grande, puede suceder una pérdida de las características y se reduce el rendimiento del Algoritmo Genético, pero se consigue que el espacio de búsqueda del problema sea explorado más a fondo.

Figura 5.

Cruce por n puntos

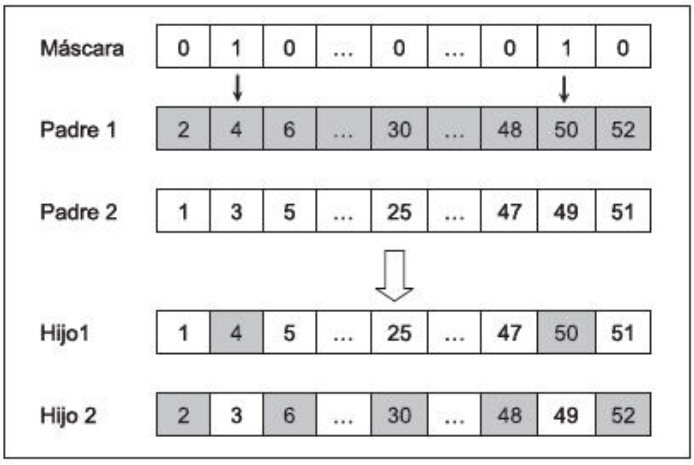


Nota. (Berzal, 2015).

Cruce uniforme: El cruce uniforme es una técnica muy diferente de las anteriores ya que cada gen de la descendencia tiene las mismas probabilidades de pertenecer a uno u otro padre, ya que se elige al azar el padre del que proviene cada gen para el primer hijo siendo el segundo hijo el inverso. Este cruce involucra la generación con valores binarios de una máscara de cruce (*crossover mask*), si en una de las posiciones de la máscara existe uno (1), el gen que se encuentra posicionado en uno de los descendientes se copie del primer padre. En caso contrario de que exista un cero (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.

Figura 6.

Cruce uniforme



Nota. Se observa la mezcla de genes de cada uno de los padres, en este, el número de puntos de cruce es fijo, pero está determinado por el tiempo medio, este cruce involucra la generación con valores binarios de una máscara de cruce (crossover mask) (González & González, 2007).

Mutación. Es una pequeña perturbación aleatoria de una solución candidata. La probabilidad de mutar a un individuo de una población es demasiado pequeña. El objetivo del operador de mutación es recuperar la información que se perdió durante la búsqueda, es decir, evitar que la población tenga una convergencia prematura (Zäpfel, Braune, & Bögl, 2010, pág. 132). Para ello es necesario usar un operador secundario, pero con baja probabilidad de realizarse.

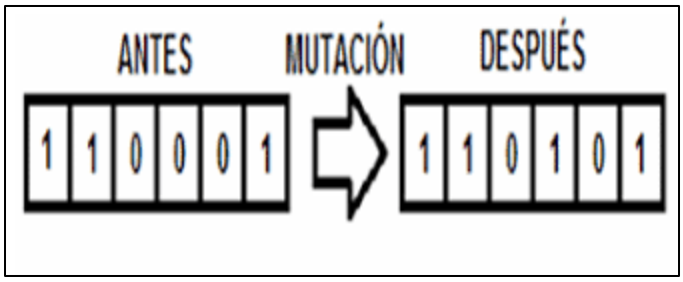
Los operadores de mutación actúan sobre el genotipo aleatorio lo que les permite diferenciarse de los demás operadores heurísticos unitarios). Su relevancia depende del tipo de la evolución del algoritmo.

Mutación estándar: Se denomina Mutación estándar, a un cromosoma al cual se le cambia el valor de cada gen con una probabilidad **Pm.**

Pm= probabilidad de mutación.

Figura 7.

Mutación estándar



Nota. (de la Fé Dotres & García, 2012).

2.8 Elementos de los problemas de secuenciación de las actividades de un proyecto

Particularmente en la etapa de planeación, las actividades y los recursos son identificados y se evalúan sus características. La secuenciación, que es el tema de interés particular, consiste en programar la ejecución en el tiempo de las actividades, sin violar las restricciones que son impuestas por las condiciones mismas del proyecto las cuales pueden ser logísticas, financieras, técnicas, de personal entre otras (Koné, 2009).

La secuenciación inmersa en la gestión de proyectos sitúa a esta dentro de las disciplinas de acción que ayudan a la gente y a las organizaciones a resolver problemas de forma práctica en el contexto de los proyectos (Shenhar & Dvir, 2008).

Los problemas de secuenciación de las actividades de un proyecto suelen componerse de cuatro elementos importantes: las actividades, las relaciones de precedencia, recursos asignados y las funciones de evaluación (Ballestín F. , 2002).

Actividades. De forma general una actividad se caracteriza por su duración, el consumo de recursos, el costo asociado y su prioridad (Koné, 2009).

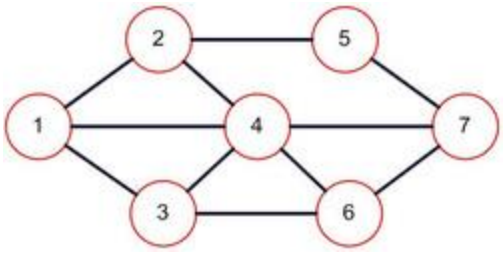
El proyecto consta de numerosas actividades que deben ser procesadas, cada actividad debe realizarse de una sola forma, y además tiene asociada una fecha de ejecución (indicando el tiempo que se debe considerar) y una fecha disponible indicando el momento donde se puede comenzar a desarrollar, además también se tiene en cuenta la forma de ejecución, el cual se refiere al a las herramientas y tecnologías, que permiten que se pueda ejecutar una actividad. Es normal encontrar que una actividad se puede realizar de diferentes formas, ya que, al momento de ejecución de las actividades, éstas tienen una fecha máxima de entrega que no afecte la duración total del proyecto.

Relación de precedencia. Los proyectos suelen estar subdivididos en actividades que son los pasos o etapas necesarias para ejecutarlos, normalmente estas actividades están relacionadas entre ellas y su nivel de relación está también dado en el nivel de afectación que una cause a otra, en otras palabras, las relaciones de precedencia representan la dependencia entre los momentos de inicio y finalización de una pareja de actividades (Gonzalez, Kalenatic, & Moreno, 2012).

Es por esto por lo que se entiende que una actividad no podría comenzar a ejecutarse si la anterior a ésta no ha finalizado, para lograr comprender mejor esta situación, se representa mediante un gráfico llamado actividad en los nodos (*Activity On Node, AON).*

Figura 8.

Actividad en los nodos (Activity On Node, AON)



Nota. En la figura cada tarea está representada por un nodo (etiquetados 1, 2,3,…,7), mientras que los arcos indican relaciones de precedencia (Klastorin, 2005).

Recursos. Normalmente para ejecutar una actividad es necesario consumir recursos, que se definen como todo elemento necesario para realizar una tarea. Hacen parte de estos el personal, los recursos económicos, los materiales y los equipos entre otros (Gonzalez, Kalenatic, & Moreno, 2012). Para el desarrollo de las actividades es necesario realizar algún tipo de consumo, este consumo se modela mediante el uso de los recursos.

En la literatura se expone la clasificación de los recursos en cuatro categorías: renovables, no renovables, parcialmente renovables y doblemente restringidos que se basan en la disponibilidad de los diferentes tipos de recursos durante la realización del proyecto.

2.8.1 Objetivos de la secuenciación de proyectos.

El objetivo que se busca con el desarrollo de esta herramienta es fijar el orden de en qué las actividades se van a ejecutar, donde se tiene como restricción la disponibilidad de los recursos que deben ser compartidos por las actividades, así como los requerimientos de cada recurso. La naturaleza de los proyectos y las actividades, están sujetos a restricciones de dichos recursos que pueden ser el personal, equipo, herramientas, materias primas, insumos y otros. Pero también existen en los proyectos anticipos, avances de recursos monetarios que están sujetos a la evolución del proyecto o al presupuesto total asignado.

2.9 Librería PSPLIB

Kolisch & Sprecher (1995) plantearon un algoritmo el cual se basa en generar instancias de una clase general de problemas de secuenciación y programación de actividades, para el RCPSP unimodal y multimodal (*ProGen*).

Este grupo de instancias generadas se reunió en una librería llamada PSPLIB. El objetivo de la PSPLIB consistió en suministrar evidencia en casos que permitiera evaluar la eficiencia de procedimientos alternos o nuevos que permitieran dar solución al RCPSP.

En la librería PSPLIB fueron desarrollados múltiples problemas aleatoriamente por medio de un diseño de experimentos el cual se basó en dos parámetros, uno fijo y el otro variable.

El primero se refiere a parámetros que se mantienen estáticos como lo son el número de actividades, duración de las actividades, cantidad de configuraciones en las cuales las actividades pueden ejecutarse, cantidad de sucesores y predecesores de las actividades, número de recursos existentes en el problema, disponibilidad máxima de cada recurso renovable y / o no renovable y consumo de cada tipo de recursos. En el segundo parámetro se hallan los que cambian, siendo aquí donde se encuentra la dificultad del problema (Kolisch, Sprecher, & Drexl, 1995).

Complejidad de la red (NC). Se refiere al número de relaciones de precedencia no redundantes de la totalidad de actividades no ficticia.

Factor de recursos (RF). Corresponde a la proporción media de la cantidad de recursos diferentes que cada activad no ficticia utilizada.

Grado de restricción de los recursos (RS). Es la relación entre el consumo de recursos y su disponibilidad.

2.10 Distribución Beta

Para la distribución beta “En la teoría de probabilidad y estadística, la distribución beta representa una familia de distribuciones de probabilidad continuas con soporte en el intervalo (0,1). La densidad beta es caracterizada por dos parámetros positivos, indicados generalmente por α y β o u y v, que son parámetros de localización y de escala. La distribución beta ha sido aplicada para

modelar el comportamiento de variables aleatorias limitadas a intervalos de amplitud finita, en una gran variedad de áreas" (González, Galvis, & Hurtado, 2018).

La distribución beta puede tomar muchas formas según los valores de α y β y se utiliza para las variables aleatorias continuas que no son negativas, por lo que su gráfica está sesgada a la derecha.

$$f(x) = \frac{\Gamma(\alpha+\beta)x^{\alpha-1}(1-x)^{\beta-1}}{\Gamma(\alpha)\Gamma(\beta)}, 0 \leq x \leq 1,\ \alpha > 0,\ \beta > 0$$

Donde,

α	parámetro de forma 1
β	parámetro de forma 2
Γ	función gamma

3. Algoritmo genético propuesto (fase proactiva)

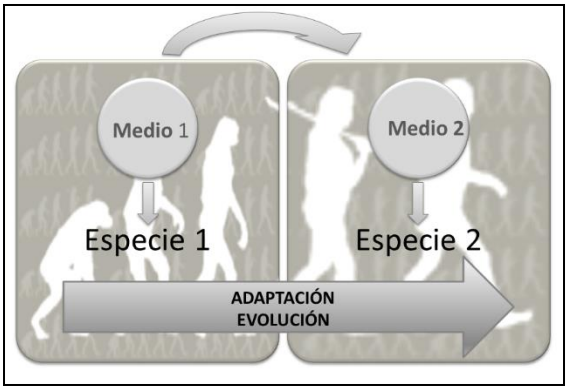
Los procedimientos para abordar el RCPSP no determinístico se diseñan con base en estrategias predictivas, reactivas y proactivas. En esta investigación inicialmente se desarrolla un algoritmo genético que pertenece a la fase proactiva; del cual se obtiene la mejor lista de actividades (mejor *makespan)*, y la línea base asociada. Posteriormente se lleva a cabo como estrategia reactiva una simulación donde se prueba la robustez de la línea base.

3.1 Filosofía de un Algoritmo Genético

Según Sacanamboy-Franco et al. (2017), un algoritmo genético consiste en un método adaptativo en búsqueda de optimización, concebido como una heurística de búsqueda inspirada en la teoría de la evolución natural de Charles Darwin en 1859. Este algoritmo refleja el proceso de selección natural en el que se seleccionan los individuos más aptos para la reproducción con el fin de producir descendencia de la próxima generación. Un esquema que se muestra en la figura 9.

Figura 9.

Proceso Evolutivo de las Especies



Nota. La imagen muestra el proceso evolutivo de las especies, (Narváez & Saltos, 2014).

El nivel de los algoritmos genéticos se genera del hecho de que se trata de una técnica robusta que se utiliza con frecuencia para encontrar soluciones óptimas o casi óptimas a problemas difíciles que, de otro modo, llevaría mucho tiempo resolver. Se utiliza con frecuencia para resolver problemas de optimización y en diferentes áreas (Holland, 1975). Aunque no se garantiza que el algoritmo genético encuentre la solución óptima del problema.

En la naturaleza, los individuos de la población compiten entre sí por recursos finitos como alimentos, agua y territorio. Incluso los miembros de la misma especie a menudo compiten para encontrar pareja. Aquellos que tienen más éxito de sobrevivir y atraer parejas tienen más probabilidades de producir una gran cantidad de descendientes. Por el contrario, individuos poco dotados producirán descendientes en menor proporción. Esto significa que los genes de los individuos más adaptables se transmitirán a más y más individuos en la descendencia. La combinación de excelentes características de diferentes antepasados a veces produce una descendencia cuya adaptabilidad es mucho mayor que la de cualquiera de sus antepasados. De esta forma, las especies pueden evolucionar para adquirir características cada vez más adecuadas a su entorno de vida.

3.2 Diseño del Algoritmo

Para la elaboración del código se utilizó Python en su versión 3.8, se escogió como lenguaje de programación para esta investigación debido a su interpretación sencilla y su versatilidad, cuya sintaxis es amigable y sin duda este elemento se destaca a la hora de programar. Cuenta con una gran cantidad de librerías de código abierto en la red donde se usó anaconda, una colección de librerías para análisis científico, y la escritura del código se hizo mediante el entorno de programación dinámica haciendo uso de la herramienta *PyCharm*, es su versión profesional,

licencia que brinda la Universidad Industrial de Santander, dicha herramienta se integra con *IPython Notebook*, la cual cuenta con una consola Python interactiva y es compatible con Anaconda, así como con múltiples paquetes científicos como *NumPy*.

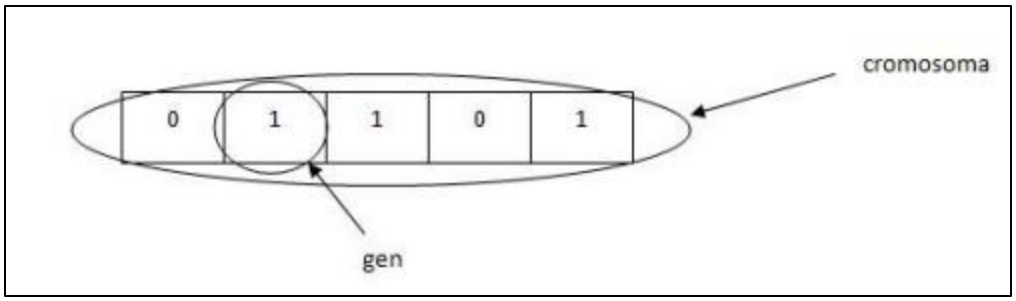
Para el Problema de Programación de Proyectos con Recursos Restringidos (RCPSP) se diseñó un algoritmo genético. Utilizando un SGS serial para la generación de la población inicial con dos reglas de prioridad GRPW y LFT. Para iniciar el Algoritmo Genético con dicha población se utilizó un SGS paralelo aplicando los operadores de selección, cruce y mutación, generando la mejor solución para el problema. Posterior a esto se realizó una simulación con la mejor solución del Algoritmo Genético en 10.000 escenarios, obteniendo los indicadores de robustez. Su codificación se presenta en el apéndice B**.**

Los individuos pueden representarse como un conjunto de genes, quienes al ser agrupados confirman a los cromosomas. Debe existir una representación de estos genes en orden de usarlos en el algoritmo genético. Se pueden considerar tres tipos básicos de codificación, a saber, binaria, entera y real.

Binaria. Esta codificación utiliza un vector de longitud donde los cromosomas son una cadena de unos y ceros y cada posición en el cromosoma representa una característica particular del problema.

Figura 10.

Ejemplo de cromosoma de representación binaria

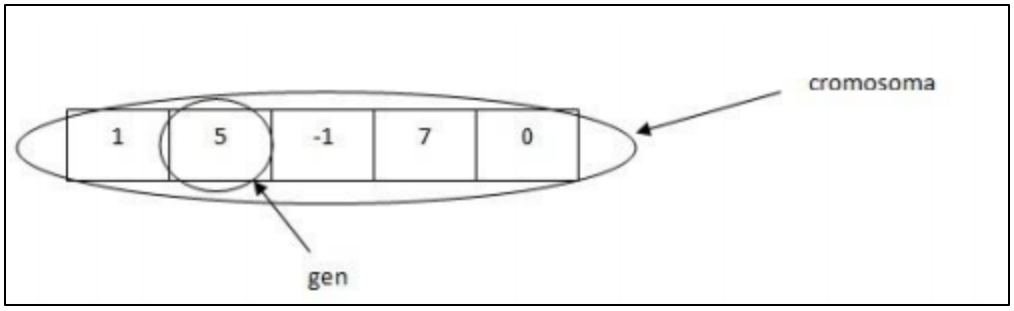


Nota: (López, 2010)

Entera. En ella se utiliza un vector cuya longitud es la del número de genes de cada individuo y el valor que puede tomar cada elemento es un número entero.

Figura 11.

Ejemplo de cromosoma de representación entera

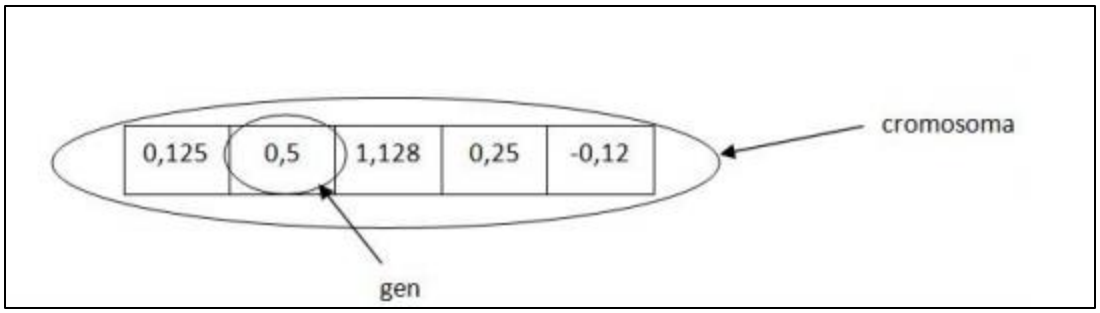


Nota: (López, 2010)

Real. En ella se utiliza un vector cuya longitud es la del número de genes de cada individuo y el valor que puede tomar cada elemento es un número real.

Figura 12.

Ejemplo de cromosoma de representación real



Nota: (López, 2010)

Según Holland (1975), el algoritmo genético opera mediante la siguiente secuencia:

1. Comenzar con una población inicial, la cual puede ser generada de manera aleatoria.
2. Calcular el fitness (aptitud) de cada individuo.
3. Aplicar el operador de selección con base en el fitness de la población.
4. Aplicar los operadores genéticos de cruce y mutación a la población actual para generar a la población de la siguiente generación.
5. Ir al paso 2 hasta que la condición de parada se satisfaga.
6. Cuando se cumple la condición de parada, se devuelve al mejor individuo encontrado (bien el mejor de todas las generaciones, bien el mejor de la última generación).

En la Figura 13 se muestra el diseño global de la estructura y funcionamiento del algoritmo propuesto.

Figura 13.

Diseño del Algoritmo Genético

3.2.1 Parámetros del Algoritmo Genético

Parámetros iniciales de corrida:

- **Población Inicial:** Cantidad de cromosomas que se van a generar en el SGS en serie.
- **Operador Selección:** Cuantos cromosomas se seleccionan de la población.
- **Operador de mutación:** Cuantas actividades se operan en mutación.
- **Iteraciones del genético:** Número de veces que se itera.
- **Prioridades:** Vector aleatorio de prioridades de las actividades.
- **Duración:** Duraciones calculadas por medio del método de duraciones redundantes.

Parámetros obtenidos de la librería

- **Número de actividades:** Cantidad de actividades a realizar teniendo en cuanta las actividades ficticias (inicial y final), dentro de la ejecución del AG.
- **Horizonte de planeación:** Horizonte de tiempo máximo de planeación del proyecto.
- **Recursos:** tipo de recurso que consume cada actividad.
- **Capacidad del recurso:** cantidad máxima de tipo de recursos disponible.
- **Precedencias:** Actividades que deben estar completadas para iniciar una nueva actividad.

3.2.2 Generación de la Población Inicial

Con relación a los algoritmos genéticos, la generación de la población inicial se considera una parte fundamental, sobre la cual se establecen las generaciones posteriores. Una tarea importante es medir el tamaño de la población ya que éste influye de manera significativa en el tiempo de ejecución del algoritmo. Es por esto que para el proceso de creación de la población inicial es necesario realizar el siguiente procedimiento:

Se crea una lista de actividades (LA) inicial, aplicando el método de la ruta crítica (sin tener en cuenta recursos restringidos) a partir de las duraciones de cada actividad obtenidas mediante el método de duraciones redundantes el cual consiste en adicionar tiempo extra a la duración original, esto para enfrentar las eventualidades que durante la ejecución del proyecto pueden suceder.

Se obtiene la población inicial del Algoritmo Genético, fijando una cantidad de individuos (cromosomas), para completar la población inicial, el segundo individuo se obtiene aplicando un esquema generador de secuencias (SGS) en serie (Ballestín, Schwindt, & Zimmermann, 2007), el cual consiste en secuenciar las actividades una tras otra, teniendo en cuenta las precedencias, recursos y prioridades. Tomando como referente dos reglas de prioridad (RP), GRPW y LFT. Para elegir la Regla de Prioridad se genera un número aleatorio y por sorteo se escoge la actividad que se va a secuenciar en ese momento.

3.2.3 Estrategia LFT

En la estrategia LFT (*Late Finish Time*), como regla, se tiene en cuenta la duración de la actividad. Se selecciona la actividad que tiene el menor tiempo de finalización. Se calcula una probabilidad de selección para cada actividad elegible j (p_j) candidata a secuenciar en ese momento (Davis & Patterson, 1975).

$$p_j = \frac{LFT_j}{\sum LFT_i}$$

Por ejemplo, si se tienen 3 actividades elegibles en un momento dado, entonces se calculan 3 p_j. supongamos que LFT de la actividad elegible 4 es 12, LFT de la actividad elegible 6 es 17 y LFT de la actividad elegible 3 es 21,

$$p_4 = \frac{12}{(12+17+21)}\ p_6 = \frac{17}{(12+17+21)}\ p_3 = \frac{21}{(12+17+21)}$$

Entonces, el LFT de las tareas sería de 11, 10 y 10 respectivamente. Donde se Selecciona de manera indiferente entre las actividades, además por tener el valor del LFT más bajo.

GRPW (*Greatest Rank Positional Weight*). Esta regla se calcula seleccionando las actividades de acuerdo con su peso posicional, el cual se obtiene sumando a la duración de la actividad, la duración de todos sus sucesores inmediatos (Álvarez-Valdés & Tamarit, 1989).

$$p_j + \sum\nolimits_i \in S_j p_i$$

Donde:

p_j es la prioridad de la actividad candidata a secuenciar en el momento

S_j es la duración de la actividad candidata a secuenciar en el momento

p_i la prioridad de las elegibles de la actividad candidata a secuenciar en el momento

Se programa la segunda actividad seleccionada y se vuelven a activar las nuevas actividades elegibles además de escoger aleatoriamente una de ellas (teniendo mayor probabilidad de selección aquella con mayor valor de RP). Esto se repite hasta que no queda ninguna actividad por secuenciar, y por lo tanto se obtiene una programación que corresponde a la segunda lista de actividades (LA).

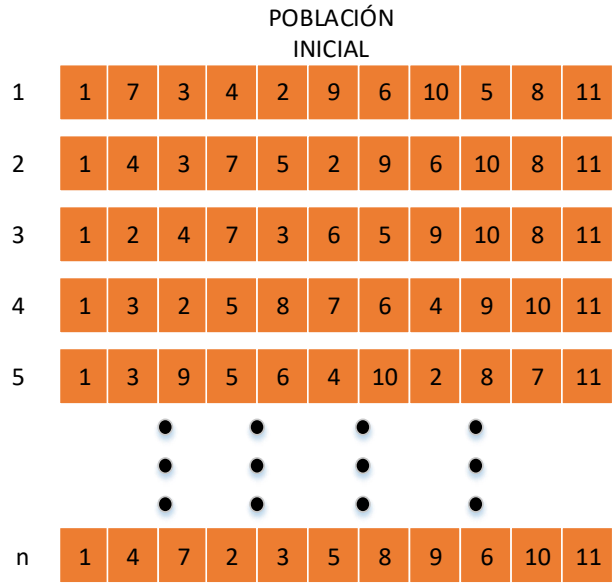
El tercer individuo se obtiene aplicando nuevamente un SGS en serie teniendo en cuenta el mismo proceso descrito para el segundo individuo.

El cuarto, quinto, sexto, etc. nPob individuo se obtiene de la misma manera. Al final se obtiene la población inicial compuesto de nPob Listas de Actividades (LA), que corresponden a nPob cromosomas o individuos.

Cuando se completa la población, inicia el AG. Aquí a cada una de las nPob Listas de Actividades (LA) se le aplica un SGS paralelo. Por lo tanto, para cada LA se obtendrá el *Makespan*. La población inicial corresponde a la primera generación del AG. De esos cromosomas (LA) se guarda en memoria el de mejor *Makespan*, como también la lista de actividades que lo generó y los tiempos de inicio (línea base).

Figura 14.

Creación de la población inicial



Nota. Elaboración propia.

3.2.4 Función Fitness

Un aspecto importante de los algoritmos genéticos es determinar la aptitud o fuerza de un individuo en la población, función que estima la calidad de la solución. Donde se evalúa la función objetivo, es decir, lo que se quiere optimizar. Teniendo en cuenta la naturaleza del RCPSP, la aptitud de un individuo será el tiempo total de ejecución del proyecto y cuanto menor sea el valor, más fuerte será la capacidad del individuo, es decir, minimizar el *Makespan* del proyecto.

Dentro de la función fitness se utiliza un SGS en Paralel el cual permite programar bloques de actividades siempre que haya disponibilidad de recursos (Ballestín, Schwindt, & Zimmermann, 2007). Dicha función inicia recibiendo la nueva población en paralelo descrita anteriormente y el vector de tiempos el cual es el vector de duraciones para cada actividad, luego se calcula el tamaño de la nueva población (s1) y se crea el vector el cual guarda los tiempos que se obtienen al final para cada cromosoma. El ciclo comienza recorriendo la población en cada iteración para cada cromosoma, se calcula el tamaño de las actividades ya que dichos cromosomas al estar paralelizados pueden tener diferentes tamaños, el siguiente ciclo anidado continua para cada cromosoma donde se calcula el vector que contiene los tiempos de cada actividad y de ese se calcula el máximo tiempo en el vector makespan.

Un algoritmo genético consta de los siguientes operadores genéticos para crear la segunda generación (hijos), a partir de la población inicial (padres) mediante las operaciones de selección, cruce y mutación.

3.2.5 Selección natural

Este proceso sirve para realizar la selección de los individuos de la población que mejor se adaptan, de tal manera que sean los progenitores de la siguiente generación. Naturalmente se tienen

en cuenta ciertos factores que definen que un individuo pueda tener descendencia. El primer factor es que consiga sobrevivir, ya sea porque logra conseguir alimento o porque no es consumido por depredadores. El segundo factor es la posibilidad de conseguir pareja para reproducirse. El último factor trata de la posibilidad de que la combinación de ambos individuos sea idónea en la creación de un nuevo individuo.

La selección natural es el proceso que aprueba al algoritmo elegir a los individuos más aptos de la población y a su vez cruzarlos de tal forma que los hijos puedan heredar las mejores características de cada padre.

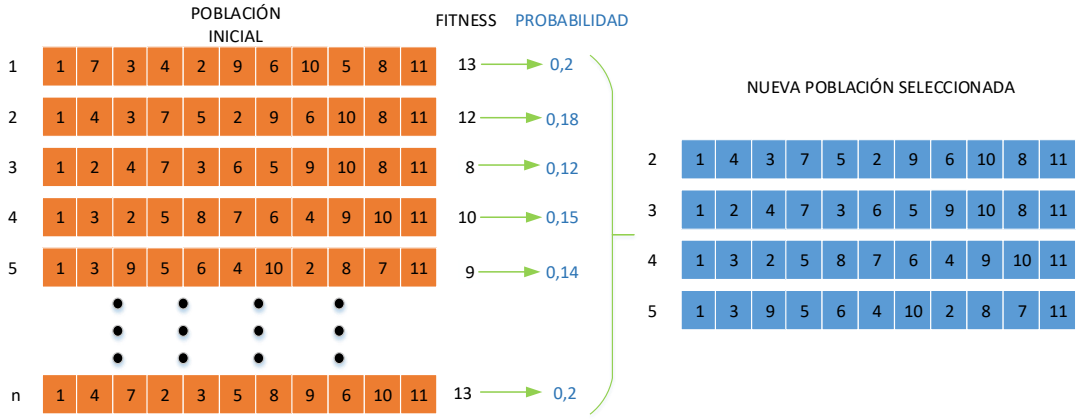
Específicamente en los algoritmos genéticos, la selección trata de un conjunto de pasos que sirven para escoger a los progenitores que pertenecen a la siguiente generación. Dichos progenitores se reproducirán en el operador de cruce generando así su propia descendencia.

Un sistema que es utilizado regularmente en los algoritmos genéticos es la selección por torneo, la cual es utilizada para este proyecto. Este sistema se basa en elegir de manera aleatoria un cierto número de individuos de la población.

El algoritmo genético del presente proyecto utiliza el criterio de selección como se describió anteriormente. Inicia recibiendo la población en serie, el *Makespan* paralelizado y el parámetro número de cromosomas que se quiere seleccionar. El primer ciclo normaliza los datos, la probabilidad de ser seleccionados es respecto a su *Makespan*. Para que en el siguiente ciclo se realice la selección por torneo.

Figura 15.

Operador de selección



Nota. Elaboración propia.

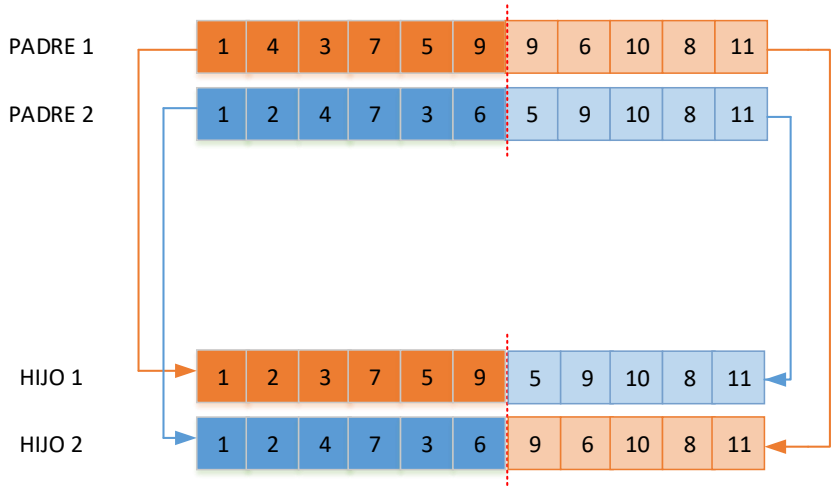
3.2.6 Cruce

El operador de cruce tiene como función cruzar o mezclar los individuos que se han seleccionado previamente en la fase anterior, esto quiere decir que los genes que pertenecen a los dos padres se mezclan entre sí para dar lugar a los diferentes hijos, estos hijos deben garantizar el cumplimiento de las restricciones de precedencia. A pesar de que existen diferentes métodos de cruce, el utilizado en el presente proyecto es cruce basado en un punto.

Cruce basado en un punto. Los dos individuos escogidos como padres son recombinados por medio de cruce por un punto, con el propósito de que luego se intercambien las secciones que se encuentran a la derecha de dicho punto. Lo anterior indica que, los genes del primer padre a la izquierda del punto de corte, hacen parte del primer hijo y los situados a la derecha formarán parte del segundo hijo, mientras que con el segundo padre sucederá lo contrario.

Figura 16.

Cruce en un punto



Nota. Elaboración propia.

El algoritmo genético del presente proyecto se enfoca en crear las parejas utilizando el operador de cruce, éste inicia recibiendo la población de los seleccionados y la matriz de precedencias, parte la población seleccionada en mitad para tener padres y madres. En el ciclo se lleva a cabo el cruce por un punto descrito anteriormente ya que cada hijo tiene una parte de ambos padres; sin embargo, los hijos deben garantizar que se cumplan las precedencias y que no tengan actividades repetidas, por esto se realiza una función de chequeo, para que todas las soluciones sean viables.

3.2.7 Mutación

El operador de mutación proporciona aleatoriedad entre los individuos de la población. Este es el que realiza la búsqueda a lo largo del espacio de posibles soluciones, es responsable de

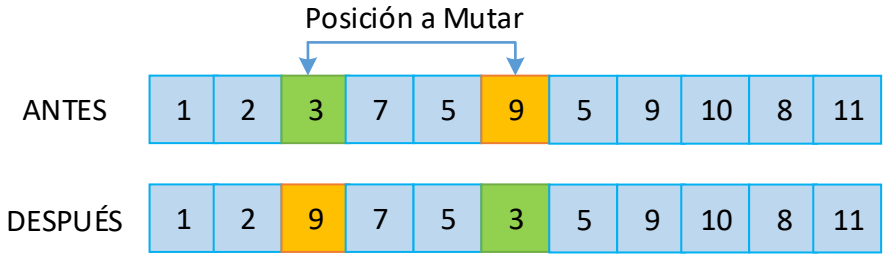
aumentar o reducir el espacio de búsqueda en el algoritmo genético y promover la variación genética de los individuos en la población.

Hay varias formas de aplicar el operador a los individuos de una población, el método más común es mutar el porcentaje del total de genes en la población. El porcentaje de genes a mutar se realiza por medio de un método fijo, que especifica el mismo porcentaje de mutaciones para todas las generaciones del algoritmo genético.

Cuando se aplica mutación sobre alguno de los individuos nuevos, es necesario asegurar que éste siga perteneciendo a la misma especie que los demás individuos, es decir, debe respetar las restricciones en cuanto a la precedencia que hacen parte del aspecto a optimizar.

Figura 17.

Mutación



Nota. Elaboración propia.

El algoritmo genético del presente proyecto utiliza el operador de mutación, que se realiza por medio de un método fijo, descrito anteriormente. Inicia recibiendo a los hijos arreglados del cruce, la matriz de precedencias y el tamaño de cromosoma que se va a mutar. Ya que realiza cambios internos de manera aleatorio dentro de cada cromosoma, para cada elemento de la población se le aplica la función, la cual garantiza que se respete las precedencias.

Después de aplicar los operadores, el algoritmo genético continua así hasta que se completen las generaciones que se definen. Al final se obtiene el mejor Makespan que quedó guardado, su correspondiente Lista de Actividades y su Línea Base.

3.3 Cálculo de las duraciones

Según Elmaghraby (2005) existen dos tipos de riesgos que pueden afectar la duración de un proyecto. El primero es el riesgo externo el cual se refiere a los eventos que hacen injerencia con el desarrollo de las actividades como falla en equipos, la pérdida de documentos, accidentes laborales, entre otros, los cuales no originan interrupción de las actividades. El segundo es el riesgo interno al que se le atribuye el margen de error originado al estimar la duración de la actividad en ausencia de riesgos externos. En esta etapa se identifican riesgos externos que se encuentran relacionados a cada actividad del proyecto, indicando su impacto y la probabilidad de que ésta ocurra. El efecto que tiene un riesgo externo en la duración es que se debe añadir una tarea extra, la cual se representa por medio de una distribución de probabilidad. De igual forma, la probabilidad de ocurrencia de cada riesgo puede ser establecida mediante la experiencia de expertos. En esta tesis, los riesgos externos son fijos, es decir, tanto la probabilidad de ocurrencia del riesgo como la duración esperada de la tarea extra a lo largo del horizonte de planeación del proyecto se consideran constantes.

3.3.1 Método de duraciones redundantes

Un procedimiento basado en el método de duraciones redundantes consiste en adicionar tiempo extra a la duración original, esto para enfrentar las eventualidades que durante la ejecución del proyecto pueden suceder.

El procedimiento presentado para hallar las duraciones está basado en el Modelo de Solución al Problema de Programación de Proyectos de Desarrollo de Nuevos Productos con Recursos Restringidos, Inserción de Tareas Y Duración Aleatoria (Ortiz-Pimiento & Díaz-Serna, 2020).

Donde se abordan cuatro procedimientos (A, B, C, D), encontrando que el procedimiento C con inserción parcial de nuevas actividades, es el que permite generar la línea-base más robusta. Es por esto que en la presente investigación se hace uso de este procedimiento, además se invita al lector a ahondar más en Ortiz-Pimiento & Díaz-Serna (2020) sobre dicho tema.

Procedimiento C. Heldman (2005) hace referencia a la exposición a un riesgo que se puede calcular multiplicando la probabilidad de que ocurra y el impacto esperado.

$$d_i = \boldsymbol{b_i} + \sum_{1}^{k} (Pr_{ki} * h_{ki})$$

La duración estimada (d_i) se calcula al sumar el valor esperado de la actividad sin riesgos ($\boldsymbol{b_i}$) con el impacto total de los riesgos, calculado al multiplicar su probabilidad de ocurrencia (Pr_{ki}) y la duración esperada de la tarea extra cuando el riesgo se materializa (h_{ki}).

La duración esperada de la actividad sin riesgos se toma de la librería PSPLIB, se utilizan dos redes de proyecto j30 y j60, los cuales se mencionan de acuerdo a la siguiente nomenclatura: una j, que significa *jobs*, continuando un número, el cual establece el número de actividades del conjunto.

En el presente proyecto se utilizan dos riesgos con una probabilidad de ocurrencia del 50% para el riesgo 1, con una duración de 0.6d y 70% para el riesgo 2, con una duración de 0.5d.

Por motivos de comparación, se toma la decisión de fijar la probabilidad de ocurrencia en las actividades donde ocurren los riesgos, por ende, se utiliza que, para las instancias con 30 actividades (J30) cinco de las diez actividades elegidas ocurre el riesgo 1 y ocho de las diez actividades elegidas ocurre el riesgo 2 ver tabla 3. Para las instancias con 60 actividades (J60) cinco de las diez actividades elegidas ocurre el riesgo 1 y ocho de las diez actividades elegidas ocurre el riesgo 2 ver tabla 4.

Tabla 3.

Ocurrencia de riesgos J30

Actividad	Riesgo 1	Riesgo 2
3	1	0
4	1	2
5	0	2
8	1	0
9	1	2
16	0	2
17	1	2
22	0	2
29	1	2
18	0	2

Tabla 4.

Ocurrencia de riesgos J60

Actividad	Riesgo 1	Riesgo 2
2	1	0
3	1	0
6	1	0

Actividad	Riesgo 1	Riesgo 2
11	1	0
12	0	1
14	0	1
20	0	1
25	1	1
27	1	0
29	0	1
30	1	0
33	1	0
38	0	1
41	1	0
43	1	1
45	1	0
50	0	1
51	0	1
56	0	1
61	1	0

3.4 Validación del Algoritmo Propuesto

Para analizar el comportamiento de los parámetros en el Algoritmo genético propuesto para resolver el RCPSP con duraciones redundantes se realizó un diseño de experimentos mediante un diseño cuasi experimental, según Hedrick (1993) “los diseños cuasi-experimentales tienen el mismo propósito que los estudios experimentales: probar la existencia de una relación causal entre dos o más variables. Cuando la asignación aleatoria es imposible, los cuasi-experimentos (semejantes a los experimentos) permiten estimar los impactos del tratamiento o programa, dependiendo de si llega a establecer una base de comparación apropiada” (p. 23).

Ya que para correr el código se requiere de un gran esfuerzo computacional, es decir, que no se realizarán experimentos aleatorios, debido a que existe una relación entre el aumento numérico de los parámetros y el porcentaje de exactitud con respecto a los resultados esperados. Se realizó por medio de Jupyter Notebook (ver Apéndice C).

Las instancias utilizadas para dicho diseño son J30 y J60, se encuentran en la librería PSPLIB, la cual es una librería de benchmark para el problema RCPSP, aplicando la modificación de las duraciones redundantes.

Tabla 5.

Instancias Utilizadas

Clase	Tamaño	Nombre de la Instancia
1	J30	j301_2
2	J30	j301_10
3	J60	j601_2
4	J60	j601_6

A partir de cada instancia anterior, se hará una variación de los parámetros de entrada del algoritmo genético, el número de población (n_pob), el número de selecciones (n_sel), el número de mutaciones (n_mut) y el número de generaciones (n_gen).

Se realizaron cinco experimentos por instancia, es decir, cinco para j301_2, cinco para j301_10 y así respectivamente para las demás instancias.

Se toma como referencia las líneas bases obtenidas del método exacto (Ortíz Pimiento & Diaz Serna, 2020) para compararlas con las líneas bases generadas por el algoritmo genético (ver

Apéndice D), en términos del makespan debido a que tanto el programa exacto, como la metaheurística aplicada para el presente proyecto se basan en el método de duraciones redundantes con igual función objetivo.

A continuación, se muestran las tablas donde se exponen los parámetros utilizados para el desarrollo del diseño de experimentos.

En las Tablas 7, 9, 11, 13 se expone el porcentaje de precisión el cual se halló en el Apéndice C, este porcentaje representa qué tan cercana es la solución del AG con respecto al makespan esperado del método exacto.

Tabla 6.

Consolidado de experimentos para j301_2

número de experimento	**1**	**2**	**3**	**4**	**5**
parámetro	numero	numero	numero	numero	numero
n_pob	240	320	360	500	800
n_sel	80	100	120	150	200
n_mut	10	15	18	25	25
n_gen	500	500	550	600	600
makespan esperado	62	62	62	62	62
makespan obtenido	105	90	81	70	64
tiempo de compilación (min)	5	10	60	180	240

Tabla 7.

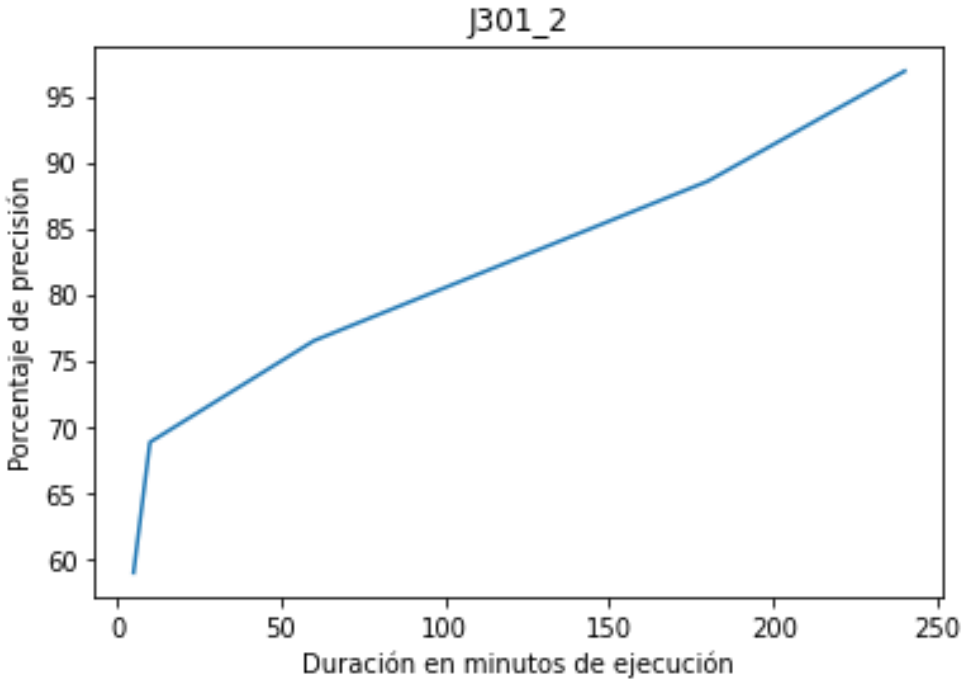
Porcentaje de precisión para j301_2

experimento	**duración (minutos)**	**makespan AG**	**makespan Esperando**	**precisión %**
1	5	105	62	59.047619

experimento	duración (minutos)	makespan AG	makespan Esperando	precisión %
2	10	90	62	68.888889
3	60	81	62	76.543210
4	180	70	62	88.571429
5	240	64	62	96.875000

Figura 18.

Duración vs porcentaje de precisión para j301_2



En la Figura 18, se observa el comportamiento de la duración en minutos de ejecución con respecto al porcentaje de precisión, lo cual indica que, entre mayor ejecución, mayor será su precisión, es decir, cuando los parámetros toman un valor grande, la posibilidad de obtener mejores resultados aumenta.

Tabla 8.

Consolidado de experimentos para j301_10

numero de experimento	**1**	**2**	**3**	**4**	**5**
parámetro	numero	numero	numero	numero	numero
n_pob	240	320	360	500	800
n_sel	80	100	120	150	200
n_mut	10	15	18	25	25
n_gen	500	500	550	600	600
makespan esperado	64	64	64	64	64
makespan obtenido	107	92	83	72	65
tiempo de compilación (min)	3	15	120	180	300

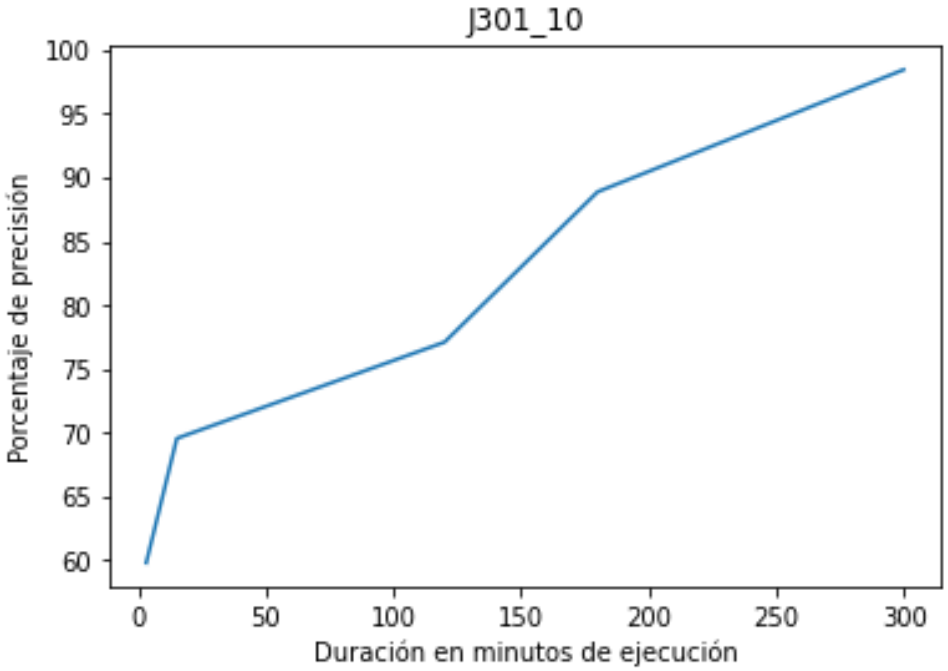
Tabla 9.

Porcentaje de precisión para j301_10

experimento	**duración (minutos)**	**makespan AG**	**makespan Esperando**	**precisión %**
1	3	107	64	59.813084
2	15	92	64	69.565217
3	120	83	64	77.108434
4	180	72	64	88.888889
5	300	65	64	98.461538

Figura 19.

Duración vs porcentaje de precisión para j301_10



En la Figura 19, se observa el comportamiento de la duración en minutos de ejecución con respecto al porcentaje de precisión, lo cual indica que, entre mayor ejecución, mayor será su precisión, es decir, cuando los parámetros toman un valor grande, la posibilidad de obtener mejores resultados aumenta.

Se puede evidenciar que los parámetros que inciden en una mayor precisión de la respuesta para las instancias de J30 que se exponen en la Tabla 6 y 8 son, la población inicial 800, el número de selecciones 200, el número de mutaciones 25 y el número de generaciones 600.

Tabla 10.

Consolidado de experimentos para j601_2

numero de experimento	1	2	3	4	5
parámetro	**numero**	**numero**	**numero**	**numero**	**numero**
n_pob	40	500	600	1000	1500
n_sel	150	200	250	300	400
n_mut	20	25	30	50	50
n_gen	1000	1000	1100	1200	1200
makespan esperado	92	92	92	92	92
makespan obtenido	135	120	111	100	96
tiempo de compilación (min)	6	60	420	660	900

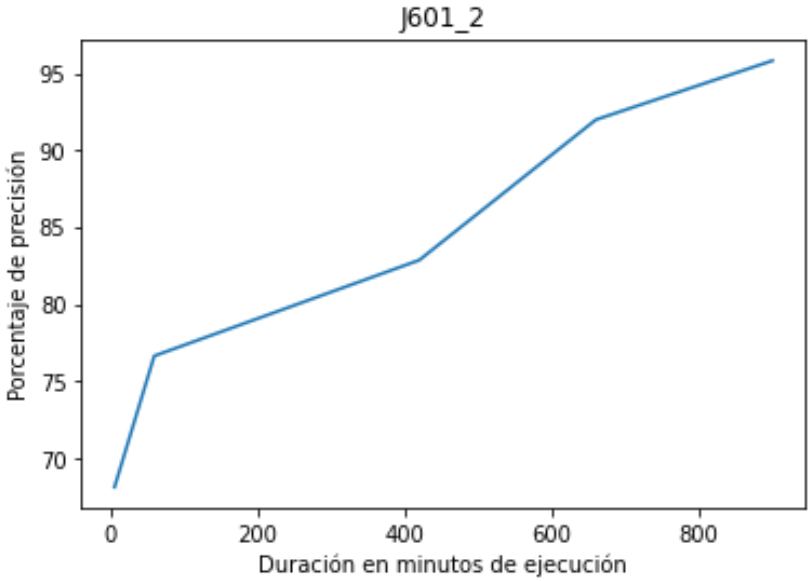
Tabla 11.

Porcentaje de precisión para j601_2

experimento	**duración (minutos)**	**makespan AG**	**makespan Esperando**	**precisión %**
1	6	135	92	68.148148
2	60	120	92	76.666667
3	420	111	92	82.882883
4	660	100	92	92
5	900	96	92	95.83333

Figura 20.

Duración vs porcentaje de precisión para j601_2



En la Figura 20, se observa el comportamiento de la duración en minutos de ejecución con respecto al porcentaje de precisión, lo cual indica que, entre mayor ejecución, mayor será su precisión, es decir, cuando los parámetros toman un valor grande, la posibilidad de obtener mejores resultados aumenta.

Tabla 12.

Consolidado de experimentos para j601_6

numero de experimento	1	2	3	4	5
parámetro	numero	numero	numero	numero	numero
n_pob	40	500	600	1000	1500
n_sel	150	200	250	300	400
n_mut	20	25	30	50	50
n_gen	1000	1000	1100	1200	1200

numero de experimento	1	2	3	4	5
parámetro	numero	numero	numero	numero	numero
makespan esperado	85	85	85	85	85
makespan obtenido	128	113	104	93	89
tiempo de compilación (min)	5	120	420	660	960

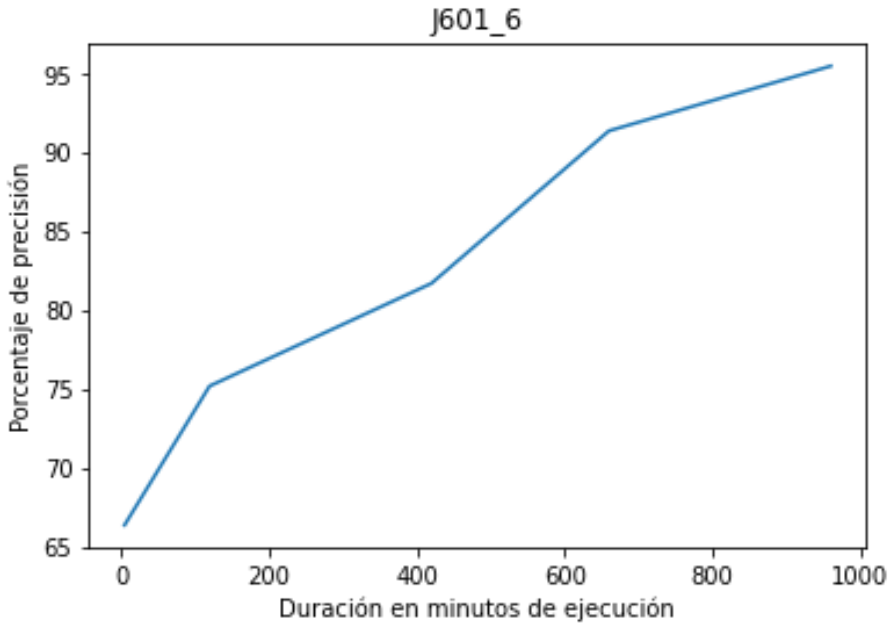
Tabla 13.

Porcentaje de precisión para j601_6

experimento	**duración (minutos)**	**makespan AG**	**makespan Esperando**	**precisión %**
1	5	128	85	66.40625
2	120	113	85	75.22139
3	420	104	85	81.730769
4	660	93	85	91.397849
5	960	89	85	95.505562

Figura 21.

Duración vs porcentaje de precisión para j601_6



En la Figura 21, se observa el comportamiento de la duración en minutos de ejecución con respecto al porcentaje de precisión, lo cual indica que, entre mayor ejecución, mayor será su precisión, es decir, cuando los parámetros toman un valor grande, la posibilidad de obtener mejores resultados aumenta.

Se puede evidenciar que los parámetros que inciden en una mayor precisión de la respuesta para las instancias de J60 que se exponen en la tabla 10 y 12 son, la población inicial 1500, el número de selecciones 400, el número de mutaciones 50 y el número de generaciones 1200. Finalmente se procede a tomar los mejores resultados para cada instancia los cuales se muestran en la Tabla 14. También se observa en la Figura 24, que las instancias de J30 alcanzan una precisión más alta que las de J60, debido al costo computacional ya que J30 corre 32 actividades

y J60 corre 62 actividades. Por lo tanto, se evidencia que el mejor resultado obtenido es el de j301_10.

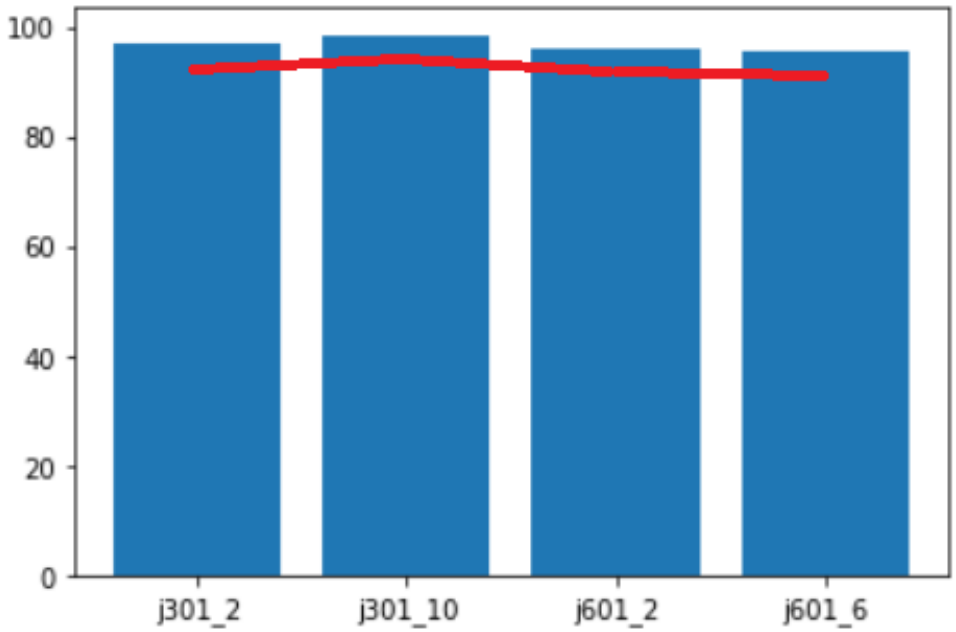
Tabla 14.

Mejores resultados para casa instancia

Instancia	% de precisión
j301_2	96.875%
j301_10	98.4615%
j601_2	95.833%
j601_6	95.505%

Figura 22.

Mejor porcentaje de precisión por instancia



De acuerdo con lo anterior, en la figura 22 se observa que, al aumentar numéricamente los parámetros para evaluarlos en el algoritmo genético, se produce un alto tiempo de ejecución, debido a que la máquina computacional utilizada no cuenta con la capacidad para desarrollar el proyecto de una mejor manera, por tal razón, para obtener resultados más precisos, el cual lleva un mayor tiempo de ejecución, se precisa de herramientas computacionales con mejores características.

4. Simulación (Fase Reactiva)

"Los procedimientos basados en el método de duraciones redundantes hacen parte del tipo de estrategias proactivas de programación, los cuales requieren de una estrategia reactiva que sirva de complemento para reprogramar las actividades durante la ejecución del proyecto" (Ortiz-Pimiento & Díaz-Serna, 2020).

Para la fase reactiva se diseñó una metodología con el fin de simular la ejecución real del proyecto y a su vez comparar las líneas base generadas del Algoritmo Genético. Su codificación se presenta en el apéndice B**.**

La programación reactiva se sustenta en un esquema generador de secuencias en paralelo, dicho generador de secuencias permite volver a planificar el orden de las actividades y hacer los ajustes necesarios en caso de aparecer cambios.

La comparación de las líneas-base de cada procedimiento se realiza teniendo en cuenta dos indicadores, robustez de la solución (SR) y robustez de la calidad (QR). El primero se calcula con los tiempos de inicio de todas las actividades, mientras el segundo se calcula con base al *makespan* (tiempo de inicio de la actividad ficticia n), respectivamente.

$$QR = \sum_{i=1}^{m} \frac{|Sn^{LB} - Sn^{ESC}|}{m}$$

Donde:

Sn^{LB} es el tiempo de inicio de la actividad ficticia n de la línea base

Sn^{ESC} es el tiempo de inicio de la actividad ficticia n de los escenarios

m corresponde al número de escenarios simulados (en este caso 10000)

$$SR = \sum_{i=1}^{n} \Delta S_i$$

Donde:

ΔS_i es la desviación media de cada una de las actividades del proyecto.

El concepto de robustez implica comparar una solución con varios escenarios de ejecución. En el presente proyecto, las líneas-base generadas a partir del Algoritmo Genético, se simulan en la programación reactiva (SGS en paralelo) para 10000 escenarios, en donde varían las duraciones (duración original + duración de la tarea extra). Para el escenario 1 se calculan las diferencias entre los n tiempos de inicio de la línea-base y el escenario 1, lo mismo se hace para el escenario 2 y así sucesivamente. Luego se promedian las 10.000 diferencias obtenidas, cabe recordar que el promedio de diferencias de la actividad n corresponde a QR. Finalmente se suman los n promedios de diferencias y obtienen SR. Además, en esta programación reactiva se va a calcular un tercer indicador, conocido como el valor esperado del *makespan*, el cual se halla promediando el *makespan* de los 10.000 escenarios simulados.

$$E(x) = \frac{\sum Makespan}{\#Escenarios}$$

5. Resultados

Los resultados de la fase proactiva y reactiva del presente proyecto, se obtuvieron mediante un procesador Intel® Pentium® N3710, con un sistema operativo Windows 10, memoria de 4GB de RAM, disco duro de 500 GB. Adicionalmente, las líneas base generadas del Algoritmo Genético desarrollado, para el Problema de Programación de Proyectos con Recursos Restringidos RCPSP no determinístico se encuentran en el Apéndice D.

Para la fase reactiva, se consolidaron los resultados en las siguientes tablas que se realizaron para 10 instancias escogidas de forma aleatoria de J30 y 10 instancias escogidas de forma aleatoria de J60. Para la interpretación de las siguientes tablas se establecieron los resultados de los indicadores de robustez, donde QR (AG) y SR(AG) hacen referencia a los indicadores del algoritmo genético y QR y SR hacen referencia al método a comparar.

5.1 Resultados J30

Tabla 15.

Resultados Indicador Robustez de la Solución

Instancia	SR (AG)	SR
j301_2.sm	348	316
j301_10.sm	301	292
j3014_3.sm	236	321
j3019_5.sm	304	277
j3020_7.sm	280	277
j3037_8.sm	432	480
j3023_9.sm	259	330
j3044_9.sm	313	286

j306_3.sm	368	336
j3047_6.sm	249	226

Tabla 16.

Resultados Indicador Robustez de la Calidad

Instancia	QR (1)	QR
j301-2.SM	28	26
j301_10.sm	24	23
j3014_3.sm	26	26
j3019_5.sm	19	20
j3020_7.sm	19	22
j3037_8.sm	31	31
j3023_9.sm	30	32
j3044_9.sm	15	17
j306_3.sm	23	20
j3047_6.sm	25	22

En la Tabla 15 y Tabla 16 se observan los resultados de los indicadores de robustez. Para la robustez de la solución se obtiene que el 60% del indicador muestra un mejor comportamiento en el caso del método exacto y el 40% el Algoritmo Genético; sin embargo, dichos resultados no tienen una variación mayor al 10% entre sí.

Para la robustez de la calidad se analiza según la Tabla 16 que el 40% de los resultados del método exacto se comportan de una mejor manera, el 40% del Algoritmo Genético tiene un mejor comportamiento y el 20 % de los resultados es el mismo en ambos métodos; sin embargo, dichos resultados no tienen una variación mayor al 12% entre sí.

Tabla 17.

Resultados Indicador Valor Esperado

Instancia	E(x) (AG)	E(x)
j301-2.SM	91	88
j301_10.sm	88	87
j3014_3.sm	100	98
j3019_5.sm	91	91
j3020_7.sm	75	74
j3037_8.sm	130	129
j3023_9.sm	102	102
j3044_9.sm	102	102
j306_3.sm	104	101
j3047_6.sm	103	97

En la tabla 17 se observan los resultados obtenidos del valor esperado. En dicha tabla se muestra el valor esperado del método exacto con un mejor comportamiento del 70% y el 30% de los resultados es el mismo en ambos métodos, dichos resultados no tienen una variación mayor al 3% entre sí.

Tabla 18.

Comportamiento del makespan

Instancia	Mejor MKS (AG)	Escenario (AG)	Mejor MKS	Escenario
j301-2.SM	57	7846	55	545
j301_10.sm	55	1896	55	9298
j3014_3.sm	70	5682	67	406

j3019_5.sm	61	4875	55	5615
j3020_7.sm	47	8940	46	337
j3037_8.sm	87	4856	86	2730
j3023_9.sm	67	8456	67	5572
j3044_9.sm	69	8132	68	7609
j306_3.sm	69	4044	65	7045
j3047_6.sm	72	9689	70	1058

Para la tabla 18 se observan los resultados obtenidos del comportamiento del makespan, en el cual se analiza que los mejores makespan se encuentran en escenarios grandes, debido a que entre mayor sea el número de simulaciones, existe una mayor probabilidad de encontrar mejores resultados.

5.2 Resultados J60

Tabla 19.

Resultados Indicador Robustez de la Solución

Instancia	SR (AG)	SR
j601_2.sm	1205	1056
j601_6.sm	838	749
j603_5.sm	736	724
j607_7.sm	1704	1520
j604_6.sm	1028	918
j6011_8.sm	736	712
j6012_10.sm	657	719
j6014_2.sm	857	718
j6017_6.sm	933	911

j6022_10.sm	887	821

Tabla 20.

Resultados Indicador Robustez de la Calidad

Instancia	QR(AG)	QR
j601_2.sm	38	38
j601_6.sm	26	28
j603_5.sm	29	33
j607_7.sm	34	38
j604_6.sm	43	48
j6011_8.sm	28	30
j6012_10.sm	30	34
j6014_2.sm	33	29
j6017_6.sm	43	46
j6022_10.sm	29	32

En la tabla 19 y tabla 20 se observan los resultados de los indicadores de robustez. Para la robustez de la solución se obtiene que el 80% del indicador muestra un mejor comportamiento en el caso del método exacto y el 20% el Algoritmo Genético; sin embargo, dichos resultados no tienen una variación mayor al 12% entre sí.

Para la robustez de la calidad se analiza según la tabla 20 que el 20% de los resultados del método exacto se comportan de una mejor manera, el 80% del Algoritmo Genético tiene un mejor comportamiento y dichos resultados no tienen una variación mayor al 10% entre sí.

Tabla 21.

Resultados Indicador Valor Esperado

Instancia	E(x)(AG)	E(x)
j601_2.sm	133	129
j601_6.sm	115	112
j603_5.sm	136	136
j607_7.sm	123	123
j604_6.sm	146	146
j6011_8.sm	116	115
j6012_10.sm	133	133
j6014_2.sm	132	124
j6017_6.sm	132	132
j6022_10.sm	129	126

En la tabla 21 se observan los resultados obtenidos del valor esperado. En dicha tabla se muestra el valor esperado del método exacto con un mejor comportamiento del 50% y el 50% de los resultados es el mismo en ambos métodos, dichos resultados no tienen una variación mayor al 2% entre sí.

Tabla 22.

Comportamiento del makespan

Instancia	Mejor MKS (AG)	Escenario (AG)	Mejor MKS	Escenario
j601_2.sm	95	4874	96	9207
j601_6.sm	85	9875	83	9644
j603_5.sm	97	8731	94	5486

Instancia	Mejor MKS (AG)	Escenario (AG)	Mejor MKS	Escenario
j607_7.sm	81	9493	84	9213
j604_6.sm	104	4678	104	3329
j6011_8.sm	82	7456	82	5174
j6012_10.sm	88	4193	91	5192
j6014_2.sm	92	8297	89	9137
j6017_6.sm	86	6879	84	5200
j6022_10.sm	96	7655	95	6271

Para la tabla 22 se observan los resultados obtenidos del comportamiento del makespan, en el cual se analiza que los mejores makespan se encuentran en escenarios grandes, debido a que entre mayor sea el número de simulaciones, existe una mayor probabilidad de encontrar mejores resultados.

6. Conclusiones

Según la revisión de literatura, en los últimos años han surgido numerosas investigaciones en torno al RCPSP. En el presente proyecto se destacan los aportes más relevantes y su búsqueda de soluciones para dicho problema mediante metaheurísticas, además se evidencia el interés de la comunidad científica en el tema; sin embargo, a lo largo de la revisión se evidencio la existencia de más información en términos determinísticos.

En el diseño de experimentos cuasi experimental se concluyó que, al aumentar numéricamente los parámetros utilizados para evaluar el algoritmo genético, se produce un alto tiempo de ejecución, debido a que la máquina computacional utilizada no cuenta con la capacidad para desarrollar el proyecto de una mejor manera, por tal razón, para obtener resultados más precisos, el cual lleva un mayor tiempo de ejecución, se precisa de herramientas computacionales con mejores características.

Debido a que se utilizan los mismos parámetros de entrada en ambos métodos, las líneas bases generadas a partir del Algoritmo Genético de la programación proactiva, son resultados aproximados de las líneas bases del método exacto, en base a que los makespan generados por el Algoritmo Genético no superan los makespan generados por el método exacto y además los indicadores de robustez son similares.

El indicador de Robustez de la calidad está en función del valor del makespan que genera la línea base de la programación proactiva, en el presente proyecto este indicador demuestra un mejor comportamiento en cuanto al Algoritmo Genético, esto debido a que los valores del makespan son más cercanos al valor esperado, por lo tanto, las diferencias son más pequeñas.

El indicador de Robustez de la solución posibilita observar la capacidad de cada línea-base para soportar las disrupciones que afectan los tiempos de inicio de las actividades del proyecto. En

este caso se concluyó que el método exacto tuvo un mejor desempeño en cuanto a este indicador, no obstante, el Algoritmo genético brinda soluciones cercanas a los resultados del método exacto con una variación máxima del 12%.

El indicador del Valor Esperado hace referencia a la estimación del makespan, es decir, la fecha de finalización del proyecto, por lo tanto, se convierte en una estrategia en la toma de decisiones para los gerentes de proyectos, debido a que el método que tenga una lista de actividades que provea un menor valor esperado tendrá mayor relevancia, ya que el objetivo es minimizar el makespan. Como se observó en los resultados el método exacto muestra un mejor comportamiento en dicho indicador; sin embargo, el Algoritmo genético brinda buenas soluciones que se encuentran cercanas a los resultados del método exacto siendo el 3% su máxima variación.

El algoritmo genético propuesto obtuvo un rendimiento homologable comparado con el método exacto ya que las respuestas que logra generar son cercanas a las óptimas en cuanto al makespan y en los indicadores de robustez en algunas ocasiones alcanza a tener mejores soluciones.

7. Recomendaciones

Para futuras investigaciones, se recomienda utilizar un equipo de cómputo que tenga las características suficientes para reducir el esfuerzo computacional, teniendo en cuanta el tipo de procesador.

Para futuros proyectos de grado, dentro de esta línea de investigación se recomienda cambiar el operador de cruce de uno a más puntos, debido a que la información genética de los hijos no siempre proporciona resultados apropiados por su único corte.

Evaluar el algoritmo genético propuesto con instancias mayores a 60 que permitan retar más el desempeño de éste.

Indagar más en el estudio de Heurísticas y Metaheurísticas cuyo objetivo sea incrementar las herramientas que den paso a desarrollar y contribuir en esta línea de investigación para el programa de Ingeniería Industrial de la Universidad Industrial de Santander.

Incentivar a los estudiantes de ingeniería industrial para profundizar en los distintos lenguajes de programación existentes, cuyo objetivo sea de fortalecer las habilidades adquiridas en la academia y así desarrollar más proyectos en este tipo de investigación ya que el desarrollo de la habilidad de programar se puede intensificar mediante ejercicios guiado por los profesores, modificando y mejorando funciones de algoritmos que se han propuesto en proyectos de grado.

Referencias bibliográficas

Álvarez-Valdés, R., & Tamarit, J. (1989). Heuristic algorithms for resource-constrained project scheduling: a review and empirical analysis. *Advances in Project scheduling , Elsevier*, 113-134.

Artigues, C., Demassey, S., & Néron, E. (2008). The Resource Constrained Project Scheduling Problem. *Resource-Constrained Project Scheduling: Models, Algorithms, Extensions and Applications*, 21-34. doi:10.1080/05695557108974820

Ballestín, F. (2002). *Nuevos métodos de resolución del problema de secuenciación de las actividades de un proyecto con recursos restringidos.* Valencia: Tesis Doctoral, Universidad De Valencia.

Ballestín, F., Schwindt, C., & Zimmermann, J. (2007). Resource Leveling in Make-to-Order production: Modeling and Heuristic Solution Method, International. *International Journal of Operations Research*(4), 50-62.

Baradaran, S., Fatemi-Ghomi, S., Ranjbar, M., & Hashemin, S. (2012). Multi-mode renewable resource-constrained allocation in PERT networks. *Applied Soft Computing Journal, 12*(1), 82-90. doi:10.1016/j.asoc.2011.09.007

Berzal, F. (2015). *Algoritmos Genéticos.* (D. D. Granada, Ed.) Obtenido de elvex.urg.es: https://elvex.ugr.es/decsai/computational-intelligence/slides/G2%20Genetic%20Algorithms.pdf

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

Brčić, M., Kalpic, D., & Fertalj, K. (2012). Resource Constrained Project Scheduling under uncertainty: . *23rd Central European Conference on Information and Intelligent Systems*, (págs. 401-409).

Browning, T., & Yassine, A. (2010). A random generator of resource-constrained multi-project network problems. *Journal of Scheduling, 13*(2), 143–161. doi:10.1007/s10951-009-0131-y

Brucker, P., Drexl, A., Mohring, R., & Neumann, K. (1999). Resource-constrained project scheduling: Notation, classification, models, and methods. *European Journal of Operational Research, 112*(1), 3-41. doi:10.1016/S0377-2217(98)00204-5

Chan, F., Wang, Z., Singh, Y., Wang, X., Ruan, J., & Tiwari, M. (2019). Activity scheduling and resource allocation with uncertainties and learning in activities. *INDUSTRIAL MANAGEMENT & DATA SYSTEMS, 119*(6), 1289-1320. doi:10.1108/IMDS-01-2019-0002

Davis, E., & Patterson, D. (1975). A Comparison of Heuristic and Optimum Solutions in Resource-Constrained Project Scheduling. *Management Science, 21*(8), 944-955. doi:10.1287/mnsc.21.8.944

de la Fé Dotres, S., & García, D. (2012). Distribución óptima de carga en emplazamientos de generadores. *Ingeniería Energética, 33*(1), 77-84.

De Reyck, B., & Herroelen, W. (1996). On the use of the complexity index as a measure of complexity in activity networks. *European Journal of Operational Research, 91*(2), 347-366. doi:10.1016/0377-2217(94)00344-0

Demeulemeester, E., & Herroelen, W. (2002). *Project Scheduling: A Research Handbook*. . Kluwer Academic Publishers.

Elmaghraby, S. (2005). On the fallacy of averages in project risk management. *European Journal of Operational Research*(165), 307-313. doi:10.1016/j.ejor.2004.04.003

Glisovic, N. (2014). Comparison of a Fuzzy Genetic and Simulated Annealing Algorithm Approach for Project Time-Cost Tradeoff. *Journal of applied mathematics*, 1-12. doi:10.1155/2014/817921

González, C., Galvis, S., & Hurtado, T. (2018). La distribución Beta Generalizada como un modelo de sobrevivencia para analizar la evasión universitaria. *Estudios Pedagógicos, 40*(1), 133-144. doi:10.4067/S0718-07052014000100008.

Gonzalez, L., Kalenatic, D., & Moreno, K. (2012). Metodología integral y dinámica aplicada a la programación y control de proyectos. *Revista Facultad De Ingeniería Universidad De Antioquia*(62), 21-32. Obtenido de https://revistas.udea.edu.co/index.php/ingenieria/article/view/11778

González, V., & González, A. (2007). Metaheurísticas aplicadas al ruteo de vehículos. Un caso de estudio. Parte 2: algoritmo genético, comparación con una solución heurística. *Ingeniería e Investigación, 27*(1), 149-157.

Hartmann, S. (1999). Project Scheduling Models. *Project Scheduling under Limited Resources, 478*, 5-10. doi:10.1007/978-3-642-58627-9

Hedrick, T., Bickman, L., & Rog, D. (1993). *Applied research design. A practical guide.* Newbury Park, CA: Sage.

Heldman, K. (2005). Risk Management: Project Managers Spotlight on. *Project Managers Spotlight on*.

Holland, J. (1975). *Adaptation in Natural and Artificial Systems.* Michigan : University of Michigan Press, Ann Arbor.

Huang, X., Dai, W., & Du, B. (2016). Resource-constrained project scheduling problem for large complex equipment: A hybrid approach using pareto genetic algorithm and interval-valued intuitionistic fuzzy sets. *Academic Journal of Manufacturing Engineering, 14*(1), 12-21. Obtenido de https://www.scopus.com/inward/record.uri?eid=2-s2.0-84990861677&partnerID=40&md5=a7e9e9aa64720ffe9c7c2070546f57df

Khalilzadeh, M., Hosseini, S., & Ghaeli, R. (2020). A genetic algorithm-based method for solving multi-mode resource-constrained project scheduling problem in uncertain environment. *Journal of project management, 5*(2), 79-86. doi:10.5267/j.jpm.2020.1.002

Klastorin, T. (2005). *Administración de proyectos*. México D.F: Alfaomega.

Klein, R. (2000). Resource-Constrained Scheduling Problems. *Scheduling of resource-constrained projects*, 73-109. doi:10.1007/978-1-4615-4629-0

Kolisch, A., Sprecher, D., & Drexl, F. (1995). Characterization and Generation of a General Class of Resource-Constrained Project Scheduling Problems. *Management Science, 41*(10), 1693-1703.

Kolish, R., & Padman, R. (2001). An Integrated Survey of Deterministic Project Scheduling. *The International Journal of Management Science, 29*, 249-272.

Koné, O. (2009). Mixed-integer linear programming formulations. *Handbook on Project Management and Scheduling, 1*, 17-41. doi:10.1007/978-3-319-05443-8

Koza, J. (1994). Genetic programming as a means for programming computers by natural selection. *Statistics and Computing, 4*(2), 87–112. doi:10.1007/BF00175355

Kucuksayacigil, F., & Ulusoy, G. (2020). Hybrid genetic algorithm for bi-objective resource-constrained project scheduling. *Frontiers of engineering management, 7*(3), 426-446. doi:10.1007/s42524-020-0100-x

Lacomme, P., Moukrim, A., Quilliot, A., & Vinot, M. (2019). Integration of routing into a resource-constrained project scheduling problem. *EURO Journal on Computational Optimization, 7*(4), 421-464. doi:10.1007/s13675-018-0104-z

Liu, S., Yung, K., & Ip, W. (2007). Genetic local search for resource-constrained project scheduling under uncertainty. *International Journal of Information and Management Sciences, 18*(4), 347–363. Obtenido de https://www.scopus.com/inward/record.uri?eid=2-s2.0-37149032066&partnerID=40&md5=21db86d5354ecb8f83ac929c71d20201

López, J. (7 de Octubre de 2010). *Introducción a los algoritmos genéticos: como implementar un algoritmo genético en JAVA*. Obtenido de https://www.adictosaltrabajo.com/: https://www.adictosaltrabajo.com/2010/10/07/jgap/

Ma, W., Che, Y., Huang, H., & Ke, H. (2016). Resource-constrained project scheduling problem with uncertain durations and renewable resources. *International Journal of Machine Learning and Cybernetics, 7*(4), 613-621. doi:10.1007/s13042-015-0444-4

Masmoudi, M., & Hait, A. (2013). Project scheduling under uncertainty using fuzzy modelling and solving techniques. *Engineering applications of artificial intelligence, 26*(1), 135–149. doi:10.1016/j.engappai.2012.07.012

Melián, M., & Glover, F. (2003). Búsqueda Tabú. Inteligencia Artificial. *Revista Iberoamericana de Inteligencia Artificial, 7*(19), 29-48.

Merkle, D., & Middendorf, M. (2002). Ant colony optimization for resource-constrained project scheduling. *IEEE Transactions on Evolutionary Computation, 6*(4), 333-346. doi:10.1109/TEVC.2002.802450.

Mika, M., Waligóra, G., & Węglarz, J. (2005). Simulated annealing and tabu search for multi-mode resourceconstrained project scheduling with positive discounted cash flows and different payment. *European Journal of Operational Research*(164), 639-668.

Mingozzi, A., Maniezzo, M., Ricciardelli, S., & Bianco, L. (1998). An Exact Algorithm for the Resource Constrained Project Scheduling Problem Based on a New Mathematical Formulation. *Management Science, 44*(5), 714-729. doi:10.1287/mnsc.44.5.714

Mogaadi, H., & Chaar, B. (2016). Scenario-based evolutionary approach for robust RCPSP. *nd International Afro-European Conference for Industrial Advancement*. *427*, págs. 45-55. Springer Verlag. doi:doi.org/10.1007/978-3-319-29504-6_6

Moreno, L., Díaz, F., Peña, G., & Rivera, J. (2007). Análisis comparitvo entre dos algoritmos heurísticos para resolver el problema de planeación de tareas con restricción de recursos (RCPSP). *DYNA: Revista de La Facultad de Minas. Universidad Nacional de Colombia. Sede Medellín, 74*(151), 171-183.

Morillo, D., Moreno, L., & Díaz, J. (2014). Metodologías Analíticas y Heurísticas para la Solución del Problema de Programaciónde Tareas con Recursos Restringidos (RCPSP): una revisión. Parte 2. *Ingeniería y Ciencia, 10*(20), 203-227. doi:10.17230/ingciencia.10.20.12

Muñoz, A., Harries, E., Contreras-Valenzuela, A., Carmona, L., Read, N., & Marcos, J. (2013). Two functional motifs define the interaction, internalization and toxicity of the cell-penetrating antifungal peptide PAF26 on fungal cells. *PLoS One, 8*(1), 1-11. doi:10.1371/journal.pone.0054813

Narváez, G., & Saltos, R. (2014). Implementación de un algoritmo genético para resolver el problema de programación de proyectos con recursos limitados. *FCNM-ESPOL, 12*(1).

Obtenido de http://www.revistas.espol.edu.ec/index.php/matematica/article/download/498/375/

Ortiz-Pimiento, N., & Díaz-Serna, F. (2020). An optimization model to solve the resource constrained project scheduling problem RCPSP in new product development projects. *Dyna, 87*(212), 179-188. doi:10.15446/dyna.v87n212.81269

Palacio, J., & Larrea, O. (2017). A lexicographic approach to the robust resource-constrained project scheduling problem. *International Transactions in Operational Research, 24*, 143-157. doi:10.1111/itor.12301

Palpant, M., Artigues, C., & Michelon, P. (2004). LSSPER: Solving the Resource-Constrained Project Scheduling Problem with Large Neighbourhood Search. *Annals of Operations Research, 131*(1), 237-257. doi:10.1023/B:ANOR.0000039521.26237.62

Rostami, M., Moradinezhad, D., & Soufipour, A. (2014). Improved and competitive algorithms for large scale multiple resource-constrained project-scheduling problems. *KSCE J Civ Eng*(18), 1261-1269. doi:10.1007/s12205-014-0401-x

Sacanamboy-Franco, M., Bolanos-Martinez, F., Bernal-Norena, A., & Nieto-Londoño, R. (2017). Genetic algorithm for task mapping in embedded systems on a hierarchical architecture based on wireless network on chip WiNoC. *DYNA, 84*(201), 202-209. doi:10.15446/dyna.v84n201.53886.

Schwiegelshohn, U. (2004). *Scheduling Problems and Solutions.* Obtenido de http://community.stern.nyu.edu/: http://community.stern.nyu.edu/om/faculty/pinedo/scheduling/sched.pdf

Shenhar, A., & Dvir, D. (2008). Project Management Research - The Challenge and Opportunity. *IEEE Engineering Management Review, 38*(2), 112-121. doi:10.1109/EMR.2008.4534315

Shou, Y., & Wang, W. (2012). Robust optimization-based genetic algorithm for project scheduling with stochastic activity durations. *Information, 15*(10), 4049-4064. Obtenido de https://www.scopus.com/inward/record.uri?eid=2-s2.0-84865446411&partnerID=40&md5=7a75825ea55b3407cdfa83fbd6e6cb21

Tahooneh, A., & Ziarati, K. (2011). Using artificial bee colony to solve stochastic resource constrained project scheduling problem. *In 2nd International Conference on Swarm Intelligence; Issue part 1. 6728*, págs. 293-302. ICSI. doi:10.1007/978-3-642-21515-5_35

Van De Vonder, S., Demeulemeester, E., Herroelen, W., & Leus, R. (2005). The use of buffers in project management: The trade-off between stability and makespan. *International Journal of Production Economics, 97*(2), 227-240. doi:10.1016/j.ijpe.2004.08.004

Verfaillie, G., & Jussien, N. (2005). Constraint solving in uncertain and dynamic environment: a survey. *Constraints, 10*(3), 253-281.

Villagra, S., Pandolfi, D., Villagra, A., Lasso, M., Rasjido, J., Mercado, V., . . . Vidal, P. (2013). Metaheurísticas avanzadas y de población descentralizada para problemas de ruteo de vehículos. *XV Workshop de Investigadores En Ciencias de La Computación*, 875-879.

Viveros, R., & Rivera, J. (2017). Formulaciones matemáticas y heurísticos simples para solucionar problemas de programación de proyectos con recursos limitados. *Universidad EAFIT, Medellín, Colombia*, 1-23. Obtenido de https://repository.eafit.edu.co/bitstream/handle/10784/11745/ViverosGutierrez_Rene_2017.pdf?sequence=1&isAllowed=y

Wei-xin, W., Xu, W., Xian-long, G., & Lei, D. (2014). Multi-objective optimization model for multi-project scheduling on critical chain. *Advances in engineering software*(68), 33–39. doi:10.1016/j.advengsoft.2013.11.004

Xiong, J., Chen, Y., Yang, K., Zhao, Q., & Xing, L. (2012). A hybrid multiobjective genetic algorithm for robust resource-constrained project scheduling with stochastic durations. *Mathematical Problems in Engineering*. doi:10.1155/2012/786923

Zäpfel, G., Braune, R., & Bögl, M. (2010). *Methauristics Based on Solution Recombination. Metaheuristic Search Concepts: A tutorial whit Applicattions to Production and Logistics.* Springer.

Zhao, Z., You, W., & Lv, Q. (2008). Applications of fuzzy critical chain method in project scheduling. *4th International Conference on Natural Computation* (págs. 473-477). ICNC. doi:10.1109/ICNC.2008.29

Zhong, C., Yu, Y., & Jia, Y. (2014). Project scheduling problem with stochastic resource-dependent activity duration. *In 2013 3rd International Conference on Mechanical Engineering. 483*, págs. 607-610. ICMEME. doi:10.4028/www.scientific.net/AMM.483.607