Desarrollo de un algoritmo genético para resolver el problema de programación de proyectos con recursos restringidos (RCPSP) y duración aleatoria, soportado en un esquema de generación de secuencias en paralelo.

Duban Alfonso Oviedo Daza y Daniel Nicolas Delgado Gómez

Trabajo de grado para optar al título de Ingeniero Industrial

Director:

Néstor Raúl Ortiz Pimiento

PhD en ingeniería

Universidad Industrial de Santander

Facultad de ingeniería Fisicomecánicas

Escuela de Estudios Industriales y Empresariales

Bucaramanga

2021

Dedicatoria

A Dios por siempre llevarme de su mano, a mi madre Mireya Gómez por ser el motor de mi vida y por todo su sacrificio, a mi padre Eduardo Delgado por su sacrificio y ejemplo, a mi hermana Natalia Delgado por llenar mi vida de alegría, a mis nonos y nonas, a mis tías Johanna y Ana, y a toda mi familia

A mis amigos, en especial a Fabian y Antonio, al programa SEA-FPC y su directora, a la familia TOV

A todas las personas que Dios puso en mi camino y aportaron su grano de arena, escuchándome, aconsejándome, enseñándome, corrigiéndome, soportándome, dándome la mano y sobre todo el empujón

Daniel Delgado

A mi padre por enseñarme todo con hechos, un día más, es un día menos, pronto todo sacrificio valdrá la pena.

A mi madre, por transmitirme su amor, por regalarme su fuerza.

A mis hermanos y amigos, por ser verde esperanza en momentos grises, por la confianza, por los momentos, por las sonrisas.

A la vida, por secarme el llanto, por arroparme en su manto, por entregarme enseñanzas, gracias a ello encontré el sentido del porque estoy vivo, es dejar algo en el mundo, es llevarme algo conmigo.

Duban Oviedo

Agradecimientos

Agradecemos a la Universidad Industrial de Santander por ser nuestra aula mater durante estos largos años, aquí dejamos en constancia tiempos de dedicación, esfuerzos, sacrificio y grandes alegrías. Es un privilegio habernos formados en sus aulas junto a compañeros y profesores, que nos dejan sus mejores conocimientos, base e inspiración para los nuevos retos y planes que depara la vida, llevando con orgullo su nombre a todos lados. Muchas gracias UIS.

Tabla de Contenido

Pág.

Lista de Tablas

Pág.

Lista de Figuras

Pág.

Lista de Apéndices

Ver apéndices adjuntos y pueden ser consultados en la base de datos de la Biblioteca UIS

Resumen

Título: Desarrollo de un algoritmo genético para resolver el problema de programación de proyectos con recursos restringidos (RCPSP) y duración aleatoria, soportado en un esquema de generación de secuencias en paralelo.[*]

Autor: Duban Alfonso Oviedo Daza, Daniel Nicolas Delgado Gómez[**]

Palabras Clave: Algoritmo Genético, Duración Aleatoria, Problema de programación de proyectos, Recursos Restringidos.

Descripción: En la presente investigación se aborda el Problema de Programación de Proyectos con Recursos Restringidos (Resource Constrained Project Scheduling Problem, RCPSP) con duración de actividades aleatoria, el cual consiste en programar actividades de un proyecto cumpliendo las restricciones de precedencias y la disponibilidad de recursos, a fin de obtener una programación base, que logre minimizar el tiempo de ejecución del proyecto bajo diferentes escenarios simulados. Para dar solución a este problema se diseñó un algoritmo genético (AG) soportado en un esquema de generación de secuencias (SGS) en paralelo, en donde se representó el cromosoma como una lista de actividades. La población inicial se generó mediante un esquema generador de secuencias (SGS) en serie, usando como reglas de prioridad el tiempo de finalización más lejano de la actividad (LFT) y el mayor peso posicional de rango de la actividad (GRPW). Para evaluar la aptitud de cada individuo se usó un esquema generador de secuencias (SGS) en paralelo, que utilizó como regla de prioridad una lista de actividades para calcular en n diferentes escenarios el makespan esperado. Como operadores genéticos se utilizaron el cruce y la mutación.

El algoritmo propuesto se comparó con un procedimiento de optimización basado en el método de duraciones redundantes propuesto en el artículo titulado An optimization model to solve the resource constrained project scheduling problem RCPSP in new product development projects (Ortíz Pimiento & Diaz Serna, 2020), utilizando 10 instancias de prueba j30 y 10 instancias de prueba j60 de la librería PSPLIB, los resultados obtenidos demuestran un rendimiento similar en ambos métodos, para la mayoría de los criterios evaluados.

---

[*] Trabajo de Grado

[**] Facultad de Ingenierías Fisicomecánicas. Escuela de Estudios Industriales y Empresariales. Director: Néstor Raúl Ortiz Pimiento. PhD en ingeniería.

Abstract

Title: Development of a genetic algorithm to solve the Resource Constrained Project Scheduling Problem (RCPSP) and random duration, supported in a parallel sequence generation scheme.[*]

Author: Duban Alfonso Oviedo Daza, Daniel Nicolas Delgado Gómez[**]

Keywords: Genetic Algorithm, Random Duration, Project Scheduling Problem, Resource Constrained.

Description: This research addresses the Resource Constrained Project Scheduling Problem (RCPSP) with a random activity duration, which consists of scheduling project activities complying with the precedence restrictions and the availability of resources, in order to obtain a base schedule that manages to minimize the project execution time under different simulated scenarios. To solve this problem, a genetic algorithm (GA) was designed supported by a sequence generation scheme (SGS) in parallel, where the chromosome was represented as a list of activities. The initial population was generated by means of a serial sequence generator scheme (SGS), using as priority rules the furthest end time of the activity (LFT) and the highest positional range weight of the activity (GRPW). To evaluate the aptitude of each individual, a sequence generator scheme (SGS) was used in parallel, which used as a priority rule a list of activities to calculate the expected makespan in *n* different scenarios. Crossing and mutation were used as genetic operators.

The proposed algorithm was compared with an optimization procedure based on the redundant duration method proposed in the article entitled An optimization model to solve the resource constrained project scheduling problem RCPSP in new product development projects (Ortíz Pimiento & Diaz Serna, 2020), using 10 test instances j30 and 10 test instances j60 from the PSPLIB library, the results show a similar performance in both methods, for most of the evaluated criteria.

[*] Trabajo de Grado

[**] Facultad de Ingenierías Fisicomecánicas. Escuela de Estudios Industriales y Empresariales. Director: Néstor Raúl Ortiz Pimiento. PhD en ingeniería.

Introducción

Los acelerados cambios en la sociedad actual requieren de proyectos eficientes que cumplan programaciones planificadas y al mismo tiempo contemplen la optimización de los recursos, haciendo de esto un trabajo minucioso y a veces complejo, en especial cuando se interrelacionan los diferentes factores que se necesitan controlar en un momento de decisión, dichos factores pueden ser: precedencias de actividades, posibles riesgos, disponibilidad de recursos, entre otros. De acuerdo a lo anterior, surgen necesidades como, acelerar el tiempo en la toma de decisiones, optimizar los diferentes tipos de recursos, minimizar las variaciones y sus riesgos, es por ello que la programación de proyectos se utiliza para garantizar una mejor ejecución, aplicándose en diversos campos como el desarrollo de nuevos productos, construcción de infraestructuras, desarrollo de software, etc.

La comunidad científica ha investigado el problema de programación de proyectos (PSP por sus siglas en inglés), que busca secuenciar de forma óptima las actividades de un proyecto respetando sus restricciones de precedencia; de este problema ha surgido una variación que añade restricciones de recursos, denominada RCPSP (por sus siglas en inglés) “problema de programación de proyectos con recursos restringido” el cual se puede abordar con dos enfoques, uno determinista, en donde todos los valores de los parámetros son fijos, y otro no determinista, que considera la variabilidad real de uno o más parámetros del problema, por ejemplo, en la duración de las actividades o en el consumo de recursos.

Esta investigación abordó el problema de programación de proyectos con recursos restringidos (RCPSP) y duración de actividades aleatorias, en investigaciones anteriores, dicho carácter aleatorio se describió por medio de distribuciones de probabilidad. El problema es

considerado NP-HARD (Ashtiani et al., 2011) debido a que no existe ningún algoritmo en tiempo polinomial que permita determinar la solución óptima al problema.

El objetivo de esta tesis es desarrollar un algoritmo genético soportado en un esquema generador de secuencias adaptable al mencionado carácter aleatorio, ofreciendo una solución con un buen nivel de robustez, y a su vez aportar al campo empresarial y académico. Desde el punto de vista empresarial, este tipo de análisis es fundamental para abordar adecuadamente la planeación y ejecución de proyectos; desde la perspectiva académica, se incentiva a la investigación relacionada con la gestión y programación de proyectos dentro del grupo de investigación OPALO.

En forma general el presente documento está organizado de la siguiente manera: el capítulo 1 expone aspectos generales del proyecto de investigación, en el capítulo 2, se realiza una revisión de literatura significativa para el problema estudiado y en el capítulo 3 se presenta el marco teórico que define los conceptos necesarios para abordar el tema investigado. Posteriormente, en el capítulo 4 se plantea el modelo matemático, que se basa en el modelo RCPSP determinístico y se adapta al carácter estocástico del RCPSP con duración de actividades aleatorias, en el capítulo 5 se describe el algoritmo genético propuesto, mediante una serie de pasos que definen la adaptación de la solución del problema (RCPSP con duración de actividades aleatoria) al modelo genético evolutivo. En el sexto capítulo se comparan los resultados hallados mediante el algoritmo genético propuesto y los obtenidos por el método propuesto en el artículo titulado *An optimization model to solve the resource constrained project scheduling problem RCPSP in new product development projects* (Ortíz Pimiento & Diaz Serna, 2020) donde se aplica un procedimiento de optimización basado en el método de duraciones redundantes, esto para instancias j30 y j60 de la librería PSBLIB. Los resultados de la investigación se analizan en el capítulo 7, posteriormente, en el

octavo capítulo se presentan las conclusiones de la investigación y finalmente el capítulo 9 se realizan unas recomendaciones y sugerencias para futuras investigaciones.

1. Generalidades

1.1 Planteamiento del Problema

Desde el apogeo del desarrollo industrial, la humanidad ha emprendido con mayor fuerza una búsqueda quizá interminable hacía la optimización de recursos, por ello, para obtener buenos rendimientos en la ejecución de proyectos, se han desarrollado modelos matemáticos y métodos que se encargan, de dar soluciones a problemas ajustados a necesidades relacionadas con recursos limitados, siendo uno de estos, el problema de programación de proyectos con recursos restringidos (RCPSP). Este problema consiste en minimizar la duración de un proyecto, sujeto a limitaciones de recursos y precedencias entre actividades, además, asume información completa sobre el uso de recursos y la duración de las actividades, determinando una programación base factible, es decir, una lista de momentos de inicio de cada actividad (Bruni et al.,2011).

La importancia de una programación base ha sido ampliamente reconocida como valioso apoyo al momento de tomar decisiones en la planeación de actividades y la asignación de recursos. Muy a menudo, la programación base es construida considerando valores promedio para la duración de las actividades y requerimientos de los recursos. Sin embargo, como lo menciona (Bruni et al.,2011), en contextos reales esta simplificación puede conducir a un rendimiento muy deficiente, en este sentido (Creemers, S., 2014) argumenta que, un supuesto fundamental del RCPSP es que las duraciones de la actividad se conocen antes de su ejecución, cuando en realidad, muchas veces nunca es el caso; debido a lo anterior, uno de los modelos desarrollados para dar una solución más acertada es el RCPSP estocástico donde, el objetivo es programar un proyecto con una solución estable en un entorno incierto.

Con el sentido de enfrentar entornos inciertos, la academia y la industria están interesadas en estimar y controlar las variaciones provocadas por eventos de ocurrencia aleatoria, que suelen surgir en el desarrollo de proyectos; estas variaciones según (Rostami, Creemers & Leus, 2018) pueden deberse a diferentes fuentes incluyendo errores de estimación, condiciones meteorológicas imprevistas, entrega tardía de algunos recursos necesarios, incidentes impredecibles, averías de las máquinas o accidentes laborales. Cuando ocurre una sobreestimación de la incertidumbre o una materialización de los riesgos (no previstos), se puede fallar en el pronóstico del proyecto perdiendo beneficios, aumentando costos y no cumpliendo objetivos cuantificados en tiempo. Es por ello que cualquier gerente de proyecto, necesita una planeación que garantice una mayor exactitud en la ejecución del proyecto, sin embargo, como lo exponen (Kutsch & Hall, 2005), el programa siempre estará expuesto a riesgos y/o comportamientos impredecibles, afectando de manera directa la eficacia en la previsión del riesgo. Por lo anterior, en el proceso de planeación de proyectos se debe tener en cuenta la variabilidad en la duración de las actividades, con el objetivo de generar una programación base que se emplee como política en la ejecución del proyecto, a fin de realizar la menor cantidad de variaciones con respecto a lo planeado, es decir lo más robusto posible.

1.2 Justificación del Proyecto

Si bien los proyectos han sido parte del trabajo del ser humano desde que él mismo se ha propuesto a alcanzar objetivos, tales como la elaboración de las primeras chozas, fabricar muebles y armas, construir sistemas de riego, acueductos, pirámides, grandes ciudades, vías que comunicaban imperios, hasta viajar a la luna, la gestión de proyectos se ha estudiado relativamente desde hace poco tiempo, ya que la mayoría de proyectos mencionados, aunque pudiesen consumir

grandes cantidades de recursos, normalmente no tenían mayores restricciones, o complejidad de la interrelación de todos los elementos que los conforman.

Como menciona (William Wallace, 2014), fue tal vez el desarrollo de la bomba atómica, el primer proyecto de complejidad que forzó a un replanteamiento y a la búsqueda de una gestión eficiente de proyectos, y pocos años después la industria de defensa de Estados Unidos experimentaba dificultades para controlar el costo y los tiempos programados de sus proyectos de sistemas de armamento a gran escala, incluidos submarinos nucleares y aviones de combate aéreo estratégicos. Se produjeron enormes sobrecostos y excesos de tiempo. El principal problema era el de intentar controlar proyectos complejos que involucraban un gran número de variables sobre las cuales el gerente no tenía ningún control inmediato, como respuesta a ello, en 1958 se desarrolló la técnica de evaluación y revisión del programa (PERT) y la compañía DuPont en 1957 creó el método de la ruta crítica (CPM, por sus siglas en ingles), y diez años más tarde, ambos métodos se combinaron con técnicas de simulación por computadora para crear un método denominado técnica de evaluación y revisión gráfica (GERT).

De acuerdo con lo anterior, se evidencia que la variabilidad en la duración de las actividades, las restricciones de recursos, la complejidad de las redes de actividades, la necesidad de cumplir con plazos más cortos, y en general, gestionar gran cantidad de variables, optimizando los resultados, llevan a la necesidad de tener gran conocimiento en la gestión de proyectos y para esto, se hace fundamental realizar investigaciones como la presentada en este documento. Específicamente, se aborda el problema de programación de proyectos con recursos restringidos (RCPSP) con duración de actividades aleatoria, logrando modelar las condiciones más importantes de los proyectos, posibilitando, una amplia aplicación a diferentes tipos de proyectos en un gran

número de áreas, y profundizando el conocimiento y herramientas sobre el tema; lo cual se traduce a una mejor planeación y gestión de proyectos.

En conclusión, cuando se planean proyectos, resulta indispensable tener en cuenta la incertidumbre para poder cumplir las metas trazadas, es por ello, que esta investigación se centra en el carácter aleatorio de las actividades, mediante el problema de programación de proyectos con recursos restringidos (RCPSP) con duración de actividades aleatorias, que permita obtener, en un tiempo computacional aceptable, una solución con un buen desempeño en términos de robustez, lo cual se traduce, en realizar pocos ajustes durante la ejecución del proyecto con respecto a la programación base.

1.3 Objetivos

1.3.1 Objetivo General

Desarrollar un algoritmo genético para resolver el problema de programación de proyectos con recursos restringidos (RCPSP) soportado en un esquema generador de secuencias adaptable al carácter aleatorio de la duración de actividades.

1.3.2 Objetivos Específicos

1. Realizar una revisión de literatura sobre el *Resource Constrained Project Scheduling Problem* (RCPSP) con duración de actividades aleatoria, así como las diferentes estrategias de optimización y los enfoques propuestos por distintos autores, con el fin de conocer variantes realizadas del problema y los métodos de solución implementados.

2. Identificar el modelo matemático del problema RCPSP con duración de actividades aleatoria, el cual se tomará como referente, para comprender su formulación, los parámetros de entrada que requiere, las principales variables de decisión y su función objetivo.

3. Desarrollar un algoritmo genético, incorporando un esquema generador de secuencias que permitirá obtener el valor esperado del makespan, de por lo menos diez redes de proyectos diferentes adaptados de la librería PSPLIB.

4. Comparar, en términos de su robustez los resultados generados por el algoritmo genético con los desarrollados por el método propuesto en el artículo titulado *An optimization model to solve the resource constrained project scheduling problem RCPSP in new product development projects* (Ortíz Pimiento & Diaz Serna, 2020) donde se aplica un método de duraciones redundantes.

5. Elaborar un artículo académico-científico de carácter publicable sobre la investigación realizada y los resultados obtenidos.

En la **Tabla 1** se menciona los capítulos en donde se evidencia el cumplimiento de los objetivos establecidos.

Tabla 1

Cumplimiento de los objetivos

Objetivo	Cumplimiento
1. Realizar una revisión de literatura sobre el *Resource Constrained Project Scheduling Problem* (RCPSP) con duraciones de actividades aleatoria.	Capítulo 2
2. Identificar el modelo matemático del problema RCPSP con duración de actividades aleatoria	Capítulo 4
3. Desarrollar un algoritmo genético, incorporando un esquema generador de secuencias que permitirá obtener el valor esperado del makespan, de por lo menos diez redes de proyectos diferentes adaptados de la librería PSPLIB.	Capítulo 5

Objetivo	Cumplimiento
4. Comparar, en términos de su robustez los resultados generados por el algoritmo genético con los desarrollados por el método propuesto en el artículo titulado *An optimization model to solve the resource constrained project scheduling problem RCPSP in new product development projects (Ortíz Pimiento & Diaz Serna, 2020)* donde se aplica un método de duraciones redundantes.	Capítulos 6 y 7
5. Elaborar un artículo académico-científico de carácter publicable sobre la investigación realizada y los resultados obtenidos	Apéndice A

2. Revisión de Literatura

Los primeros intentos por dar uniformidad a la notación, clasificación y modelo del RCPSP fue hecho por (Brucker et al., 1999) donde se empieza a hablar de la duración aleatoria de las actividades. Desde entonces se han trabajado en la literatura distintas variaciones al modelo RCPSP con un enfoque no determinístico, llamado también SRCPSP “Stochastic Resource Constrained Project Scheduling Problem” en donde la característica estocástica se puede generar por uno o más factores como: la duración aleatoria de las actividades, la disponibilidad de recursos, el consumo de los mismos, o una mezcla de ambos elementos.

También se ha abordado el problema teniendo en cuenta los distintos modos en las que se puede llevar a cabo una actividad (“Multimode Resource Constrained Project Scheduling Problem” o “MPRCSP”); y además se han agregado otras características como retrasos mínimos y máximos o la necesidad de cumplir con múltiples objetivos al mismo tiempo, todo esto ha dado una amplia variedad de problemas derivados del RCPSP que logran abarcar diferentes características que se presentan en el momento de realizar proyectos.

2.1 Fuentes de Incertidumbre en el SRCPSP

2.1.1 Duración de Actividades Aleatoria

El RCPSP con duración de actividades aleatoria ha sido definido por algunos autores como el modelo clásico RCPSP añadiéndole un enfoque estocástico, esto se ha realizado asociando una distribución de probabilidad a la duración de las actividades, tal es el caso de Ballestín (2007), Tahooneh & Ziarati (2011), Huang et al. (2011), Shou & Wang (2012), C. J. Zhong et al. (2014), Mogaadi & Chaar (2016) ,Leus et al. (2016), Rostami et al. (2018). Zaman et al. (2018), Lamas, P., Demeulemeester, E. (2016), Baradaran et al. (2010) y Chakrabortty et al. (2017). Ballestín (2007), Tahooneh & Ziarati (2011) y Shou & Wang (2012), los cuales utilizan políticas, denotada como Π, para una programación reactiva, una política define para cada tiempo de decisión $t \geq 0$ las actividades que se inician en t. En este enfoque, la duración final de la actividad se conoce solo cuando se ha completado dicha actividad, en ese sentido la solución esta dada en terminos de una política de prioridad, calculando el valor esperado del makespan que genera dicha política.

Lombardi & Milano (2010), Fang et al. (2015) y Creemers (2015) mencionaron que, también se puede representar la estructura del proyecto con un gráfico dirigido $G =(N, A)$ donde N es un conjunto de nodos y $A = \{(i, j) \mid i, j \in N\}$ es un conjunto de arcos. Los nodos representan actividades del proyecto mientras que los arcos representan relaciones de precedencia. (Lu & Li, 2008) trabajaron el SRCPSP para el desarrollo de nuevos productos, una de las aplicaciones donde este problema ha tomado mayor relevancia, para este caso los autores manejaron una curva de aprendizaje LC (Learning Curve) y un comportamiento variable del consumo de recursos DC (Resource Requirement Drop-Down Curve).

Una de las herramientas más usadas para modelar el problema ha sido las cadenas de Márkov, por ejemplo Choi et al., 2004 optaron por modelar: la duración, el resultado (niveles de

éxito que una actividad puede obtener: fracaso, éxito moderado y éxito alto) y el costo de las actividades, mediante de cadenas de Márkov, lo que les permitió a su vez modelar la correlación probabilística de la incertidumbre de los parámetros. Creemers (2015) también modelo la duración de las actividades mediante una red PERT de Márkov e investigó el impacto que genera tomar decisiones durante la ejecución de una actividad, en lugar de solo al final de una actividad. Este enfoque de cadenas de Márkov también fue utilizado por H. Li & Womer (2015), Davari & Demeulemeester (2019b) y Brčić et al. (2014).

Con respecto a la distribución que modela la variabilidad de la duración de la actividad, los autores que han trabajado con cadenas de Márkov como Creemers (2014) optaron por una distribución de tipo de fase PH. Por su parte Z. Chen et al. (2018) y Z. Chen et al. (2018) probaron diferentes distribuciones como la uniforme continua, la exponencial, la beta y la logarítmica normal, aclarando que la distribución beta es la más recomendada en la técnica de evaluación y revisión de programas (PERT), específicamente usaron la distribución PERT-beta B (a, b, p, q).

Las matemáticas difusas han sido una buena herramienta para modelar proyectos completamente nuevos y con gran incertidumbre, donde no se conoce ningún dato histórico o previo, tal es el caso de Huang et al. (2011), Masmoudi & Haït (2012), Knyazeva et al. (2015), L. Chen & Zhang (2016), Kassandra et al. (2018) y Alipouri et al. (2020). Un enfoque muy interesante es el de Csébfalvi et al. (2011), donde logran plantear un modelo unificado entre lo probabilísticos y posibilistas (enfoque difuso), haciendo una crítica a los ortodoxos difusos.

Otro método para manejar la variabilidad de las duraciones y compensar su variación ha sido el de inserción de buffer, una especie de amortiguador de tiempo que permite sopesar la variación de la duración dentro del proyecto, ya sea en cada actividad o en puntos estratégicos del

proyecto, tal enfoque ha sido abordado en investigaciones como las presentados por Zhang et al. (2011) y Roghanian et al. (2018).

2.1.2 Recursos Aleatorios

Otra de la fuentes de variación en la duración del proyecto investigadas han sido los recursos, en este sentido Lambrechts et al. (2008) abordaron el problema teniendo aleatoriedad en la disponibilidad de recursos, lo que genera una incertidumbre en la ejecución del proyecto. C. J. Zhong et al. (2014) proponen el RCPSP con duración de actividad aleatoria la cual depende de los recursos, estos últimos tienen una confiabilidad de disponibilidad.

2.1.3 Diversas Fuentes de Variación

Algunas investigaciones que han trabajado con dos fuentes de variación, por ejemplo, X. Zhong & Zhang (2015) consideraron una incertidumbre tanto en la duración de las actividades como en el consumo o disponibilidad de los recursos, con una distribución beta para la disponibilidad de los recursos y una distribución exponencial para las actividades, este enfoque de doble fuente de incertidumbre fue trabajado también por Ma et al. (2019) y Chakrabortty et al. (2019) entre otros. Browning & Yassine (2010) Abordaron el multi proyectos RCPSP o RCMPSP o MPRCSP con duraciones de actividades aleatorias donde se trabajan con recursos disponible que son compartidos y consumidos por varios proyectos, estos modelos son muy pertinentes para empresas que trabajen desarrollando varios proyectos al tiempo.

2.2 Enfoques para Resolver el SRCPSP

Por su parte Brčić et al. (2012), explican que hay tres enfoques principales para resolver los SRCPSP, esto son: procedimientos predictivos, reactivos y proactivos, o una combinación entre ellos, siendo la combinación entre un enfoque proactivo-reactivo una de las más trabajadas en la literatura; por ejemplo Ashtiani et al. (2011) abordan el problema con este último, presentando una

nueva clase de programación de políticas, en las que una serie de decisiones de secuenciación a priori se toman en una fase de preprocesamiento y las decisiones restantes se toman de forma dinámica durante la ejecución del proyecto. Otros en tener un enfoque proactivo y reactivo son Bruni et al. (2015), Ballestín (2007), Davari & Demeulemeester (2019a), con lo que llamaron PR-RCPSP (programación de proyectos proactivo y reactivo con limitaciones de recursos), entre otros.

2.2.1 SGS

Los dos tipos de esquemas de generación de programa (SGS paralelos y SGS en serie) aplicados en el RCPSP determinísticos, fueron llevados también al estocástico, sin embargo, se han diseñado nuevos esquemas de generación de secuencias adaptados a las características SRCPSP, como es el caso de Ballestín (2007) quien propuso un SGS estocástico para el SRCPSP, una mezcla de los SGS en serie y paralelos. Chakrabortty et al. (2020) al ver que, los SGS que solo están basados en métodos paralelos, son a menudo criticados por no determinar secuencias óptimas para el problema y los que están basados en serie no se pueden aplicar en un enfoque reactivo, deciden trabajar con un DSGS (esquema de generación de programación dinámica). Bruni et al. (2011) por su parte, trabajaron con un esquema de generación dinámica estocástica SDGS.

2.2.2 Herramientas para Resolver el RCPSP con Duraciones de Actividades Aleatoria

Los métodos de simulación fueron usados por autores como: Fernandez et al. (1998), Choi et al. (2004), Lu & Li (2008) , S. Li et al. (2012), Tseng & Ko (2016) y Z. Chen et al. (2018); y ha sido el complemento para otros métodos, tal es el caso de Pet-Edwards & Mollaghasemi (1996), quienes trabajaron uniendo un algoritmo genético y un método de simulación. Los métodos de búsqueda han sido abordados por Rostami et al. (2018) y Chakrabortty et al. (2020), quienes propusieron heurísticas de búsqueda local, Chakrabortty et al. específicamente desarrollaron una

búsqueda local de vecindad variable (VNSLSH). Baradaran et al. (2010) diseñaron un algoritmo de búsqueda de dispersión hibrida, Danka (2014) propusieron un algoritmo de búsqueda de armonía (HS) y Leus et al. (2016) plantearon un algoritmo de búsqueda de dos fases GRASP (Greedy Randomized Adaptive Search Pro-cedure).

Otros autores y sus respectivas propuestas se mencionan a continuación: Bruni et al. (2011) algoritmo de descomposición por etapas y Bruni et al. (2018) algoritmo con descomposición de Benders. Tahooneh & Ziarati (2011) algoritmo de colonia de abejas, Lombardi & Milano (2012) algoritmo de flujo mínimo para detectar conjuntos críticos mínimos (MCS), Lamas & Demeulemeester (2016) aproximación promedio SAA - Ramificación y corte, Yu et al. (2012) programación dinámica basada en agente, Brčić et al. (2014) adaptación del algoritmo de ruta más corta para los Gráficos Acíclicos Dirigidos (DAG), Fang et al. (2015) algoritmo de estimación de la distribución, H. Li & Womer (2015) métodos efectivos y algoritmos eficientes de programación dinámica aproximada (ADP) hibrido, Zaman et al. (2018) algoritmo consolidado de optimización (COA) y Chakrabortty & Ryan (2019) heurística basada en búsqueda de vecindario variable (VNSH) y heurística del factor de flotación dependiente del flujo de recursos (RFDFF). Chakrabortty et al. (2017) propusieron 6 métodos para convertir las duraciones en determinísticas, entre ellas la Branch and Cut (B&C).

2.2.2.1 Algoritmos Genéticos. Los métodos evolutivos, en especial el algoritmo genético (AG) ha sido uno de los más usados por los investigadores en los últimos 20 años para resolver el RCPSP con duración de actividades aleatorio.

Liu et al. (2007) propusieron un algoritmo llamado búsqueda genética local específica, un AG elitista que trabaja únicamente en un conjunto de números locales, utilizando el valor de prioridad para representar cada cromosoma; luego de obtener los hijos hacen una búsqueda local

generando una solución vecina a estos, proceso que se detiene si la solución no se ha mejorado durante 10 iteraciones continuas. Los operadores usados en el proceso fueron: selección de ruleta, cruce de un punto y mutación. Como parámetros de entrada definieron: un tamaño de población popSize= 60, una probabilidad de cruce pc = 0,9, probabilidad de mutación pm = 0,05 y elitismo = 5 con una cantidad de generaciones Gen = 40 para la búsqueda local y Gen = 400 para el GA. Para codificar y descodificar el cromosoma utilizaron un método en paralelo.

Ballestín (2007) representó cada cromosoma como una lista de actividades; para generar la población inicial el autor propuso un muestreo aleatorio sesgado basado en el arrepentimiento, el cual trabajaba con la regla de prioridad del tiempo de finalización más lejano (LFT) para calcular la probabilidad de seleccionar una actividad (dentro de un conjunto de elegibles a programar). Para obtener la aptitud del individuo, el AG de Ballestín realiza una redefinición de su respectiva lista de actividades aplicando un SGS en serie o en paralelo, obteniendo así una lista de actividades prima, con esta última, es calculada la aptitud del individuo en *d* escenarios distintos de acuerdo con el makespan generado por la misma. Ballestín uso un operador de cruce de 2 puntos y un operador de mutación para su AG, además, conformó las parejas de padres aleatoriamente entre todos los individuos.

Ballestín (2007) trabajó con instancias de prueba J120 de la librería PSPLIB, en donde experimentó con un máximo de 25000 escenarios para la fase de cálculo de la aptitud, aunque después demostró mejores resultados con menos de 100, sin embargo, para probar la solución final obtenida por el AG trabajó con 1000 escenarios; en cuanto a la distribución que modela las duraciones de las actividades utilizó las siguientes tres: $U\ (d_i \pm \sqrt{d}_i\ ), U(0, d_i)$ y $\text{Exp}(d_i)$, mientras que, parar compararse con métodos de otros autores también usó la distribución beta con parámetros α= 2 y β = 3.

Huang et al. (2011) trabajaron su modelo matemático con parámetros de tiempo difusos y propusieron un algoritmo genético basado en un esquema de generación de programación paralela difuso, en el cual usaron como representación de cada cromosoma una lista de actividades. Para generar la población inicial optaron un método de muestreo aleatorio puro; en cuanto a la aptitud de cada individuo los autores la calcularon como la inversa de la integral del tiempo de finalización difusa de la última actividad; los operadores utilizados por ellos fueron: selección de dos torneos (garantizaron que los dos mejores individuos se cruzaran), el operador de cruce de un punto y el operador de mutación; el algoritmo lo pusieron a prueba con instancias J30 de la librería PSPLIB, con tamaño poblacional de 50 individuos para un total de 100 generaciones, con probabilidad de cruce de 0,9 y probabilidad de mutación de 0,01.

Ashtiani et al. (2011) propusieron un Algoritmo Genético de dos fases con búsqueda local, representando cada cromosoma como una lista de actividades; para crear la población inicial, los autores adaptaron el muestreo aleatorio sesgado basado en el arrepentimiento trabajado antes por Ballestín (2007), de este autor tambien comparten el metodo para calcular la aptitud del individuo; la seleccionan de los cromosomas la hicieron por medio de una la clasificación simple, eligiendo a los mejores individuos; los autores trabajaron con un operador de cruce de dos puntos, un operador de mutación y agregan enseguida un operador llamado doble justificación, para finalmente mejorar los resultados del AG con una busqueda local; el algortimo fue analizado con instancias j120 de la librería PSPLIB, trabajando con distribuciónes: uniforme, exponenciales y distribución beta.

Shou & Wang (2012) propusieron un algoritmo genético bi-objectivo, donde utilizaron una representación directa del cromosoma con un vector de tiempos de inicio; el AG crea población inicial con un enfoque aleatorio puro y evalúa la aptitud del individuo calculando el valor inverso

de la función objetivo de su modelo matemático. Los autores usaron para su algoritmo un operador de cambio de subred aplicado solo a los dos mejores cromosomas, mientras que, todos los individuos fueron sometidos a los siguientes operadores: selección de ruleta, cruce de un punto y 3 diferentes operadores de mutación (uniforme, incremental, no uniforme); en algunos casos después de aplicar estos operadores, era posible que el cromosoma no comenzara desde el período de tiempo cero, para esto se plantearon un operador de desplazamiento a la izquierda.

Shou & Wang pusieron a prueba su AG solucionando 480 instancias de prueba j30 generados con ProGen, trabajando con probabilidades de cruce de 0,3 , 0,6 y 0,9; y de mutación de 0,01 , 0,05 y 0,1, en donde, la probabilidad de cruce de 0,9 y de mutación de 0,05 fueron las que ayudaron a generar mejores resultados. En cuanto a las generaciones, estos autores probaron con valores entre 1 a 500 generaciones, con población entre 10 a 5000 individuos, de acuerdo con los resultados obtenidos encontraron mejores resultados con 125 generaciones y un tamaño de población de 40.

Mogaadi & Chaar (2016) en su algoritmo genético utilizaron la lista de actividad para representar el cromosoma, y para descodificarlo optaron por el esquema de generación de programación en serie; la población inicial del algoritmo la crearon mediante la generación de múltiples escenarios y una heurística de muestreo de múltiples pasos, con lo cual, obtenían una solución aproximadamente óptima y por ende individuos iniciales con una buena calidad. Los autores usaron para medir la aptitud del individuo la robustez de este, también trabajaron con el operador de selección por ranking, un cruce de dos puntos de orden lineal y un operador de mutación. Mogaadi & Chaar además, incluyeron en su algoritmo la mejora hacía adelante y hacía atrás, la cual llamaron FBI (forward-backward improvement), esta mejora fue antes trabajada por

Csébfalvi et al. (2011) en su método heurístico. El AG lo probaron en la librería PSPLIB con instancias J30.

Chand et al. (2016) propusieron un algoritmo evolutivo simple basado en población, para representar cada cromosoma optaron por una lista de actividad, posteriormente para decodificarlo, utilizaron un esquema de generación de horarios en serie, en donde calcularon la robustez del individuo para después usarla como el valor de su aptitud. La población inicial la crearon con un método aleatorio, mientras que los operadores trabajados en el algoritmo fueron: selección de torneo de tres opciones, un cruce de 2 puntos, un operador mutación de inserción basado en la ventana de movimiento (asociado a un porcentaje de inserción) y un operador de reparación para corregir las posibles inviabilidades de precedencia dentro del cromosoma. Para el análisis del algoritmo usaron la librería PSPLIB con instancias J30, con un tamaño de población de 10 y un porcentaje de inserción del 10%.

Wang et al. (2019) plantearon un modelo bi-objectivo, buscando minimizar el costo de transferencia y maximizar la solidez de la solución, en presencia de variabilidad en la duración de la actividad, para ello propusieron dos metaheurísticas, un NSGA-II (algoritmo genético de clasificación no dominado) y un PSA (apareamiento simulado de Pareto); en el NSGA- II, para representar cada cromosoma los autores diseñaron una matriz bidimensional binaria que indicando cómo se transfieren los recursos de una actividad a otra. La metaheurística propuesta por Wang et al., crea la población inicial aleatoriamente, hace uso de una clasificación elitista y para comparar la calidad de diferentes soluciones usa estimación de distancia abarrotada, complementándola con una selección de torneo, además utiliza un cruce de dos puntos y un operador de mutación; los autores trabajaron con instancias de prueba J30 de la librería PSPLIB.

3. Marco Teórico

3.1 Problemas de Programación de Proyectos (PSP)

Según Ortiz & Díaz (2019) el problema de programación del proyecto (PSP, por sus siglas en inglés) es un nombre genérico para cada problema que se centra, en la optimización de duración del proyecto, la asignación de los recursos del proyecto, los costos estimados del proyecto y el flujo de caja del proyecto, entre otros. Para lograr estos objetivos principales, el PSP apunta a generar una programación de actividades, organizadas de acuerdo con un criterio de decisión para dar una solución adecuada al problema abordado, el PSP básico ha dado origen a otros problemas que tienen en cuenta más aspectos en la programación de proyectos, tal es el caso del problema de programacion de proyectos con resurso restringidos (RCPSP por sus siglas en ingles).

3.1.1 Problema de Programación de Proyectos con Recursos Restringidos Estocástico (SRCPSP)

El RCPSP tiene en cuenta la necesidad de tener que cumplir con las restricciones de recursos a las que están condicionados los proyectos, ya que estos recursos suelen ser escasos y deben ser compartidos por varias actividades al tiempo. Para hacer más realista el problema, se puede tener en cuenta el carácter estocástico de los proyectos, planteando así el SRCPSP (Stochastic Resource Constrained Project Scheduling Problem), en este sentido Choi et al. (2004) mencionan que el SRCPSP se caracteriza por ser una secuencia de decisiones combinatorias tomadas con respecto a un portafolio de actividades y la asignación de recursos, los cuales pueden depender del estado del sistema en el momento de la decisión, por lo cual el problema se puede clasificar como un "problema de optimización estocástica de múltiples etapas con recurso" o un "problema de control de retroalimentación óptimo estocástico".

3.1.1.1 RCPSP con Duración de Actividades Aleatoria. Este es un caso particular del SRCPSP, en donde el factor estocástico del problema es dado por la duración aleatoria de actividades, estas variaciones son debido a los posibles retrasos y/o adelantos, internos o externos, que se puedan presentar en la ejecución del proyecto.

3.2 Método de la Ruta Crítica (CPM)

Es un método usado que permite planear la programación de un proyecto para su posterior control durante la ejecución, teniendo como base los tiempos en los que inician y terminan cada actividad para determinar cuáles son las actividades que determinan directamente la duración total del proyecto al no tener holguras para retrasos.

3.2.1 Lenguaje CPM

3.2.1.1 Diagrama de RED. Este diagrama está compuesto por un conjunto de nodos que representan las actividades del proyecto y unos grafos dirigidos que los conectan representando la relación de precedencia entre ellas.

3.2.1.2 Ruta Crítica. Es la secuencia de actividades que determina la duración total del proyecto al no tener holguras para retrasos o adelantos.

3.2.1.3 Actividad Precedente. Es una actividad que requiere ser terminada para que se inicie la siguiente.

3.2.1.4 Tiempo de Inicio más Temprano ES (Early Start). El tiempo más pronto en el que puede empezar la actividad.

3.2.1.5 Tiempo de Finalización más Temprano EF (Early Finish). El tiempo más pronto en el que puede terminar la actividad.

3.2.1.6 Tiempo de Inicio más Lejano LS (Late Start). El tiempo más tarde en el que puede comenzar la actividad sin retrasarse el término el proyecto.

3.2.1.7 Tiempo de Finalización más Lejano LF (Late Finish). El tiempo más tarde en el que puede terminar la actividad sin retrasarse el término del proyecto.

3.3 Problemas de Optimización Combinatoria

El RCPSP con duración de actividades aleatoria hace parte de los problemas denominados de optimización combinatoria, los cuales según Martí (2001) tiene el objetivo de encontrar el máximo (o el mínimo) de una determinada función sobre un conjunto finito de soluciones (con variables discretas) que se denotan por S, donde el número de elementos de S es muy elevado, haciendo impracticable la evaluación de todas sus soluciones para determinar el óptimo, esta dificultad se puede medir en la llamada complejidad computacional; cabe aclarar que no se exige ninguna condición o propiedad sobre la función objetivo o la definición del conjunto.

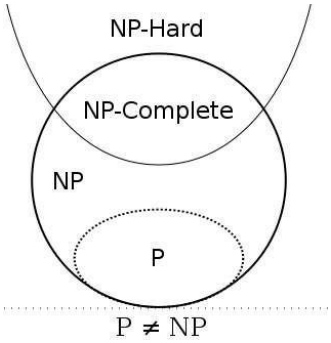
3.3.1 Complejidad Computacional

Para Palace T & Postrand G (2018) la teoría de la complejidad computacional analiza recursos computacionales como: el tiempo, cantidad de pasos en un cálculo y espacio usado de almacenamiento (en un equipo de cómputo), necesarios para resolver un problema de optimización combinatoria, programación lineal, criptología entre otros; todo esto con el fin de clasificar dichos problemas según su complejidad.

3.3.1.1 Niveles de Complejidad Computacional. Los niveles de complejidad computacional según Moreno Díaz et al. (2007) son: clase P, clase NP, clase NP-Completos y NP-Hard.

Figura 1

Niveles de complejidad computacional



Nota. Representación de los niveles de complejidad computacional. Tomada de Por Dow J. (2019).

- *Problemas de la clase P.* En esta clase se incluyen los problemas que se pueden resolver computacionalmente en tiempo polinomial, pudiendo ser resueltos por una máquina de Turing.

- *Problemas de la clase NP.* A esta clase pertenecen los problemas que se pueden resolver utilizando una máquina imaginaria llamada máquina de Turing no determinista en un número de pasos polinomial.

- *Problemas de clase NP-Completos.* Son más difíciles de resolver que los NP, y cumplen la propiedad que todo problema de la clase NP se puede transformar polinomialmente en NP-Completos.

-*Problemas de clase NP-Hard.* A esta clase pertenecen los problemas para los cuales no existe ningún algoritmo en tiempo polinomial que permita determinar la solución óptima al problema, por tanto, no hay forma de garantizar que la solución encontrada sea el óptimo global.

3.3.1.2 Complejidad del RCPSP con Duración de Actividades Aleatoria. En el caso del RCPSP con duración de actividades aleatoria es considerado como un problema NP-hard por varios autores como por ejemplo por (Ashtiani et al., 2011) debido a que el esfuerzo computacional requerido para dar solución a este tipo de problemas es elevado, esto se ve en que

cualquier algoritmo polinomial no determinista (NP) para solucionar el problema, no puede ejecutarse en tiempo polinomial.

La complejidad del RCPSP con duración de actividades aleatoria depende de varios factores como los son: el número de actividades, el número y disponibilidad de recursos analizados, y la complejidad de la red de proyecto, los cuales pueden ser medidos utilizando indicadores de complejidad, los más conocidos son:

- *Complejidad de la Red (NC).* Este indicador tiene en cuenta el número medio de arcos no redundantes por nodo incluidas las actividades ficticias.

- *Factor de Recursos (RF).* Mide el nivel de dependencia entre actividades y tipo de recursos: si RF=1, cada actividad requiere todos los tipos de recurso, pero si RF=0, las actividades no requieren ningún tipo de recurso.

- *Fuerza de Recursos (RS).* Mide la fuerza de las restricciones de recurso, para RS<1 tienen recursos limitados.

3.4 Variable Aleatoria y Distribución de Probabilidad

Uno de los elementos principales del RCPSP con duración de actividades aleatoria, es el uso de las variables aleatorias, también llamadas estocásticas. Spiegel & Stephens (Spiegel & Stephens, 2009) define así una variable aleatoria "Si una variable X toma un conjunto discreto de valores $X_1, X_{2,} \dots X_k$ con probabilidades respectivas $p_1, p_2, \dots p_k$, donde $p_1 + p_2 \dots + p_k = 1$ esto se define como una distribución de probabilidad discreta de X. La función $P(x)$, que tiene los valores $p_1, p_2, \dots p_k$, para $X = X_1, X_{2,} \dots X_k$ respectivamente, se llama función de probabilidad o función de frecuencia de X." En otras palabras, una variable aleatoria X es aquella que puede tomar ciertos valores con determinadas probabilidades.

3.4.1 Distribución Beta (p,q)

En el modelo matemático expuesto en el presente trabajo se escogió la función de distribución beta para modelar la duración de las actividades, ya que una de las principales ventajas de esta distribución es que se puede ajustar a una gran variedad de distribuciones empíricas, esto cambiando los valores de los parámetros de forma p y q, mediante los que viene definida la distribución, por esta cualidad dicha distribución es la más recomendada por (Z. Chen et al., 2018) en la técnica de evaluación y revisión de programas (PERT) y esto se ve reflejado en el uso de la misma en gran variedad de artículos.

Según González C et al. (2014) "…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 p y q o u y v, que son parámetros de localización y de escala." González C et al. (2014), en donde su función de densidad puede tener variados comportamientos dependiendo de los valores asignados a los parámetros, desde comportamientos simétricos hasta totalmente asimétricos.

Su densidad es:

Figura 2

Función de densidad Por (González C et al., 2014)

$$f(x)=\frac{\Gamma(u+v)}{\Gamma(u)\Gamma(v)}(x)^{u-1}(1-x)^{v-1};x\in(0,1)$$

3.5 Tipos de Recursos

Otro de los elementos importantes que le dan forma al RCPSP con duración de actividades aleatoria, (Brucker et al. 1999) son los recursos. Brucker menciona que en cualquier SRCPSP se

puede trabajar tres tipos de recursos: los renovables, los no renovables y los parcialmente renovables.

Los recursos renovables, son aquellos que tienen una disponibilidad fija en cada periodo sin importar que en el anterior se haya consumido parcial o totalmente, como el caso de las horas disponibles de alguna máquina; los recursos parcialmente renovables varían su cantidad disponible en cada periodo y sirven para modelar regulaciones laborales complicadas siendo una herramienta fundamental para elaborar secuencias y turnos programados y los no renovables son aquellos de los cuales solo se dispone una cantidad en un solo periodo, para los otros periodos aunque se pueda consumir, no se contará con una nueva disponibilidad; también se puede manejar al mismo tiempo una combinación de los tres tipos de recursos. En el modelo expuesto en el presente documento se trabaja solamente con recursos renovables.

3.6 Tipos de Programación

Para resolver el RCPSP con duración de actividades aleatoria se pueden tener diferentes enfoques de programación , según (Brčić et al., 2012) son: predictiva, reactiva y proactiva, y combinaciones entre ellos.

3.6.1 Programación Predictiva

Ignora la estocasticidad del problema y utiliza estimaciones puntuales, generalmente expectativas o mediana, en lugar de variables aleatorias, logrando ser resuelto de forma determinística, (Brčić et al., 2012).

3.6.2 Programación Reactiva

Toman decisiones de programación durante el tiempo de ejecución del proyecto. Pueden trabajar con el cronograma de referencia, como procedimientos de reparación programados, y sin

el cronograma de referencia como procedimientos completamente en línea (Brčić et al., 2012). En este enfoque Brčić et al. (2012) habla de dos grupos de métodos de programación reactiva:

- Los métodos de programación dinámica, son aquellas en donde el algoritmo no tiene en cuenta una programación de referencia generada con anterioridad, en su lugar, realiza un proceso de decisiones en múltiples etapas, creando dinámicamente una programación usando políticas para seleccionar las actividades en cada momento de decisión.

- Los métodos de reparación de un programa de referencia, en este tipo de programación se hace uso de un programa de referencia ajustándolo a medida que se programan las actividades, el programa de referencia puede ser generada por algún procedimiento predictivo o proactivo, por lo tanto, dependiendo del origen de este, se puede hablar de programaciones *predictivas-reactiva*s o programaciones *proactivas-reactivas*.

3.6.3 Programación Proactiva

Es aquella programación que crea un cronograma de referencia de mayor solidez a resultados inesperados como una duración de la actividad más larga de lo previsto (Brčić et al., 2012).Los métodos para programación proactiva según Brčić et al., (2012) son:

- Los métodos basados en duraciones redundantes (redundancy based methods), buscan dar un tiempo adicional al tiempo de las actividades, sirviendo de soporte para enfrentar las eventualidades que puedan aparecer, evitando así el incumplimiento en la fecha de finalización del proyecto. Este tiempo adicional se puede generar extendiendo la duración original de cada actividad o insertando amortiguadores de tiempo (buffers) en sitios estratégicos de la red del proyecto, para estos métodos es necesario estudiar los riesgos potenciales del proyecto y cuantificar su impacto.

- La estrategia proactiva basada en métodos de programación robusta (robust scheduling methods) esto proporcionan una línea-base para el proyecto, la cual ha sido previamente evaluada mediante alguna medida de robustez.

- La estrategia basada en métodos de programación contingente (contingent scheduling) estas proporcionan más de una línea-base para el proyecto, las cuales han sido creadas con base en las diferentes alternativas de riesgo, creando una línea-base para cada posibilidad, para así contar con planes de acción alternativos durante la ejecución del proyecto.

3.6.4 Programación Reactiva-Proactiva

Este es otro enfoque encontrado en la literatura, en donde de forma proactiva se toman decisiones antes de la ejecución del proyecto, generando por ejemplo un programa guía o de referencia antes de que comience el proyecto, para que posteriormente en la ejecución de este se tomen decisiones restantes y se actúe frente a las variaciones presentadas (Bruni et al., 2015).

3.7 Herramientas para la Programación de Proyectos

Dentro de la programación de proyectos se usan diferentes procedimientos y herramientas que, de forma individual, en conjunto, o como parte de heurísticas, ayudan a programar y solucionar problemas RCPSP y SRCPSP estos son: Esquemas de generación de secuencia, políticas de prioridad y reglas de prioridad.

3.7.1 Esquemas de Generación de Secuencias.

Según (Ballestín, 2007) son procedimientos para construir una programación de una lista de actividades, esta propiedad ha hecho que se utilicen en heurísticas, especialmente cabe mencionar su uso en los algoritmos genéticos para codificar y decodificar cromosomas. Estos esquemas pueden ser de tres tipos:

3.7.1.1 SGS Paralelo. Este secuenciador itera en puntos de decisión denominados PD(t), estos puntos se definen como el instante *t* en el cual se liberan o están disponibles los recursos. En cada PD(*t*) se crea un conjunto de actividades elegibles (AE(*t*)) para ese instante *t*, estas actividades deben ser factibles en precedencias, así, en cada PD(t) se programa una actividad que se encuentre dentro del conjunto AE(*t*), la cual será elegida basado en una representación, por ejemplo, en el orden de prioridad de una lista de actividades. Este proceso se repite hasta que no se puedan programar más actividades en ese instante y se define un nuevo punto de decisión PD(*t*). El secuenciador finaliza cuando el conjunto AE(*t*) está vacío (MorilloTorres, 2017).

3.7.1.2 SGS en Serie. Este secuenciador trabaja con una lista de actividades, tomando una por una de ellas, para programarla tan pronto como sea posible, teniendo en cuenta la relación de precedencia y la restricción de recursos (Ballestín, 2007).

3.7.1.3 SGS en Serie Estocástico. Este es una mezcla de un SGS en paralelo y en serie. Un SGS en serie estocástico trabaja como un SGS en serie, pero agrega una restricción secundaria que obliga al secuenciador a tomar las actividades una por una en el orden de la lista y programarlas tan pronto sea posible, pero sin adelantar las actividades ya programadas (Ballestín, 2007).

3.7.2 Política Π

Según Ballestín (2007) una política es una función, $\Pi: R_n^+ \rightarrow R_n^+$, que mapea muestras de vector de duración de actividad $S(d)\ de\ R_n^+$ de las actividades factibles, definiendo para cada tiempo de decisión $t \geq 0$ las actividades que se inician en *t*. Los tipos de políticas según (Leus et al., 2016) son: basadas en recursos, basadas en actividades, de inicio más temprano y de preprocesador.

3.7.3 Reglas de Prioridad

Una regla de prioridad es un algoritmo que le da a cada actividad *i* un valor v(*i*), el cual durante la programación de un proyecto, la actividad con el valor extremo (es decir, el valor máximo o mínimo) se selecciona para ejecutarse primero, Chen et al. (2018). Algunas de las reglas de prioridad más usadas según Faria et al. (2015) son: LJN (número de trabajo más bajo), SPT (menor tiempo de procesamiento), LPT (mayor tiempo de procesamiento), MIS (sucesores más inmediatos), MTS (la mayoría de los sucesores totales), LNRJ (menor número de trabajos relacionados), GRPW (mayor peso posicional de rango) ,EST (hora de inicio más temprana), EFT (hora de finalización más temprana), LST (última hora de inicio), LFT (tiempo de finalización más lejano), MSLK (holgura mínima), GRWC (mayor contenido de trabajo de recursos), GCRWC (mayor contenido de trabajo de recursos acumulativos).

Estas reglas de prioridad son propias del RCPSP determinístico, sin embargo se pueden aplicar fácilmente a los SRCPSP dando buen rendimiento, tal como lo demostraron (Z. Chen et al., 2018), estos proponen dos reglas especialmente diseñadas para el SRCPSP, la última hora de inicio estadística (SLST) y la última hora de finalización estadística (SLFT), las cuales se calculan en función de la última hora de inicio prevista y la última hora de finalización prevista respectivamente, dicha horas son un valor promedio basado en simulación.

3.8 Métodos de Solución

Para solucionar los problemas de optimización combinatoria de complejidad P y NP de forma eficiente, encontrando óptimos locales (la mejor solución en un área determinada de soluciones) o el óptimo global (la mejor solución en toda el área de soluciones posibles), se usan métodos heurísticos, esto también pueden ser usados para problemas NP-Hard, sin embargo, son los métodos metaheurísticos los que están diseñados especialmente para este nivel de complejidad.

3.8.1 Método Heurístico

Los métodos heurístico resuelven problemas de optimización mediante una aproximación intuitiva, limitándose a proporcionar una buena solución no necesariamente óptima Martí (2001), algunos métodos definidos por este autor son: *de descomposición, inductivos, de reducción, constructivos y de búsqueda local.*

3.8.2 Métodos Metaheurísticos

Los métodos metaheurísticos están diseñados para resolver problemas difíciles de optimización combinatoria, en los que los heurísticos clásicos no son efectivos. Los Metaheurísticos proporcionan un marco general para crear nuevos algoritmos híbridos combinando diferentes conceptos derivados de la inteligencia artificial, la evolución biológica y los mecanismos estadísticos Martí (2001). Hay diferentes formas de clasificar las técnicas metaheurísticas, una de las más populares las divide en metaheurísticas *basadas en trayectoria y basadas en población,* Chicano García (2007).

3.8.2.1 Metaheurísticas Basadas en Trayectoria. Estas manipulan en cada paso un único elemento del espacio de búsqueda, partiendo de una solución y mediante la exploración del vecindario, van actualizando la solución actual, formando una trayectoria, Chicano García (2007), dentro de las metaheuristicas se pueden encontrar los siguientes métodos: *enfriamiento simulado (SA), búsqueda tabú (TS)***,** *el procedimiento de búsqueda miope aleatorizado y adaptativo (GRASP)***,** *búsqueda con vecindario variable (VNS) y búsqueda local iterada (ILS).*

3.8.2.2 Metaheurísticas Basadas en Población. Estas se caracterizan por trabajar con un conjunto de soluciones (denominado población) en cada iteración Chicano García (2007), dentro de las metaheurísticas se pueden encontrar los siguiente métodos: *búsqueda dispersa (SS),*

optimización basada en colonias de hormigas (ACO), *optimización basada en cúmulos de partículas (PSO), algoritmos de estimación de la distribución, algoritmos evolutivos (EA).*

3.8.2.2.1 Algoritmos Evolutivos o Evolutionary Algorithms (EA). Están inspirados en la teoría de la evolución natural, el conjunto de soluciones con que trabaja es denominado población de individuos, estos son sometidos generalmente a 3 procesos principales: selección, reproducción y reemplazo, Chicano García (2007).

3.9 Algoritmo Genético

Dentro de los métodos metaheurísticos en este trabajo se ha escogido el método evolutivo algoritmo genético (AG) para resolver el RCPSP con duración de actividades aleatoria. García Sánchez (2013) informa que esta metaheurística "…fue ideada por Holland en 1975 y está inspirada en los procesos de adaptación de los seres vivos. Los resultados de los patrones de evolución de los seres vivos han sido sobradamente probados con éxito (a lo largo de la evolución de las especies) y constituyen la base de los algoritmos genéticos".

3.9.1 Lenguaje Algoritmo Genético

3.9.1.1 Cromosoma. Son cadenas de caracteres (genes) que contienen la información genética de los individuos, para efectos prácticos se puede tomar como individuos. Los cromosomas son los que se manipulan para que haya evolución y con ello se vaya mejorando la solución.

3.9.1.2 Individuo. Cadena de caracteres que representan una solución factible (o un conjunto de ellas) al problema.

3.9.1.3 Población. Conjunto de individuos.

3.9.1.4 Generación. Individuos generados en una misma fase del algoritmo.

3.9.1.5 Función de Adaptación. Capacidad del individuo para sobrevivir y generar descendencia. Esta función logra darle un valor asociado al individuo llamado aptitud.

3.9.2 Representación de la solución

3.9.2.2 Representación para el RCPSP con Duración de Actividades Aleatoria. En la literatura se encuentran varias formas de codificar el cromosoma, las más usadas según (Liu et al., 2007) son: *representación de lista de actividades*, *representación del valor de prioridad* (representación de clave aleatoria), *representación de la regla de prioridad*, *representación de vector de cambio* y r*epresentación de esquema de programación*.

3.9.2.3 Aptitud del Individuo. Mide que tan buena solución es el individuo para el problema, resultado de evaluar su efectividad mediante la función objetivo del problema, calculando un número al cual se le denomina la aptitud.

3.9.2.4 Tasa de Selección. Porcentaje de individuos a seleccionar de la población anterior.

3.9.3 Operadores

Los operadores son pequeñas funciones que hacen los cambios en los cromosomas y por ende son los responsables de la evolución, por ello son los que le dan forma al AG. Los operadores más usados son los de selección, cruce y mutación.

3.9.3.1 Operador de Selección. Con este operador se elige los individuos de la población actual que va a generar la descendencia de la nueva generación, García Sánchez (2013). Algunos operadores de selección son:

- *Selección proporcional.* Este es el propuesto originalmente por Holland, en donde se calcula y se usa una probabilidad de selección de cada individuo proporcional a su función de adaptación, García Sánchez (2013).

- *Selección determinista.* Con este operador se selecciona los individuos directamente proporcionales a su función de adaptación, (García Sánchez, 2013).

- *Normalización.* Este tipo selección se suele usar cuando no se desea una gran diferencia entre las probabilidades de selección de los individuos, para con esto evitar una convergencia prematura hacia una región individuos con mayores valores de la función de adaptación, para ello se normalizan los valores de la función de adaptación, (García Sánchez, 2013).

- *Clasificación.* Para aplicar esta selección se ordenan los individuos por valor creciente de la función de adaptación asignando probabilidades de selección directamente proporcionales al lugar que ocupa cada individuo, (García Sánchez, 2013).

Selección por torneo. En esta selección se van comparando parejas de individuos (elegidas al azar) para seleccionar la mejor de ellas y se va estableciendo una clasificación de los individuos, (García Sánchez, 2013).

3.9.3.2 Operador de Cruce. Este operador combinar los genes de los cromosomas de diferentes individuos, formando uno nuevo. Los operadores de cruce son:

- *Operador de cruce simple.* En este operador se selecciona un punto de corte dentro del cromosoma generando dos partes, esto en cada uno de los padres, inmediatamente se combinan los genes de las cuatro partes de los cromosomas generando nuevos individuos, (García Sánchez, 2013).

- *Operador con dos puntos de corte.* Este es similar al operador anterior, con la diferencia que se selecciona dos puntos de corte, para luego combinar los genes de los padres generando nuevos individuos, (García Sánchez, 2013).

- *Operador de cruce conforme a una máscara de cruce.* En este operador se hace uso de una máscara(cadenas de uno y ceros) como criterio de selección de los genes a combinar, (García Sánchez, 2013).

3.9.3.3 Operador de Reproducción. Con este operador se selecciona a los individuos de una generación para que pertenezcan también a la siguiente, sin sufrir ningún otro tipo de modificación, (García Sánchez, 2013).

3.9.3.4 Operador de Mutación. En este operador se modifica uno o varios genes del individuo, sin usar como referencia otros individuos, permitiendo introducir posibles variedades totalmente nuevas, (García Sánchez, 2013). El operador se aplica con una probabilidad, llamada probabilidad de mutación, la cual debe garantizar un equilibrio entre explorar nuevas características y profundizar en características ya encontradas.

3.10 Medidas de Robustez

Son medidas usadas para evaluar el desempeño de la solución dada por el algoritmo, estas son usadas en la programación proactiva y proactiva-reactiva. Las medida de robustez son definidas por Brčić et al. (2014) como la cantidad de ajuste que se le debe hacer a la solución una vez sea puesto el proyecto en marcha o simulado; según el mismo autor pueden ser de dos tipos: *robustez de la calidad y robustez de la solución.*

3.10.1 Robustez de la Calidad

Corresponde a maximizar la probabilidad de completar el proyecto a tiempo (Brčić et al., 2014), es decir que la duración total del proyecto sea la esperada, por lo que se busca que, al ejecutarse el proyecto, no tenga mayor variación el makespan obtenido del makespan planeado.

3.10.2 La Robustez de la Solución o Estabilidad de la Programación

Tiene como objetivo hacer que la programación sea estable con respecto a posibles interrupciones, por lo que no cambia mucho durante la ejecución (Brčić et al., 2014), para ello se necesita una mínima variación de los tiempos de inicio durante la ejecución del proyecto respecto a los tiempos planeados.

4. Modelo Matemático

Ballestín (2007) y Tahooneh & Ziarati, (2011) Afirman que el RCPSP con actividades aleatorias, se puede definir con base en el RCPSP determinístico, por ello se plantea primero el RCPSP determinístico.

4.1 Modelo RCPSP Determinístico

El modelo RCPSP determinístico tiene un conjunto de actividades, denotado por un subíndice i y un subíndice espejo j, conformado por $N = \{1,2,3, \dots, \mathrm{n} + 2\}$ donde n representa el número de actividades del proyecto a las cuales se le agrega 2 actividades ficticias (con duración cero) para denotar el inicio y el fin del proyecto, las cuales son respectivamente 1 y n+2. La duración de la actividad i denotada por d_i, está dada por una distribución de probabilidad y pertenece a un vector aleatorio $D = (d_{1,} d_{2,} \dots, d_{n+2})$ además, se tienen una cantidad limitada de diferentes tipos k de recursos, $k = 1,2, \dots, K$ (los cuales son renovables), con una disponibilidad de cada uno de ellos en cualquier momento t denotada como R_k y un requerimiento por parte de la actividad i denotado r_{ik}. Las actividades al ser ejecutadas deben que seguir un orden de precedencia de un conjunto A en N tal que $(i, j) \in A$, si y solo si la actividad j no puede empezar hasta que la actividad i no haya finalizado, la solución se da en forma de una programación base

factible para el proyecto representada por $S = (S_1, S_2, \dots, S_{n+2})$ donde S_i es el inicio de la actividad i, y la finalización del proyecto es S_{n+2} también denotada como C_{max}.

Con los elementos expuesto la formulación matemática es la siguiente:

Conjuntos

i Actividades 1, 2, 3, … , n+2

j Actividades (conjunto espejo) 1, 2, 3, … , n+2

k Recursos 1, 2, 3, … , K

t Momento 1, 2, 3, … , T

Parámetros

d_i Duración de la actividad *i*

r_i Requerimiento de recurso de la actividad *i*

R_k Disponibilidad del recurso *k* en cualquier momento *t*

A_{ij} Matriz de precedencia de las actividades *i j*

Variable

S_i Tiempo de inicio de la actividad *i*

Función objetivo

$$Minimizar\ C_{max} = S_{n+2}$$

Restricciones

$$S_i \leq S_j - d_i \quad \forall (i,j) \in A \qquad [1]$$

$$\sum_{i \in F(s,t)} r_{ik} \leq R_k \quad k \in R,\ t \in [0, S_{n+2}] \quad [2]$$

$$S_i \geq 0 \quad \forall i \in N \qquad [3]$$

La primera restricción [1] obliga a que no se inicie una actividad si antes haber finalizado todas sus actividades predecesoras, garantizando el cumplimiento del orden de precedencias de las

actividades proyecto. La restricción [2] exige que el requerimiento de recurso de las actividades que estén programadas en un momento *t*, no excedan que la disponibilidad de los recursos en ese mismo momento, la restricción [3] garantiza la no negatividad de la variable S_i.

4.2 Modelo RCPSP con Duración de Actividades Aleatoria

Para plantear el modelo del RCPSP con duración de actividades aleatoria con base al determinístico, se hacen 3 cambios, el primero es el parámetro de duración de la actividad d_i, el cual ahora está dada por una distribución y pertenece a un vector aleatorio $D = (d_1, d_2, \dots, d_{n+2})$; el segundo es la introducción de una política como la variable de solución, al igual que Ballestín (2007) y Tahooneh & Ziarati (2011) en la presente investigación se usa como política una lista de actividades λ, la cual representa la solución y se usa para generar la programación base $S = (S_1, S_2, \dots, S_{n+2})$ en un escenario *p*. La tercera es la redefinición de la función objetivo, la cual se adapta a la necesidad de evaluar el comportamiento de la solución en diferentes escenarios y es expresada de la siguiente forma:

$$Minimizar\ E[\lambda, \text{nscen}] = \frac{1}{nscen} \sum_{p \in P} C_{max}(S^{\lambda}(p))$$

Así para cada escenario *p*, que pertenecen al conjunto $P = (p_1, p_2, \dots, nscen)$ se crea la programación $S_i^{\lambda}(p) = (S_1^{\lambda}(p), \dots, S_{n+2}^{\lambda}(d))$ aplicando un SGS en el que se usa como regla de prioridad la lista de actividades (solución) λ, para calcular el tiempo de finalización del proyecto en ese escenario, denotado $C_{max}(S^{\lambda}(p)$ o $S_{n+2}^{\lambda}(p)$, finalmente se calcula el valor esperado, que genera la lista de actividades (solución), como el promedio de todos los tiempos de finalización del proyecto en los diversos escenarios.

Conjuntos

i Actividades 1, 2, 3, … , n+2

j Actividades (conjunto espejo) 1, 2, 3, … , n+2

k Recursos 1, 2, 3, … , K

t Momento 1, 2, 3, … , T

p Escenario 1,2,3, … , nscen

Parámetros

d_i Duración de la actividad *i* (vector aleatorio)

r_i Requerimiento de recurso de la actividad *i*

R_k Disponibilidad del recurso *k* en cualquier momento *t*

A_{ij} Matriz de precedencia de las actividades *i j*

Variable

λ Lista de actividades

$S_i^{\lambda}(p)$ Programa con los tiempos de inicios de las actividades *i* aplicando un SGS con la lista de prioridad λ en el escenario *p.*

Función objetivo

$$Minimizar\ \ E[\lambda\,, \text{nscen}] = \frac{1}{nscen}\sum\nolimits_{p\in P} C_{max}\,(S_i^{\lambda}(p))$$

Restricciones

$$S_i^{\lambda}(p) \le S_j^{\lambda}(p) - d_i \quad \forall\,(i,j)\in A\,,\ \forall\,p \ \ [1]$$

$$\textstyle\sum_{i\in P(s,t)} r_{ik} \le R_k \quad \forall\,k\in R\ ,\ \forall\,t\ \in \left[0, S_{n+2}^{\lambda}(p)\right],\forall\,p\ [2]$$

$$S_i^{\lambda}(p) \ge 0 \quad \forall\,i\in N\,,\forall\,p \quad [3]$$

El modelo busca una lista de actividades λ que al ser utilizada en diferentes escenarios *p* para crear programas $S^{\lambda}(p)$ se logre minimizar el valor esperado del tiempo de finalización $S_{n+2}^{\lambda}(p)$ del proyecto. Al igual que en el RCPSP determinístico, en este modelo en cada escenario se deben cumplir las siguientes restricciones: la primera restricción [1] obliga a que no se inicie

una actividad sin antes haber finalizado todas sus actividades predecesoras, garantizando el cumplimiento del orden de precedencias de las actividades proyecto. La restricción [2] exige que el requerimiento de recurso de las actividades que estén programadas en un momento *t no* excedan que la disponibilidad de los recursos en ese mismo momento, la restricción [3] garantiza la no negatividad de la variable $S_i^{\lambda}(p)$.

5. Algoritmo Propuesto

Para dar solución al modelo planteado en el capítulo anterior, se desarrolló un algoritmo genético soportado en un esquema generador de secuencias en paralelo; a lo largo de esta sección será descrito el AG y su script se puede apreciar en el apéndice B y C, para ello, se tomará como punto de partida el momento en el que se ingresan los parámetros de entrada. Es de destacar, que el algoritmo genético está escrito en lenguaje Python, el cual se caracteriza por ser un lenguaje de programación utilizado en investigación, ser multiplataforma y estar orientado a objetos, además, fue creado para desarrollo bajo licencia de código abierto.

5.1 Parámetros de Entrada

Un parámetro de entrada es definido en programación, como una variable cuyo valor puede ser cambiado a necesidad del programador o usuario en cada ejecución; según lo anterior, para el AG propuesto en el presente proyecto, se establecieron los siguientes parámetros de entrada:

- ✓ Tamaño de la población
- ✓ Número de generaciones
- ✓ Cantidad de puntos de cruce
- ✓ Tasa de selección

- ✓ Instancia

El parámetro “Instancia” hace referencia al conjunto de datos del proyecto a programar, los cuales son:

- ✓ Número de actividades
- ✓ Duración media de la actividad
- ✓ Recursos utilizados por cada actividad
- ✓ Sucesores de cada actividad
- ✓ Disponibilidad de recursos

Después de ingresar los parámetros se procede a crear la población.

5.2 Generación de la Población Inicial

Para el desarrollo de algoritmos evolutivos es esencial la creación de una población, el tamaño de dicha población es fijado por los investigadores según sus lineamientos y/o referencias investigativas; para la presente investigación se genera *n* individuos aplicando un esquema generador de secuencia (SGS) en serie.

En el método propuesto se altera la duración de las actividades mediante una distribución de probabilidad, esta última es asignada conforme a características o comportamientos del proyecto a programar, posteriormente se procede a aplicar un SGS en serie.

5.2.1 SGS en Serie

El secuenciador trabaja buscando dentro del conjunto de actividades elegibles (conformadas por las actividades que no han sido programadas, cuyas precedencias están completas y cumplen con la restricción de recursos) para seleccionar una de ellas y programarla lo más pronto posible, para dicha selección se hace uso del método de la ruleta y dos reglas de prioridad, primero se crea el conjunto de actividades elegibles, luego se genera un número pseudo

aleatorio (producido por el generador aleatorio de Python, usando una distribución uniforme y un periodo grande (Python Software Foundation, 2021)) entre 0 y 1 para asignar la regla con la que se operará en el momento t, si el número generado es menor a 0,5 se operará con la regla de prioridad LFT y si es mayor a 0,5 se operará con la regla de prioridad GRPW, luego se calculan los valores asociados a la regla elegida, para posteriormente, utilizarlos en el método de la ruleta en donde se selecciona la actividad que se va a programar lo más pronto posible; a continuación, se explica el funcionamiento de las dos reglas y del método de la ruleta.

5.2.2 Regla de prioridad LFT

Para aplicar la regla del tiempo de finalización más lejano LFT (por sus siglas en inglés), se calculan los tiempos de finalización más lejanos (LF) de cada una de las actividades elegibles. Una vez calculado dichos tiempos, se utilizan como insumo para el método de la ruleta.

5.2.3 Regla de prioridad GRPW

La regla del mayor peso posicional de rango GRPW (por sus siglas en inglés), se usa calculando un peso posicional a cada una de las actividades elegibles, sumando la duración de dicha tarea y la de todas sus sucesoras, dichos pesos se utilizan como insumo para aplicar el método de la ruleta.

5.2.4 Método de la Ruleta

Para concluir la elección de las actividades elegibles, después de calcular los valores de una de las dos reglas de prioridad, a cada actividad elegible, en el momento de decisión t, se le otorga un porcentaje de probabilidad acorde a su valor de prioridad, por ejemplo, si se tienen disponibles tres actividades elegibles con valores de LF de 36, 25 y 15, se procede a calcular la suma de todos los valores (para el ejemplo, la suma de los 3 valores de LF es de 76) y la proporción que aporta cada una ellas a dicha suma, así se obtienen valores de 36/76, 25/76 y 15/76

respectivamente, por lo cual, las actividades tendrán aproximadamente, 47%, 33% y 20% de probabilidad de ser elegidas, luego se escoge una de estas actividades utilizando un numero aleatorio entre 0 y 100.

Conforme se van programando las actividades, se construye un vector fila que representa el individuo.

5.3 Representación del Individuo

A lo largo del tiempo, se han desarrollado diferentes maneras de representar los individuos, sin embargo, una de las representaciones encontradas con mayor frecuencia en la literatura, es la codificación del cromosoma en forma de lista de actividades, los autores (S Chand et al., 2016) argumentan que es simple, intuitiva y ayuda a mantener la naturaleza discreta del problema, adicionalmente, permite una fácil interpretación. Basándose en ello, se utiliza la mencionada codificación para representar los individuos del algoritmo propuesto; la lista de actividades es definida como un vector fila que contiene todas las actividades *i* del proyecto ordenadas en secuencia, cumpliendo con las restricciones de precedencia; En la **Figura *3*** se puede apreciar un cromosoma que representa un proyecto de 8 actividades reales y dos ficticias, estas últimas con duración cero, siendo así el inicio y el fin del proyecto.

Figura 3

Cromosoma

1	2	3	5	4	7	6	8	9	10

5.4 Cálculo de la Aptitud del Individuo

Para evaluar la aptitud del individuo se busca encontrar un valor que sea coherente con la función objetivo del modelo; para esto, se generan una cantidad especifica (parámetro de entrada) de escenarios en donde se programa utilizando la lista de actividades mediante un SGS paralelo.

5.4.1 SGS Paralelo

El SGS paralelo usa el individuo (lista de actividades) como regla de prioridad para construir *s* programaciones en distintos escenarios manipulando la duración de las actividades con una distribución de probabilidad.

En el proceso se actúa con momentos de decisión *t*, en donde, busca dentro de un conjunto de actividades elegibles para seleccionar una de ellas y programarla en ese mismo momento, otorgando prioridad según la ubicación de las actividades dentro de la lista de actividades evaluada, es decir, en el momento *t* se programa la actividad elegible que primero aparezca en la lista de actividades (al recórrela de principio a fin); una vez sea elegida, se verifica si su requerimiento de recursos no supera los recursos disponibles, si es así, se programa la actividad, de lo contrario, se realiza de nuevo el proceso de selección hasta que se logra programar una, luego se busca programar otra actividad en ese mismo momento, solo si aún queden recursos disponibles, programando tantas actividades del conjunto de elegibles como lo permitan los recursos.

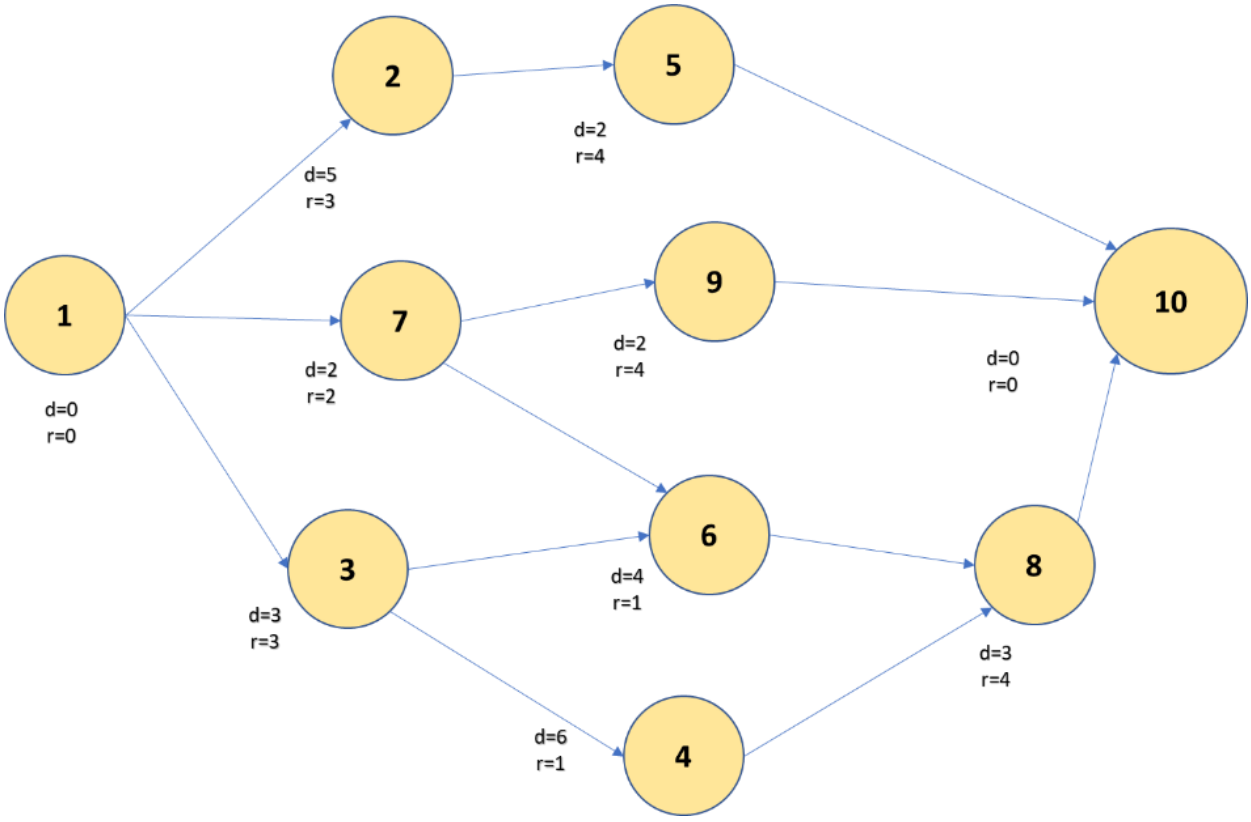
Este SGS paralelo recorre todos los tiempos del proyecto en orden cronológico, buscando aquellos momentos donde exista disponibilidad de recursos, estos se toman como puntos de decisión y allí se programan tantas actividades como sea posible, el proceso se repite hasta programar todas las actividades del proyecto, el tiempo de finalización de la última actividad es tomado como el makespan del escenario.

Tomando como ejemplo el proyecto representado en la **Figura *4***, donde d=duración y r=recursos, con una lista actividades $\lambda = \{ 1, 2, 3,7, 5, 4, 9, 6,5,8,10\}$, aplicando el SGS paralelo, se empieza a programar la actividad 1 en el tiempo t=0, como en ese momento quedan recursos, se puede elegir entre las actividades 2, 3 y 7, en este caso se programa la 2 por ser la que aparece primero al recorrer λ de izquierda a derecha, esta actividad se programa de t=0 a t=5 consumiendo

3 unidades de recurso de las 5 disponibles, ver **Figura *5***; como quedan recursos libres se puede programar otra actividad en el mismo tiempo, para ello se sigue buscando entre las elegibles y se programa la que primero aparezca en la lista de prioridad, para el caso es la actividad 3, sin embargo no se puede programar por exceder los recursos disponibles, así se programa la actividad 7 ya que es la siguiente elegible que aparece en la lista de prioridad, esta actividad va de t=0 a t=2.

Figura 4

Grafica de la red de proyecto

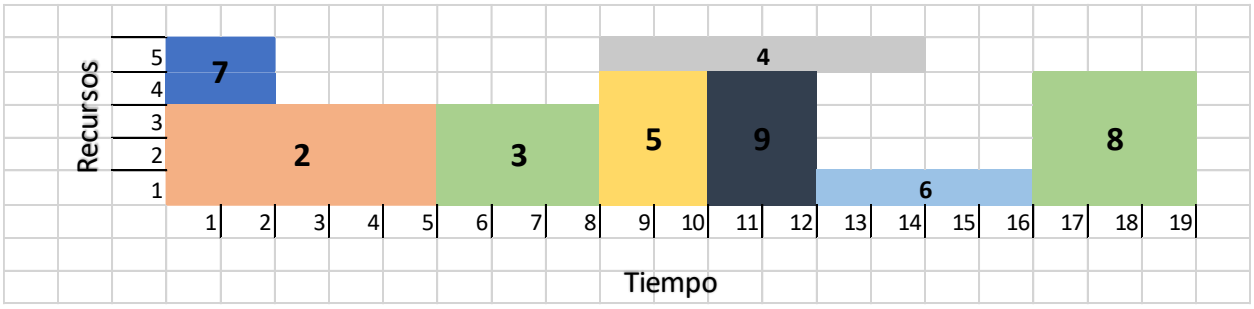


Se busca el momento donde se liberan recursos suficientes para poder programar las actividades, en este caso es el momento t=5, siendo elegibles las actividades 3, 5 y 9, pudiéndose programar (de acuerdo con la prioridad y los recursos) la actividad 3, desde t=5 a t=8. El siguiente momento de decisión es t=8 en donde las elegibles son las actividades 4, 5, 6 y 9, programándose la actividad 5 (de acuerdo con la prioridad y los recursos) de t=8 a t=10 con consumo de 4 unidades

de recurso; desde el tiempo t=5 a t=8 no hay recursos suficiente para programar las actividades elegible, es hasta después de t=8 donde se puede volver a programar , allí la actividad 4 (de acuerdo con la prioridad y los recursos) se programa, desde el momento t=8 a t=14; este proceso se repite hasta programar todas las actividades. La programación obtenida se puede ver gráficamente en la **Figura *5***, en donde el eje vertical son las unidades de recurso y el horizontal son los tiempo o momentos *t*.

Figura 5

Grafica del programa con tiempo vs recurso



Se generan *p* escenarios diferentes utilizando duraciones aleatorias, en cada uno de ellos se aplica un SGS en paralelo para obtener un makespan, luego, es calculado el promedio de todos los makespan generados en los escenarios, y finalmente es asignado dicho valor como makespan esperado a la lista de actividades, para así crear la aptitud del individuo. Este procedimiento está basado en el trabajo hecho por Ballestín (2007).

5.5 Generación de Nuevas Poblaciones

Luego de generar la población inicial, se aplican los procesos de selección, cruce y mutación para crear nuevas generaciones.

5.5.1 Selección

Una vez se tienen los individuos iniciales, se ordenan de menor a mayor aptitud, posteriormente, se eligen una cantidad determinada de los mejores de ellos (según la tasa de

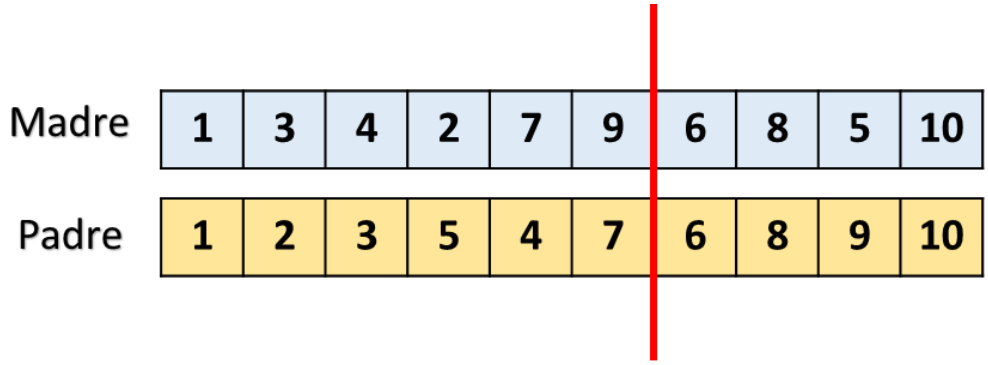
selección), conforme a ello, los individuos que falten para completar el tamaño población, son creados como la primera población, este proceso de introducción de nuevos individuos se hace para asegurar la diversidad de la población; una vez seleccionados y obtenidos los individuos, posteriormente se procede a aplicar el operador de cruce.

5.5.2 Cruce

Este procedimiento se realiza cruzando los mejores padres con toda la población. El operador consta de elegir aleatoriamente la ubicación de un determinado número de puntos en los cromosomas padres para poderlos subdividir. En el algoritmo propuesto se puede elegir la cantidad de puntos de cruces que se desean; por ejemplo, para aplicar un punto de cruce a los cromosomas de la **Figura *6***, en primer lugar, se genera la ubicación del punto aleatoriamente (en este caso se estableció el punto entre las posiciones 6 y 7).

Figura 6

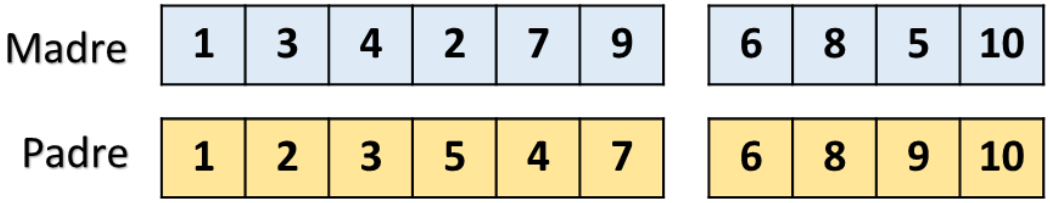
Ubicación de 1 punto de cruce



Una vez se tiene el punto de cruce, se hace un corte dividiéndose cada cromosoma en dos partes o secciones, la primera de ellas tendrá los genes antes del punto de corte y la segunda los genes después del corte, ver **Figura *7***.

Figura 7.

Cromosomas subdivididos



Luego, es seleccionada la primera sección de la madre con la segunda sección del padre para generar el hijo 1, y la primera sección del padre con la segunda sección de la madre, para generar el hijo 2; este procedimiento se puede apreciar en la **Figura *8*** y **Figura *9***.

Figura 8

Procedimiento de creación de los hijos

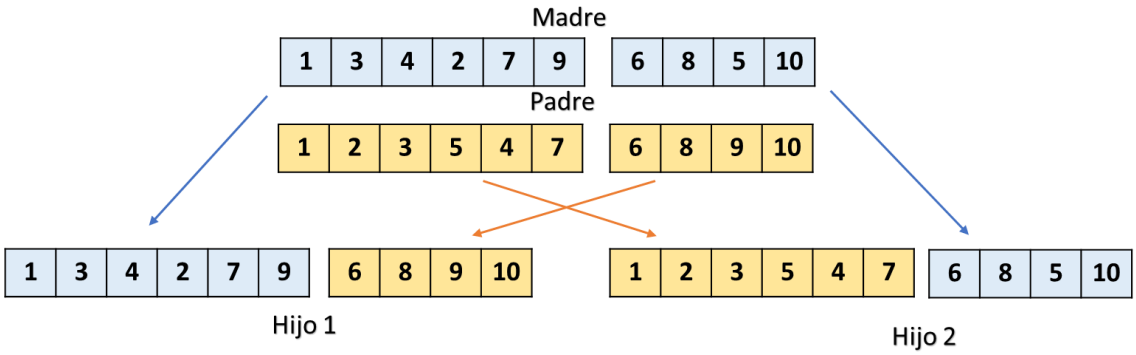
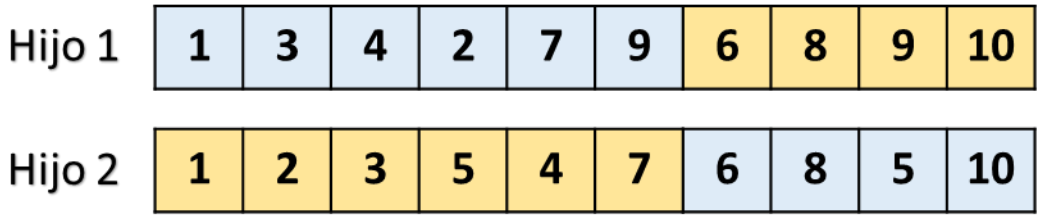


Figura 9

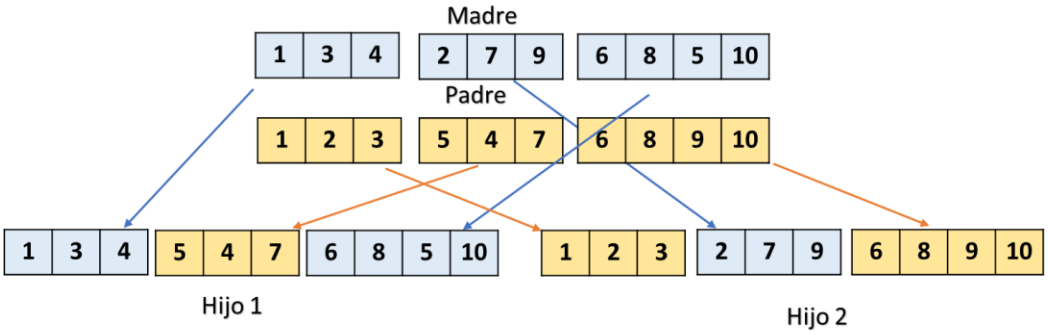
Cromosomas hijos



Para el caso de un cruce de 2 puntos, el cruce se hace dividiendo cada cromosoma en tres secciones; a fin de formar el primer individuo, se usa la primera sección de la madre, la central del padre y la última de la madre; para el segundo individuo, se usa la primera sección del padre, la central de la madre, y la última del padre, como se muestra en la **Figura *10***.

Figura 10

Conformación de hijos con un cruce de 2 puntos



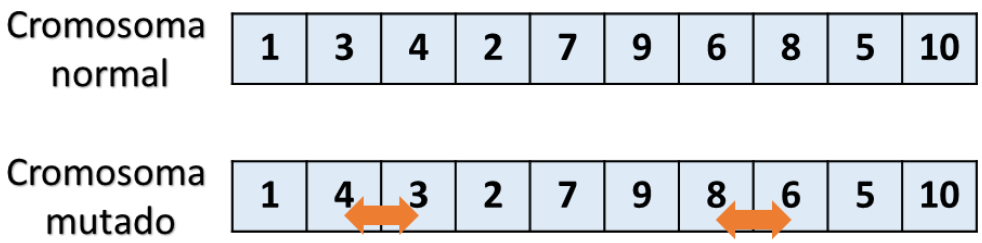
Luego de aplicar el operador de cruce se procede a realizar la mutación de los individuos.

5.5.3 Mutación

Para aportar diversidad de individuos, se aplica el operador de mutación, por ello se clonan los individuos, en donde a cada uno de los clones se les muta una o varias parejas de genes consecutivos, de acuerdo con una probabilidad de mutación del gen; la mutación de un gen se hace intercambiando la posición de los genes, ver **Figura *11***.

Figura 11

Procedimiento de mutación



Luego de aplicar estos operadores, se evalúa que el individuo no contenga genes repetidos y cumpla con las restricciones de precedencia, en caso de presentarse una de estas situaciones el cromosoma se descarta; finalmente calcula la aptitud del individuo la forma antes descrita (SGS en paralelo). Para generar la siguiente población se hace el mismo proceso de selección, cruce y

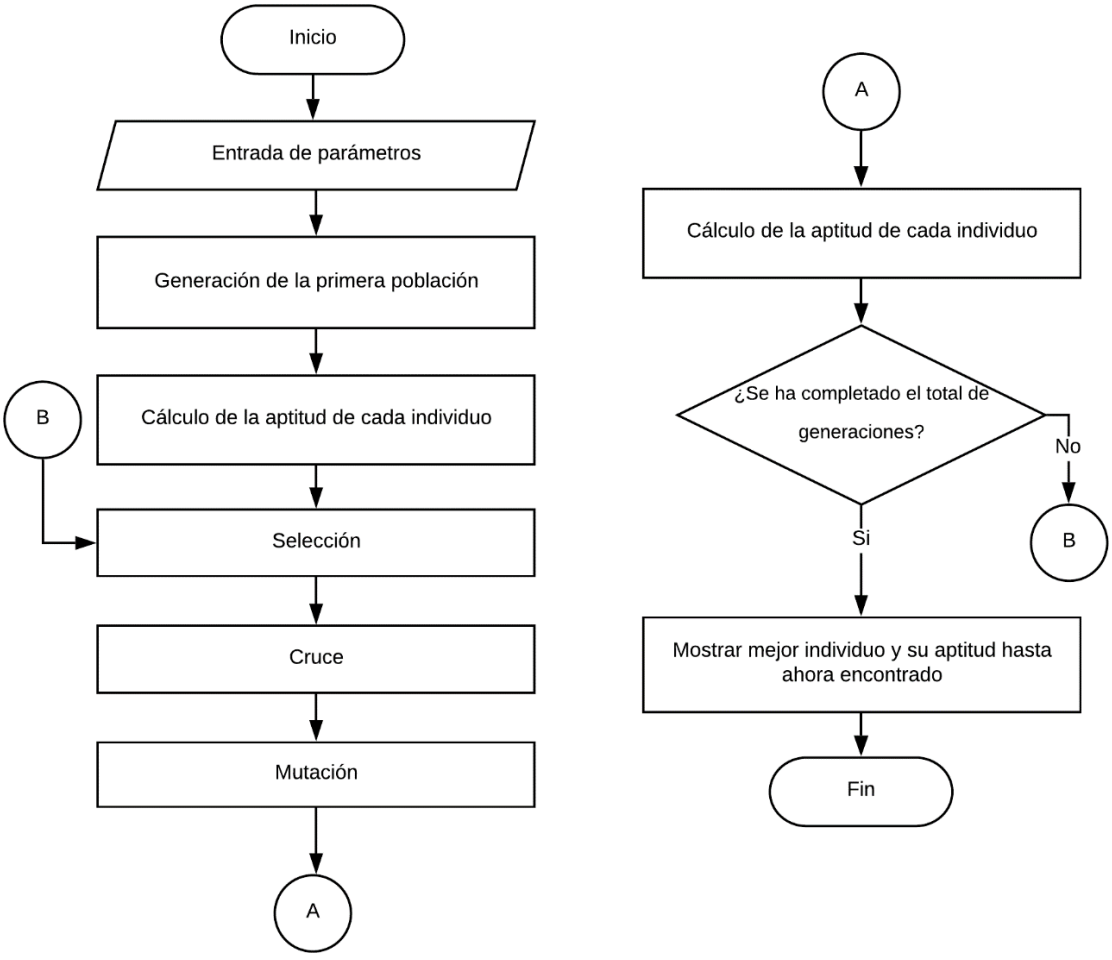
mutación, repitiéndose hasta completar la última generación (parámetro de entrada), finalmente el AG arroja el mejor individuo y su aptitud.

5.6 Diagrama General del Algoritmo

En la **Figura *12*** se presenta un diagrama general del AG propuesto.

Figura 12

Diagrama general del Algoritmo Genético



6. Comparación del Algoritmo

El algoritmo propuesto se comparó con un procedimiento de optimización basado en el método de duraciones redundantes (DR), propuesto en el artículo titulado *An optimization model to solve the resource constrained project scheduling problem RCPSP in new product development projects* (Ortíz Pimiento & Diaz Serna, 2020), dicho procedimiento de solución está escrito en lenguaje de programación GAMS. Para contrastar los resultados se usó la librería PSPLIB, de donde se seleccionaron aleatoriamente 10 instancias de prueba j30, con tamaño de 30 actividades cada una, con una duración esperada de cada actividad y un consumo de recursos determinado, además, de 2 actividades ficticias con duración y consumo de recursos nulo; en total se dispone de 4 tipos de recursos renovables en cada proyecto, con una disponibilidad fija en cada uno de ellos; también se seleccionaron aleatoriamente 10 instancias j60, con tamaño de 60 actividades, y las mismas características de las instancias j30.

Para cada una de las 20 instancias se aplicaron los 2 procedimientos de optimización, obteniendo así 2 soluciones diferentes para cada una de ellas, estas soluciones se utilizan como políticas en la programación reactiva usando un SGS en paralelo; es importante aclarar que la solución obtenida por el método de Ortíz Pimiento & Diaz Serna (2020) , está dada en términos de una línea base (vector de tiempos de inicios de cada actividad del proyecto), mientras que, el AG desarrollado en el presente trabajo, presenta la solución en forma de lista de actividades, por lo tanto, fue necesario usarla como política aplicando un SGS en serie para obtener una línea base.

6.1 Aplicación de Riesgos

Para la comparación de los métodos fue necesario introducir el concepto de riesgo, por lo tanto, se aplican dos riesgos a 10 actividades para las instancias j30 y a 20 actividades para las instancias j60, dichas actividades fueron seleccionadas previamente de manera aleatoria, ver **Tabla 2** y **Tabla 3**; luego, se modifica la duración esperada (para este caso, denotada como D_{SR}),

adicionando a la magnitud de su duración un valor de 50% si ocurre el riesgo 1, 60% para el riesgo 2 o 110% si ambos riesgos afectan a la actividad, la duración adicional es denotada como DR. Por ejemplo, para una actividad con duración esperada de 8 unidades (donde se aplica el riesgo), su duración esperada cambia a ser de 12,13 o 17 unidades, según el o los riesgos que se le apliquen. Los riesgos se aplican como parámetro de entrada, además, tienen una probabilidad de ocurrencia de 50% para el riesgo 1 y del 70% para el riesgo 2, por ende, si el riesgo afecta a la actividad, la variable P toma valor igual a 1, de lo contrario, es igual a cero. La duración con riesgo de la actividad, denotada como D_{CR} se obtiene al aplicar la siguiente ecuación:

$$D_{CR} = D_{SR} + (P_1 * DR_1) + (P_2 * DR_2)$$

Tabla 2.

Riesgos instancias j30

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

Tabla 3.

Riesgos instancias j60

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

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

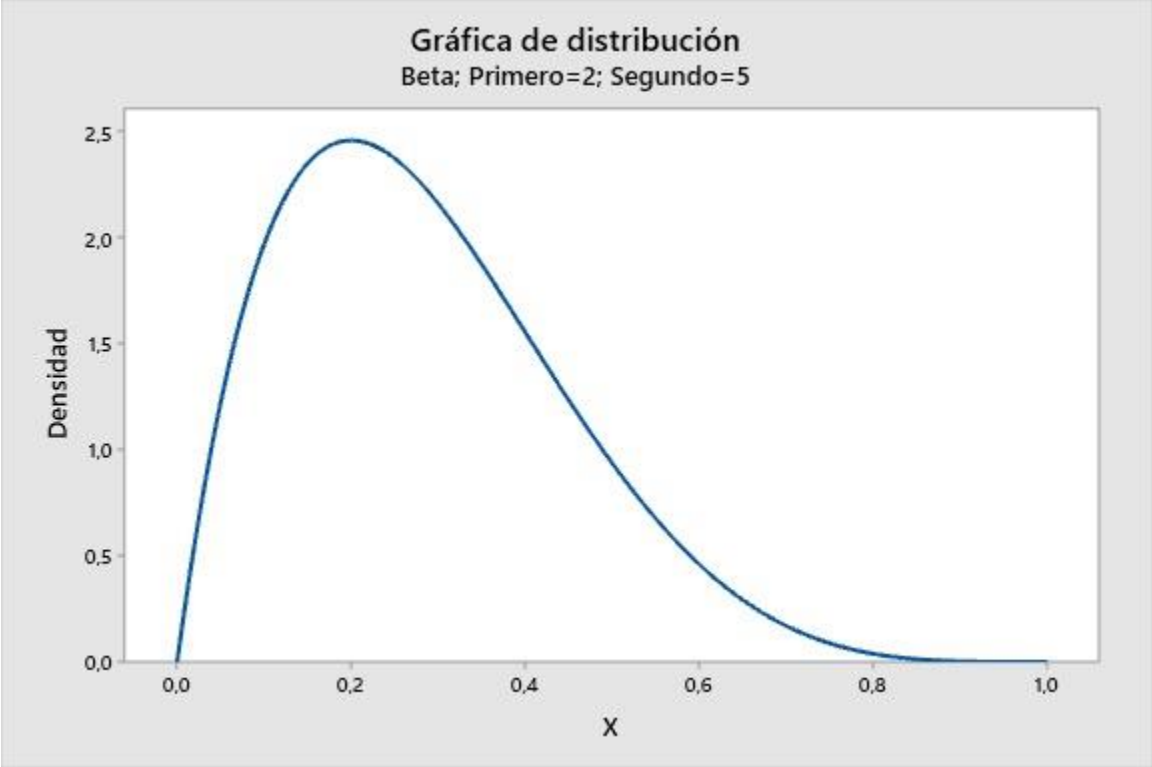
Es importante mencionar que el anterior procedimiento, se aplicó al iniciar el algoritmo y posteriormente, en la programación reactiva.

6.2 Distribución de Probabilidad

Soportándose en la revisión de literatura realizada en la presente investigación, la distribución de probabilidad usada para dar variabilidad a la duración de las actividades, tanto en la programación proactiva como reactiva, fue una distribución beta con rango [a= $\frac{11}{20}d_i$, b= $\frac{23}{8}d_i$] y parámetros de forma *p=2* y *q=5,* satisfaciendo la condición P>1 y Q>1 para asegurar que la distribución de probabilidad sea unimodal; los parámetros anteriormente mencionados al ser P<Q hacen que la distribución esté sesgada a la derecha, debido a que es más realista para pronosticar la duración de la actividad. La varianza para los parámetros usados es de $(\frac{b-a}{6})^2 = \frac{961}{6400}{d_i}^2$ y la localización del valor máximo con respecto a los extremos es igual a $\frac{p*b+q*a}{p+q} = \frac{17}{14}d_i$. Se puede notar que la distribución tiene desviación relativamente pequeña, esto se debe a que las distribuciones PERT-Beta suelen producir coeficientes de variación bajos (Z. Chen et al., 2018).

Figura 13

Grafica de la distribución beta con parámetros p=2 y q=5



6.3 Parámetros del Algoritmo Genético

Según la revisión de literatura realizada y las características del algoritmo propuesto, los parámetros usados en el análisis de las soluciones del este fueron los siguientes:

- ✓ Tamaño de la población: 40
- ✓ Número de generaciones: 100
- ✓ Cantidad de puntos de cruce: 2
- ✓ Tasa de selección: 60
- ✓ Cantidad de escenarios para cálculo de la aptitud: 50
- ✓ Parámetros de la distribución beta: p=2 , q=5 a= $\frac{11}{20}d_i$, b= $\frac{23}{8}d_i$
- ✓ Probabilidad de mutación del gen: 10%

6.4 Procedimiento de Optimización Basado en el Método de Duraciones Redundantes

El procedimiento propuesto por Ortíz Pimiento & Diaz Serna (2020), consiste en tres etapas: identificación de riesgos, estimación de la duración de las actividades y generación de la línea base del proyecto.

6.4.1 Identificación del Riesgo

En esta etapa se puede trabajar con 2 tipos de riesgos, los cuales afectan la duración del proyecto; el primer riesgo, conocido como riesgo externo, se refiere a eventos que interfieren con el desarrollo normal de actividades, como fallas en los equipos de trabajo y accidentes de trabajo, entre otros. El segundo riesgo, llamado riesgo interno, se refiere al margen de error causado al estimar la duración de la actividad en ausencia de riesgos externos, es decir que, este riesgo no está relacionado con eventos externos y no causa interrupción en las actividades. Ortíz Pimiento & Diaz Serna (2020), trabajaron con riesgos externos, para ello, identificaron los riesgos y los asociaron a cada actividad del proyecto, indicando su impacto y probabilidad de ocurrencia. La probabilidad de ocurrencia de cada riesgo se puede determinar en función de la experiencia y el juicio de expertos; si ocurre el riesgo externo, el impacto que este genere, implica realizar una tarea extra, cuya duración se puede representar mediante una distribución de probabilidad, para el caso de Ortíz Pimiento & Diaz Serna (2020), los riesgos externos se asumieron como independientes y la probabilidad de ocurrencia de riesgos externos del proyecto se consideró constante en el horizonte de planificación.

6.4.2 Estimación de la Duración de las Actividades

En esta etapa, se puede utilizar cualquier método basado en redundancia para calcular la duración de las actividades, en el caso de Ortíz Pimiento & Diaz Serna (2020), demostraron un mejor rendimiento de la solución generada con el método basado en el valor esperado del impacto

del riesgo; Zafra-Cabeza et al. (2008), utilizaron este método analizando el impacto de los riesgos estrategias de mitigación en la programación del proyecto y calculando la duración de cada actividad en función del impacto negativo de riesgos externos y el impacto positivo de la mitigación estrategias. Así, se calcula el impacto individual de un riesgo sobre una actividad i multiplicando la probabilidad de ocurrencia del riesgo k (Pr_{ki}) y la duración esperada de la tarea extra asociada a ese riesgo (h_{ki}), posteriormente para estimar la duración de la actividad i (d_i) como la duración total esperada, se suma el valor esperado de la duración básica de la actividad (b_i) y el valor esperado del impacto total de los riesgos sobre la actividad, dicho procedimiento se puede apreciar en la siguiente ecuación:

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

Con el método descrito, Ortíz Pimiento & Diaz Serna (2020) calculan las duraciones de las actividades de cada proyecto, agregando tiempo extra a la duración original para enfrentar eventualidades que puedan surgir durante la ejecución del proyecto.

6.4.3 Generación de la Línea Base del Proyecto

Una vez que calculadas las duraciones de las actividades d_i, la tercera etapa del procedimiento de Ortíz Pimiento & Diaz Serna (2020), consiste en generar una línea base resolviendo el siguiente modelo matemático con programación lineal entera:

Conjuntos

i Actividades 1, 2, 3, … , n+2

j Actividades (conjunto espejo) 1, 2, 3, … , n+2

k Recursos 1, 2, 3, … , K

t Momento 1, 2, 3, … , T

Parámetros

d_i Duración estimada de la actividad *i*

P_{ij} Parámetro binario que indica la relación de precedencia entre actividades, siendo igual a 1 si la actividad *i* precede la actividad *j*, e igual a cero en caso contrario

Rw_{kt} Disponibilidad del recurso *k* en el momento *t*

r_{ik} Consumo de recurso *k* por la actividad *i*

M Número muy grande

Variable

S_i Tiempo de inicio planeado de la actividad *i*

A_{it} Variable binaria que indica si la actividad *i* se ejecuta en el periodo *t*, siendo igual a 1 si la actividad *i* se encuentra activa en el periodo *t*, e igual a cero en caso contrario

Función objetivo

$$Minimizar\ S_{n+2}$$

Restricciones

Sujeto a:

$$S_j \geq P_{ij} * [S_i + d_i] \quad \forall\, i,\ \forall\, j \quad [1]$$

$$S_i + d_i - 1 \geq t * A_{it} \quad \forall\, i,\ \forall\, j \quad [2]$$

$$S_i \leq t + (1 - A_{it}) * M \quad \forall\, i,\ \forall j \quad [3]$$

$$\sum_{t=1}^{T} A_{it} = d_i \ \forall\, i \quad [4]$$

$$\sum_{j=1}^{n} (r_{ik} * A_{it}) \leq Rw_{kt} \ \forall\, k, \forall\, t \quad [5]$$

$$S_i \geq 0 \; entero \quad \forall \, i \; [6]$$

$$A_{it} \in \{0,1\} \quad \forall \, i, \; \forall \, t \qquad [7]$$

La función objetivo del modelo busca minimizar el Makespan o duración total del proyecto (S_{n+2} indica el tiempo de inicio de la actividad ficticia n+2). La ecuación [1] asegura que las actividades inician solo cuando aquellas que la preceden han finalizado. Las ecuaciones [2] y [3] aseguran que A_{it} sea igual a 1 cuando la actividad *i* se encuentra activa en el momento *t*, en caso contrario, A_{it} es igual a cero. El aporte de cada ecuación es presentado a continuación: la ecuación [2] permite que A_{it} sea 1 solo si la actividad *i* no ha finalizado en el momento *t*, la ecuación [3] permite que A_{it} sea 1 solo si la actividad *i* inició después del momento *t*, y la ecuación [4] asegura que A_{it} sea 1 para el intervalo $\{ S_i \, , \; S_i + \; d_i \}$.

De otra parte, la ecuación [5] permite programar simultáneamente varias actividades siempre y cuando no se superen los recursos disponibles. Finalmente, las ecuaciones [6] y [7] indican las condiciones de no negatividad y el tipo de variable requerida.

El modelo tuvo en cuenta las siguientes consideraciones:

- Las duraciones obtenidas (d_i) se redondean al valor entero más cercano. Esta estrategia de redondeo permite obtener una cifra más cercana al valor calculado previamente por el método de duraciones redundantes.

- Las actividades a insertar se incorporan al programa lineal entero sin tener en cuenta su probabilidad de ocurrencia.

- El modelo propuesto solo tiene en cuenta los recursos renovables del proyecto.

El anterior modelo de programación lineal entera fue programado en GAMS por Ortíz Pimiento & Diaz Serna (2020), usando el solver MIP de CPLEX, el cual utiliza un potente algoritmo de ramificación y corte. Este algoritmo es una versión avanzada del algoritmo de

ramificación y acotamiento, ya que ejecuta el proceso de ramificación y emplea planos de corte para ajustar las relajaciones de la programación lineal.

6.5 Evaluación de Desempeño Mediante Programación Reactiva

Las soluciones generadas por cada método fueron puestas a prueba en una programación reactiva, simulando 10000 escenarios para cada uno de los proyectos, dicha simulación fue realizada en condigo Python y su script se puede apreciar en los apéndices D, E, F y G, dándole una aleatoriedad a las duraciones de las actividades por medio de la misma distribución beta usada en el AG. En cada uno de estos escenarios se evaluó el desempeño de cada solución, para ello, se usó un SGS en paralelo que utiliza como regla de prioridad una línea base para programar las actividades del proyecto, obteniendo programaciones con los tiempos de inicios de cada actividad del proyecto, con estos se calculan los siguientes criterios de evaluación: makespan esperado, índice de robustez de la calidad e índice de robustez de la solución.

En la práctica, el orden en que sean tomados los criterios para evaluar una programación base depende de los objetivos planteados y las necesidades de los interesados. A fin de analizar cuál es el mejor método de los evaluados, en el presente trabajo, se utilizó el siguiente orden: makespan esperado e índices de robustez de la calidad y de la solución.

El primer criterio que se tuvo en cuenta fue el makespan esperado, esto debido a la importancia que tiene realizar la totalidad del proyecto en el menor tiempo posible, a pesar de que surjan o no, eventos inciertos, por lo que se busca obtener el menor makespan esperado posible. El segundo criterio que se tuvo en cuenta fue el índice de robustez de la calidad, el cual se asocia al nivel de variabilidad de la duración total del proyecto con respecto a la duración esperada, siendo un menor valor lo que se desea. Como tercer criterio se tuvo el índice de robustez de la solución, el cual mide la estabilidad de la programación con respecto a posibles interrupciones para cada

actividad, por ello, se necesita una mínima variación de los tiempos de inicio durante la ejecución del proyecto respecto a los tiempos planeados.

6.5.1 Medidas de Robustez

Se usaron dos tipos de medidas de robustez: de la solución (ecuación [A]) y de la calidad (ecuación [B]), para calcular estas se utilizaron los siguientes elementos: un conjunto de las actividades del proyecto $N = \{1,2,3, \dots, \mathrm{n}+2\}$ con i como subíndice, un conjunto de escenario $\mathrm{E}= \{1,2,3, \dots, \mathrm{nscen}\}$ con e como subíndice, el tiempo de inicio esperado de la actividad i representado con s_i , el makespan esperado del proyecto (s_{n+2}), el tiempo de inicio de la actividad i en el escenario e denotado como s_{ie} y el makespan del proyecto en el escenario e $\left(s_{n+2,e}\right)$.

$$\sum_{i=1}^{n+2}\left(\sum_{e=1}^{nscen}\frac{|s_i - s_{ie}|}{\mathrm{nscen}}\right) \quad [A]$$

$$\sum_{e=1}^{nscen}\frac{\left|s_{n+2} - s_{n+2,e}\right|}{\mathrm{nscen}} \quad \forall\, e \in E \;\; [B]$$

7. Resultados

7.1 Resultados de los Métodos

Las **Tabla *44*** y **Tabla 5** muestran los resultados obtenidos por los métodos comparados de cada una de las 20 instancias, ver apéndice H, I, J y K.

Tabla 4

Resultados instancias j30 de los métodos comparados

	AG	**Duraciones redundantes**
Instancia	Aptitud	Makespan
j301_2	81	62
j3014_3	97	72

j3023_9	97	70
j3044_9	95	86
j3020_7	68	53
j301_10	81	64
j3019_5	84	71
j306_3	90	82
j3037_8	124	98
j3047_6	98	75

Tabla 5

Resultados instancias j60 de los métodos comparados

	AG	**Duraciones redundantes**
Instancia	Aptitud	Makespan
j6014_2	116	95
j607_7	125	98
j601_2	111	92
j6022_10	121	95
j6017_6	120	86
j603_5	128	104
j604_6	116	86
j601_6	106	85
j6011_8	109	85
j6012_10	125	99

Se observa que la aptitud del mejor individuo para cada instancia (la cual es un promedio obtenido de 50 escenarios) es mayor al makespan calculado por el procedimiento de optimización basado en el método de duraciones redundantes, sin embargo, no es correcto comparar estos resultados directamente debido a la forma y la naturaleza en la que cada procedimiento encuentra su solución, por ello, se representan todas las soluciones en forma de vectores de tiempos de inicios (línea base) y se recurre a la programación reactiva en 10000 escenarios, en donde de manera estocástica se obtiene un makespan esperado para cada uno de las instancias, con sus respectivas medidas de robustez.

7.1.1 Resultado de la Simulación

La simulación de la programación reactiva se llevó a cabo en el lenguaje Python; los resultados obtenidos para cada una de las instancias j30 y j60 seleccionadas se muestran en las

Tabla 6, Tabla 7, Tabla 8 y **Tabla 9**.

Tabla 6

Resultados simulación instancias j30 con AG

		Cantidad de Escenarios					
		1000		5000		10000	
		Makespan		Makespan		Makespan	
	Instancia	Promedio	Desviación	Promedio	Desviación	Promedio	Desviación
AG	j301_2	93,935	11,591	94,004	11,857	93,999	11,756
	j3014_3	105,888	9,680	105,688	9,713	105,531	9,685
	j3023_9	103,793	10,430	104,083	10,742	104,211	10,950
	j3044_9	102,134	10,860	101,756	11,177	101,903	11,211
	j3020_7	75,578	10,290	75,233	10,071	75,095	10,097
	j301_10	90,920	12,678	91,259	12,841	91,279	12,759
	j3019_5	92,419	11,546	92,165	11,538	92,105	11,532
	j306_3	98,325	11,348	98,551	11,273	98,619	11,324
	j3037_8	141,663	16,576	141,724	16,773	141,944	16,597
	j3047_6	99,365	10,638	99,451	10,868	99,4821	10,989

Tabla 7

Resultados simulación instancias j30 con DR

		1000		5000		10000	
		Makespan		Makespan		Makespan	
		Promedio	Desviación	Promedio	Desviación	Promedio	Desviación
DURACIONES REDUNDANTES	j301_2	88,289	11,083	87,943	11,107	88,185	11,135
	j3014_3	98,621	8,778	98,549	8,998	98,493	9,049
	j3023_9	102,366	11,278	101,997	11,140	102,099	11,193
	j3044_9	102,613	11,408	102,386	11,128	102,484	11,092
	j3020_7	74,870	9,880	74,865	9,980	74,944	10,035
	j301_10	87,900	11,219	87,862	11,556	88,014	11,641
	j3019_5	91,491	12,125	91,593	11,718	91,548	11,617
	j306_3	102,272	11,720	102,153	11,564	101,983	11,577
	j3037_8	128,246	12,587	128,744	12,876	128,869	12,877

	j3047_6	102,081	10,693	102,381	10,543	102,360	10,387

Tabla 8

Resultados simulación instancias j60 con AG

		Cantidad de Escenarios					
		1000		5000		10000	
		Makespan		Makespan		Makespan	
	Instancia	Promedio	Desviación	Promedio	Desviación	Promedio	Desviación
AG	j6014_2	113,798	8,915	113,689	8,733	113,877	8,839
	j607_7	152,820	15,646	153,729	15,574	153,781	15,542
	j601_2	133,107	13,521	133,422	13,366	133,496	13,384
	j6022_10	121,671	10,368	121,819	10,204	121,959	10,291
	j6017_6	127,674	12,903	127,111	12,672	127,143	12,590
	j603_5	133,491	11,859	133,573	12,058	133,553	12,021
	j604_6	107,472	10,753	107,997	10,772	107,862	10,850
	j601_6	100,882	8,645	100,787	8,529	100,863	8,573
	j6011_8	125,872	12,112	126,435	11,837	126,418	11,817
	j6012_10	139,145	14,110	139,144	14,517	139,037	14,500

Tabla 9

Resultados simulación instancias j60 con DR

		Cantidad de Escenarios					
		1000		5000		10000	
		Makespan		Makespan		Makespan	
		Promedio	Desviación	Promedio	Desviación	Promedio	Desviación
DURACIONES REDUNDANTES	j6014_2	123,499	13,232	123,673	13,150	123,905	13,226
	j607_7	139,802	14,086	140,056	14,049	139,962	13,880
	j601_2	129,877	12,709	129,812	12,308	129,813	12,181
	j6022_10	128,669	10,816	128,000	10,349	128,178	10,460
	j6017_6	131,630	17,528	132,435	17,849	132,556	17,767
	j603_5	137,177	13,118	136,788	130,130	136,809	12,855
	j604_6	119,585	13,844	119,389	13,369	119,262	13,257
	j601_6	112,875	10,824	112,718	10,764	112,924	10,671
	j6011_8	115,534	9,704	115,638	10,159	115,600	10,239
	j6012_10	133,466	13,9940	133,545	14,230	133,543	14,219

Se observa que la diferencia del promedio y de la desviación entre los resultados de cada instancia para 1000, 5000 y 10000 escenarios no es amplia, debido a esto, se realizó un diseño de

experimento de un solo factor con efectos fijos, ver apéndice L y M, donde se seleccionaron 3 de las 10 instancias j30 (j306_3, j3037_8, j3044_9) y 3 de las 10 instancias j60 (j604_6, j6014_2, j6022_10), teniendo en cuenta su nivel de complejidad ver **Tabla 10** y **Tabla 11**, según lo expuesto en (Shelvin Chand et al., 2017) y (Kolisch & Sprecher, 1997).

Tabla 10

Nivel de complejidad instancias j30

Instancia	**NC**	**RF_R**	**RS_R**	**Clasificación**
j3044_9	2,1	0,75	1	Bajo
j306_3	1,5	0,5	0,5	Medio
j3037_8	2,1	0,5	0,2	Alto

Tabla 11

Nivel de complejidad instancias j60

Instancia	**NC**	**RF_R**	**RS_R**	**Clasificación**
j604_6	1,5	0,25	1	Bajo
j6022_10	1,8	0,5	0,5	Medio
j6014_2	1,5	1	0,5	Alto

Luego de ello se establecieron 3 tratamientos, producto de variar la cantidad de escenarios (1000, 5000 y 10000), efectuando 3 réplicas para cada uno, lo que permitió establecer la hipótesis nula H_0, donde se plantea que no hay diferencia significativa entre las medias de los tratamientos, y en consecuencia una hipótesis alternativa donde se plantea que para al menos un par de tratamientos hay una diferencia significativa;

$$H_0: \mu_1 = \mu_2 = \mu_3$$

$H_1: \mu_i \neq \mu_j$ para al menos un par (i,j)

En la **Tabla 12** y **Tabla 13** se muestran los resultados obtenidos para la instancia j306_3 tanto para el AG como para el procedimiento de optimización basado en el método de duraciones redundantes respectivamente.

Tabla 12

Resultados experimento de escenarios para instancia j306_3 con AG

j306_3 con AG						
Tratamiento	**Makespan promedio**			**Desviación**		
1000	98,679	98,454	98,325	11,563	11,361	11,348
5000	98,759	98,676	98,551	11,392	11,462	11,273
10000	98,599	98,648	98,648	11,328	11,404	11,324

Tabla 13

Resultados experimento de escenarios para instancia j306_3 con DR

j306_3 Duraciones Redundantes						
Tratamiento	**Makespan promedio**			**Desviación**		
1000	102,600	104,800	102,272	7,130	12,584	11,720
5000	101,930	101,925	102,153	11,768	11,633	11,564
10000	101,908	101,990	101,983	11,733	11,716	11,577

Con los datos de las **Tabla 13** y **Tabla 14** se realizó un análisis de varianza ANOVA dando como resultado un valor $F_0 = 1{,}82$ para el AG y $F_0 = 0{,}47$ para el procedimiento de optimización basado en el método de duraciones redundantes, siendo ambos valores menores al $F_{teorico} = 5{,}14$, por lo tanto, no hay evidencia suficiente para rechazar la hipótesis nula en ambos métodos. Este mismo resultado se obtuvo con las demás instancias j30 y j60 (Apéndice L y M).

En la **Tabla 14** y **Tabla 15** se puede observar los resultados de los índices de robustez y el valor esperado, obtenidos en la programación reactiva con la línea base de cada método en cada una de las instancias j30 y j60 seleccionadas.

Tabla 14

Resultados del makespan esperado e índices de robustez para instancias j30

	AG			Duraciones Redundantes		
Instancia	**Makespan Esperado**	**Robustez**		**Makespan Esperado**	**Robustez**	
		De la calidad	**De la solución**		**De la calidad**	**De la solución**
j301_2	93,999	22,105	471,265	88,185	26,193	315,351
j3014_3	105,531	23,545	353,453	98,493	26,494	332,283
j3023_9	104,211	21,306	424,974	102,099	32,100	327,796
j3044_9	101,903	29,912	394,446	102,484	16,746	284,158
j3020_7	75,095	24,101	372,712	74,944	21,960	276,623
j301_10	91,279	21,485	462,978	88,014	24,052	300,030
j3019_5	92,105	23,167	388,885	91,548	20,694	280,410
j306_3	98,619	26,636	403,904	101,983	20,200	335,413
j3037_8	141,944	17,374	518,183	128,869	30,896	479,193
j3047_6	99,482	23,507	272,657	102,360	27,362	264,527

Nota: los valores son los encontrados simulando 10000 escenarios

Tabla 15

Resultados del makespan esperado e índices de robustez para instancias j60

	AG			Duraciones Redundantes		
Instancia	**Makespan Esperado**	**Robustez**		**Makespan Esperado**	**Robustez**	
		De la calidad	**De la solución**		**De la calidad**	**De la solución**
j6014_2	113,877	71,877	653,208	123,905	28,920	718,08
j607_7	153,781	99,781	1.082,918	139,962	41,964	799,588
j601_2	133,496	73,496	1244,918	129,813	37,813	1.050,451
j6022_10	128,178	66,178	782,782	126,922	31,923	829,145
j6017_6	127,143	74,143	850,138	132,556	46,556	917,754
j603_5	133,554	95,554	1212,911	136,809	32,816	730,647
j604_6	107,862	90,861	836,919	119,262	33,27	1537,342
j601_6	100,863	57,863	855,509	112,924	27,928	748,168
j6011_8	126,418	94,418	1093,891	115,600	30,600	727,212
j6012_10	139,037	104,037	1370,824	133,543	34,553	721,327

Nota: los valores son los encontrados simulando 10000 escenarios

En la **Tabla 14** se observa que el procedimiento de optimización basado en el método de duraciones redundantes obtuvo un mejor makespan esperado para 7 de las 10 instancias j30, por

lo tanto, se muestra inferior el AG propuesto. El valor absoluto de la diferencia del makespan esperado entre los métodos para cada una de las instancias seleccionadas se encuentra entre 0,151 y 13,075 (ver **Tabla 16**).

Tabla 16

Diferencia entre los makespan esperados de las instancias j30

Instancia	Makespan esperado		Valor absoluto de la diferencia
	AG	**DR**	
j301_2	93,999	88,185	5,814
j3014_3	105,531	98,493	7,038
j3023_9	104,211	102,099	2,112
j3044_9	101,903	102,484	0,581
j3020_7	75,095	74,944	0,151
j301_10	91,279	88,014	3,265
j3019_5	92,105	91,548	0,557
j306_3	98,619	101,983	3,364
j3037_8	141,944	128,869	13,075
J3047_6	99,482	102,360	2,878

En cuanto a las instancias j60 en la **Tabla 15** se observa que el AG propuesto obtuvo un mejor makespan para 5 de las 10 instancias, con respecto al procedimiento de optimización basado en el método de duraciones redundantes, por lo tanto, se evidencia que ambos métodos tuvieron un comportamiento similar. El valor absoluto de la diferencia del makespan esperado entre los métodos para cada una de las instancias seleccionadas se encuentra entre 1,256 y 13,018 (ver **Tabla 17**).

Tabla 17

Diferencia entre los makespan esperados de las instancias j60

Instancia	Makespan esperado		Valor absoluto de la diferencia
	AG	**DR**	
j6014_2	113,877	123,905	10,028
j607_7	153,781	139,962	13,018
j601_2	133,496	129,813	3,23

j6022_10	128,178	126,922	1,256
j6017_6	127,143	132,556	3,956
j603_5	133,554	136,809	3,255
j604_6	107,862	119,262	11,430
j601_6	100,863	112,924	12,061
j6011_8	126,418	115,600	10,818
j6012_10	139,037	133,543	5,503

El índice de robustez de la calidad para las instancias j30, fue mejor en 6 de las 10 para el AG. En cuanto a las instancias j60, el índice de robustez de la calidad fue mejor en todas las instancias para el procedimiento de optimización basado en el método de duraciones redundantes. Respecto al índice de robustez de la solución, fue mejor en todas las instancias j30, para el procedimiento de optimización basado en el método de duraciones redundantes.

Finalmente, el índice de robustez de la solución para 6 de las 10 instancias j60 fue mejor el AG propuesto.

Para analizar el comportamiento de los métodos en la simulación, se grafican los resultados de los makespan obtenidos en la programación reactiva para las instancias j3037_8 y j6022_10, ver **Figura 14, Figura 145. Figura 16** y **Figura 17**, agrupando por intervalos dichos makespan (eje horizontal) para hallar su frecuencia (eje vertical). Los diagramas para otras instancias se muestran en los apéndices N y Ñ.

Figura 14

Makespan por simulación instancia j3037_8 con AG

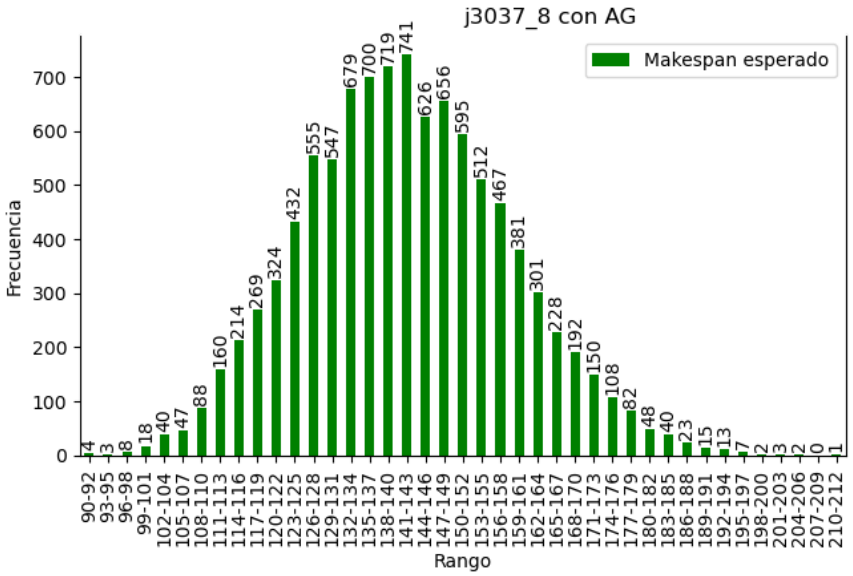


Figura 15

Makespan por simulación instancia j3037_8 con DR

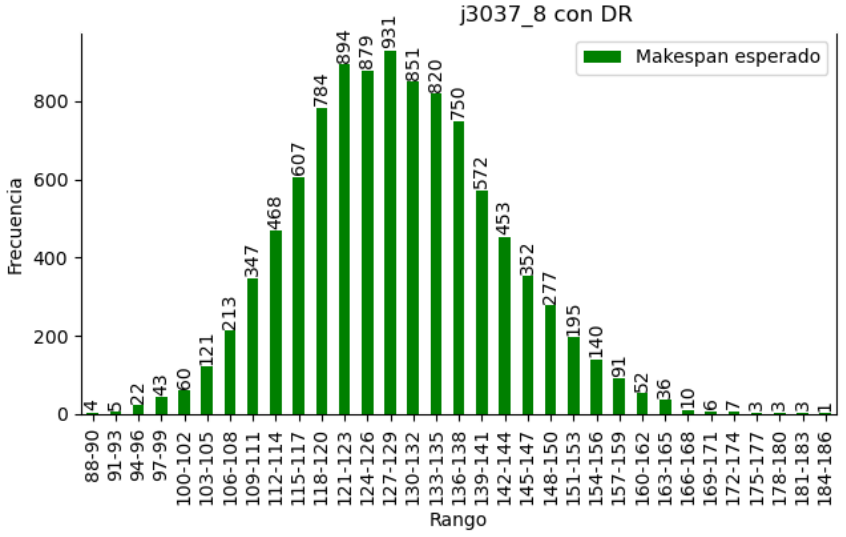


Figura 16

Makespan por simulación instancia j6022_10 con AG

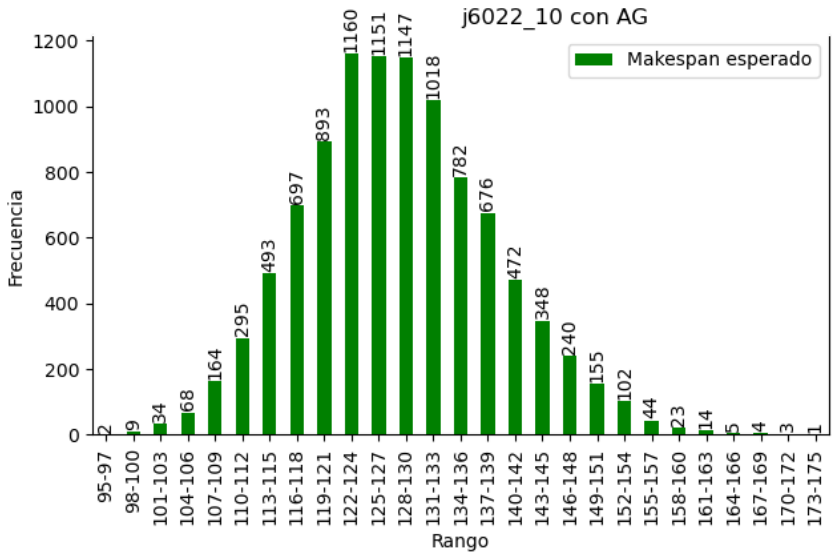
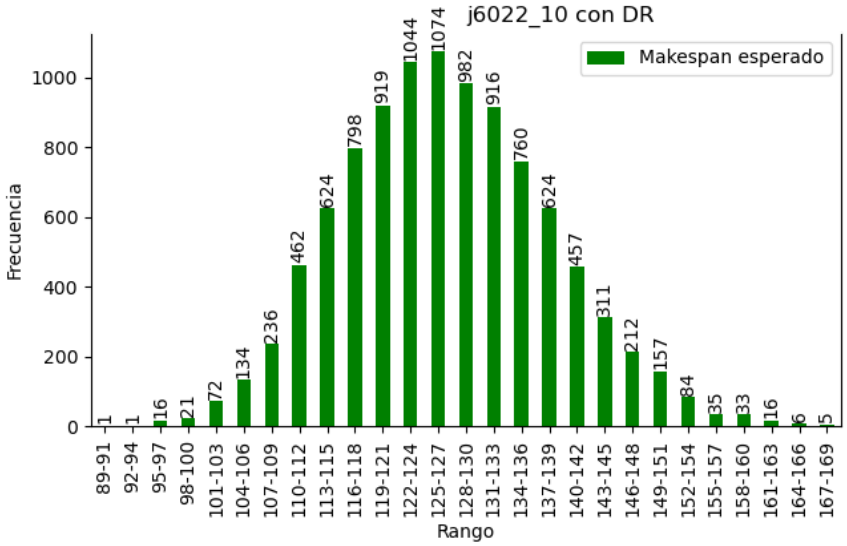


Figura 17

Makespan por simulación instancia j6022_10 con DR



Para seguir analizando el comportamiento de los métodos estudiados, se realizó un diagrama de Pareto con los resultados de los makespan obtenidos en la programación reactiva para las instancias j3037_8 y j6022_10, ver **Figura 18**, **Figura 19**, **Figura 20** y **Figura 21** organizando de mayor a menor los intervalos con mayor frecuencia. Los diagramas para otras instancias se muestran en los apéndices N y Ñ.

Figura 18

Diagrama de Pareto makespan por simulación j3037_8 con AG

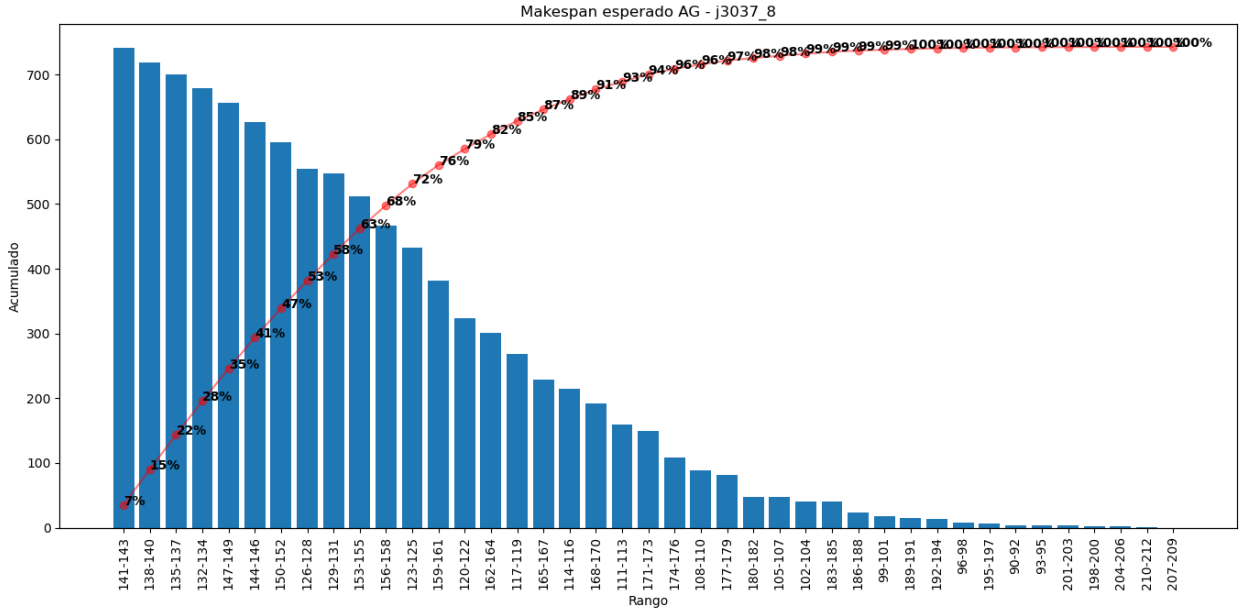


Figura 19

Diagrama de Pareto makespan por simulación j3037_8 con DR

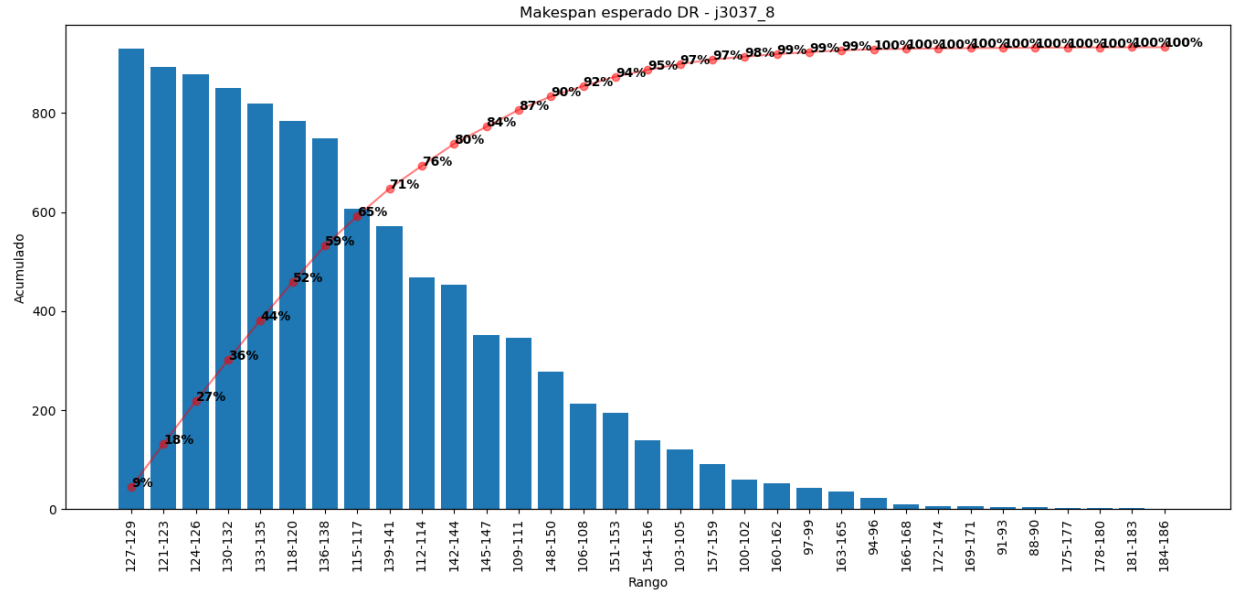


Figura 20

Diagrama de Pareto makespan por simulación j6022_10 con AG

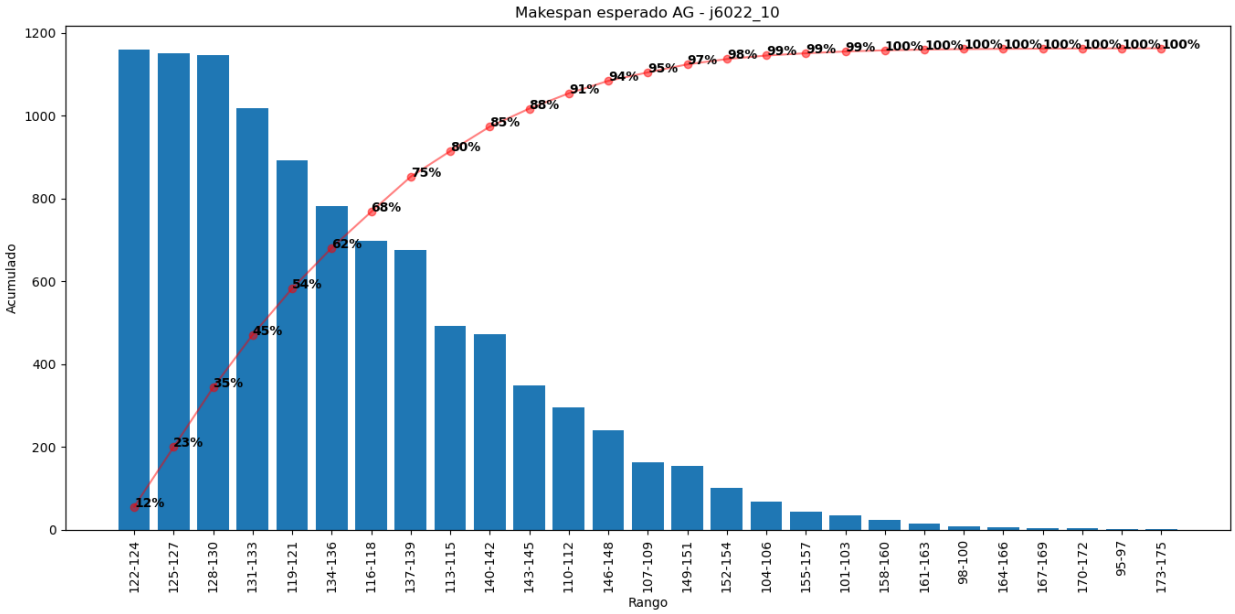
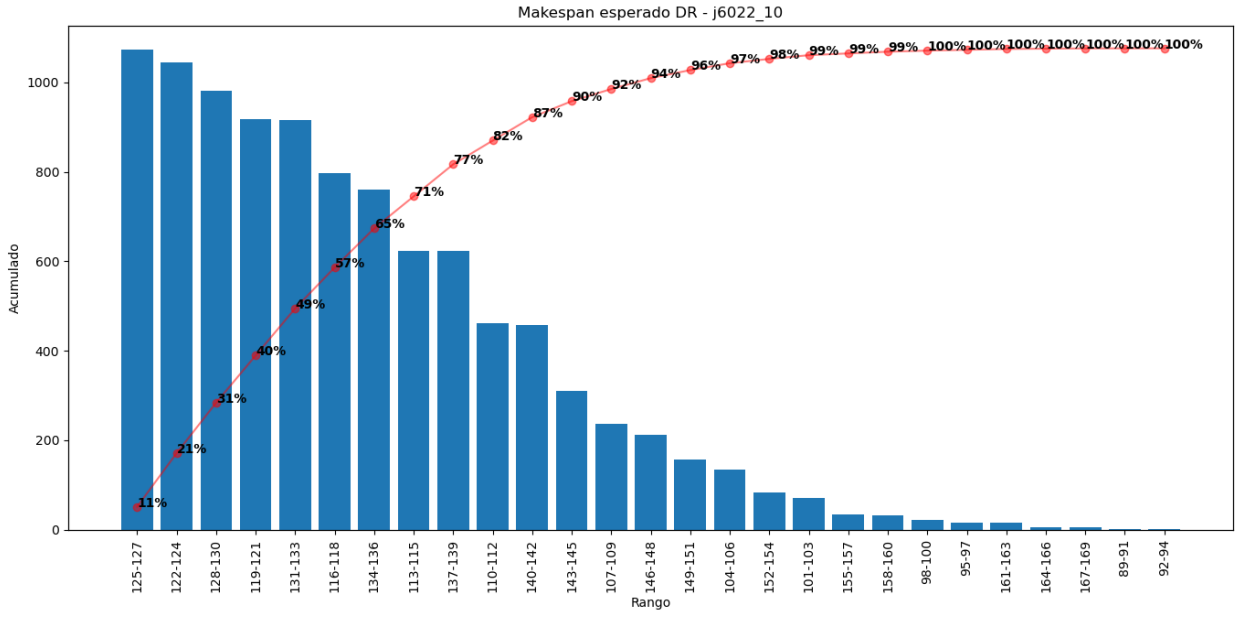


Figura 21

Diagrama de Pareto makespan por simulación j6022_10 con DR



En la **Figura 1**8 se observa que, en la simulación de la solución obtenida con el AG para la instancia j3037_8, aproximadamente el 80% de los makespan están dentro del rango [120, 161],

mientras que en para la simulación con el procedimiento de optimización basado en el método duraciones redundantes, esta misma cantidad de datos se encuentran dentro del rango [112, 144], ver **Figura 19**.

En la **Tabla 18** se muestran los rangos en donde se encuentran aproximadamente el 80% de los makespan obtenidos, en la simulación de algunas instancias j30 y j60, con las soluciones de cada uno de los procedimientos comparados. Se observa que el AG propuesto tiene valores más altos en cuanto a magnitud, pero rangos más pequeños en cuanto a posibles resultados en 4 de las instancias analizadas.

Tabla 18

Rangos de concentración del 80% aproximado de los makespan

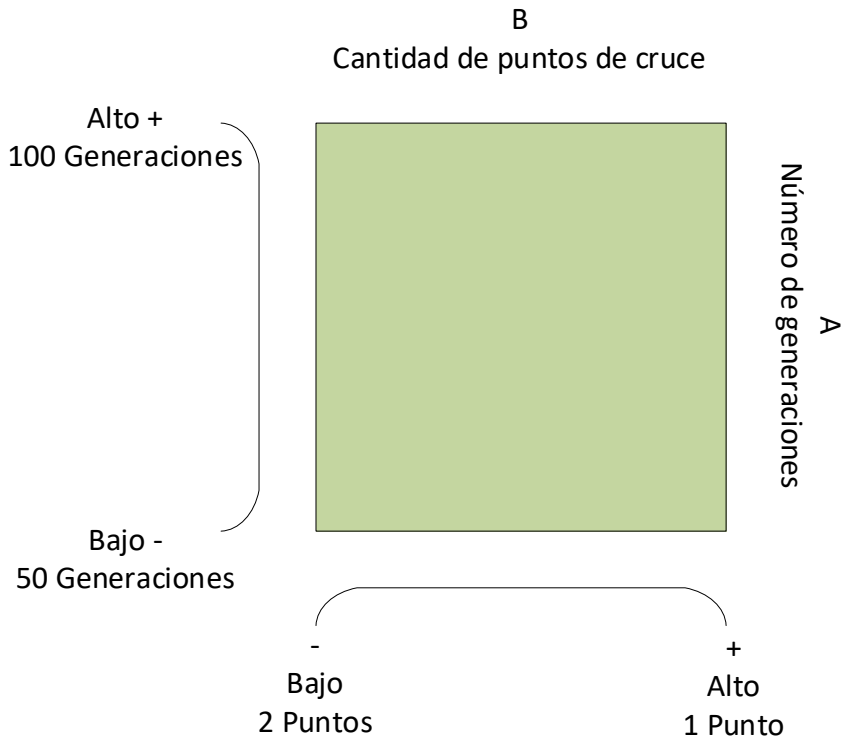
	Rango AG		**Rango DR**	
Instancia	**valor inferior**	**valor superior**	**valor inferior**	**valor superior**
j3044_9	88	117	88	116
j306_3	82	111	87	116
j3037_8	120	161	112	144
j604_6	93	122	107	139
j6022_10	113	139	110	139
j6014_2	103	126	112	144

7.2 Diseño de Experimento para el AG

Finalmente se hizo un diseño de experimento para comprobar la incidencia de los siguientes parámetros en la solución: cantidad de generaciones y cantidad de puntos de cruce, por lo tanto, se realizó un análisis de varianza ANOVA para un diseño 2^2 con 3 réplicas para la instancia j306_3. Los factores y sus niveles se muestran en la **Figura *22***.

Figura 22

Factores y niveles del diseño de experimento de parámetros del AG



Las 3 hipótesis planteadas son:

$$H_0: \alpha_1 = \alpha_2 = \alpha_3 = 0 \ (el\ factor\ A\ no\ influye)$$

$$H_1: algún\ \alpha_i \neq 0 \ \ (el\ factor\ A\ \ influye)$$

$$H_0: \beta_1 = \beta_2 = \beta_3 = 0 \ (el\ factor\ B\ no\ influye)$$

$$H_1: algún\ \beta_i \neq 0 \ \ (el\ factor\ B\ \ influye)$$

$$H_0: (\alpha\beta)_{11} = \cdots = (\alpha\beta)_{33} = 0 \ (no\ hay\ interacción)$$

$$H_1: algún\ (\alpha\beta)_{ij} \neq 0 \ \ (hay\ interacción)$$

Los resultados obtenidos de las réplicas para cada combinación de factores se muestran en la **Tabla 19** y en el apéndice O y P.

Tabla 19

Replicas según los niveles de cada factor, diseño de experimento del AG

Factor		Contaminación de tratamiento		Réplica			
A	**B**			**I**	**II**	**III**	**total**
-	-	A bajo	B bajo	95	95	94	284
-	+	A alto	B bajo	91	91	93	275
+	-	A bajo	B alto	94	94	94	282
+	+	A alto	B alto	90	89	90	269

Los valores F0 obtenidos en el análisis de varianza, para el factor A y B, permiten rechazar las hipótesis nulas respectivas, por lo tanto, se concluye que los efectos principales son estadísticamente significativos, en cuanto a la interacción AB, con F0 menor al F teórico, no hay evidencia suficiente que permita rechazar la hipótesis nula, por ende, no hay interacción entre estos factores.

Tabla 20

Análisis de varianza para número de generaciones y cantidad de puntos de cruce

Fuente de variación	Suma de cuadrados	Grados de libertad	Promedio de los cuadrados	F	Probabilidad	Valor crítico para F
A	5,333	1,000	5,333	10,667	0,011	5,318
B	40,333	1,000	40,333	80,667	0,000	5,318
AB	1,333	1,000	1,333	2,667	0,141	5,318
Error	4	8	0,5			
Total	51	11				

8. Conclusiones

En la presente investigación se concluye a un nivel de confianza del 95%, que los makespan promedio obtenidos del esquema de generación de secuencias en paralelo, con el cual se realizó la programación reactiva, no presentan diferencias significativas al usar 1000, 5000 o 10000 escenarios, por lo tanto, se recomienda usar 1000 escenarios para analizar las soluciones generadas por los dos métodos estudiados.

El procedimiento de optimización basado en el método de duraciones redundantes propuesto por (Ortíz Pimiento & Diaz Serna, 2020), obtiene mejores soluciones en la mayoría (8 de 10) de instancias j30 observadas, tomando como primer criterio de evaluación, el makespan esperado, y en todas las instancias es superior en cuanto al índice de robustez de la solución, sin embargo, para el índice robustez de la solución de calidad, el AG propuesto fue mejor en 6 de las 10 instancias.

En cuanto a las instancias j60 el AG obtuvo mejores resultados en 6 de las 10 instancias, para el makespan esperado, aunque, se mostró inferior en todas las instancias tomando como referencia el índice de robustez de la calidad. Finalmente, los dos procedimientos obtuvieron un rendimiento similar, para el índice de robustez de la solución.

Por lo tanto, se concluye que, para planeadores que se interesen en valor el esperado y en los tiempos de inicio de las actividades, en proyectos de tamaño igual a las instancias j30, el procedimiento de optimización basado en el método de duraciones redundantes es mejor opción, sin embargo, el AG propuesto tiene un rendimiento aceptable en cuanto a índices de robustez de la calidad.

Para proyectos de tamaño igual a las instancias j60, los dos métodos presentan rendimientos similares en los makespan esperado y en los índices de robustez de la solución, siendo mejor el procedimiento de optimización basado en el método de duraciones redundantes en términos de índice de robustez de la calidad, por lo anterior, en proyectos de tamaño j60 pueden ser usados cualquiera de los dos métodos, sin embargo, si se desea obtener mayor certeza en cuanto al makespan esperado, se recomienda usar el procedimiento de optimización basado en el método de duraciones redundantes.

Se concluye que los rangos en los cuales se encuentran alrededor del 80% de los makespan son más amplios para el método propuesto por (Ortíz Pimiento & Diaz Serna, 2020), por lo tanto, el AG propuesto entrega una solución en un espacio de resultados más pequeño, puesto que contiene una mayor concentración de valores esperados en un menor rango.

Se concluye que a un nivel de confianza del 95% los parámetros que garantizan mejores resultados para el AG propuestos son: cruce de 2 puntos y 100 generaciones.

9. Recomendaciones

Se recomienda experimentar otra forma de generación del riesgo para la simulación de escenarios, por ejemplo, que en cada uno de ellos se pueda dar o no la ocurrencia del riesgo en distintas actividades de acuerdo con probabilidades correspondientes.

Se recomienda experimentar el funcionamiento del algoritmo genético en un caso de estudio, debido a que con ello se puede realizar un análisis con datos reales de variabilidad.

Se recomienda experimentar con un algoritmo genético similar, donde en la última generación se evalué la aptitud del individuo mediante un SGS paralelo, usando 100 o más

escenarios, y para las anteriores generaciones se utilice el mismo SGS paralelo, en un número pequeño de escenarios o incluso en un solo escenario, lo anterior, debido al ahorro en tiempo computacional que pueda generar.

Se recomienda para futuras investigaciones, tener en cuenta el nivel de variabilidad del método a proponer, debido a que, mayor variabilidad en las duraciones de un método en relación con otro representa un mayor espacio de resultados, generando con ello, desventaja al comparar.

Se recomienda para futuras investigaciones que aborden el RCPSP con duración de actividades aleatoria, considerar también la variabilidad en la disponibilidad de recursos, además la posibilidad de que una actividad pueda ejecutarse de diferentes modos.

Para las investigaciones en donde se desarrollen algoritmos genéticos que busquen solucionar el RCPSP con duración de actividades aleatoria, sería interesante analizar el efecto que producen en la solución parámetros como, cantidad de generaciones, tamaños de población, cantidad de puntos de cruce, probabilidad de mutación, entre otros. Además, se recomienda estudiar los resultados en relación con la complejidad de las instancias.

Referencias Bibliografía

Alipouri, Y., Sebt, M. H., Ardeshir, A., & Zarandi, M. H. F. (2020). A mixed-integer linear programming model for solving fuzzy stochastic resource constrained project scheduling problem. *Operational Research*, *20*(1), 197–217. https://doi.org/10.1007/s12351-017-0321-x

Ashtiani, B., Leus, R., & Aryanezhad, M.-B. (2011). New competitive results for the stochastic resource-constrained project scheduling problem: Exploring the benefits of pre-processing. *Journal of Scheduling*, *14*(2), 157–171. https://doi.org/10.1007/s10951-009-0143-7

Ballestín, F. (2007). When it is worthwhile to work with the stochastic RCPSP? *Journal of Scheduling*, *10*(3), 153–166. https://doi.org/10.1007/s10951-007-0012-1

Ballestín, Francisco. (2007). When it is worthwhile to work with the stochastic RCPSP? *Journal of Scheduling*, *10*(3), 153–166. https://doi.org/10.1007/s10951-007-0012-1

Baradaran, S., Fatemi Ghomi, S. M. T., Mobini, M., & Hashemin, S. S. (2010). A hybrid scatter search approach for resource-constrained project scheduling problem in PERT-type networks. *Advances in Engineering Software*, *41*(7–8), 966–975. https://doi.org/10.1016/j.advengsoft.2010.05.010

Brcic, M., Kalpic, D., & Fertalj, K. (2012). Resource Constrained Project Scheduling under Uncertainty: A Survey. *23rd Central European Conference on Information and Intelligent Systems*, 401–409. http://bib.irb.hr/datoteka/590574.2012_Resource_Constrained_Project_Scheduling_under_Uncertainty-_A_Survey.pdf

Brčić, M., Kalpić, D., & Katić, M. (2014). Proactive reactive scheduling in resource constrained projects with flexibility and quality robustness requirements. *Lecture Notes in Computer*

Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), *8596 LNCS*, 112–124. https://doi.org/10.1007/978-3-319-09174-7_10

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

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

Bruni, M. E., Beraldi, P., & Guerriero, F. (2015). The stochastic resource-constrained project scheduling problem. In *Handbook on Project Management and Scheduling Vol. 2*. Springer International Publishing. https://doi.org/10.1007/978-3-319-05915-0_7

Bruni, M. E., Beraldi, P., Guerriero, F., & Pinto, E. (2011). A heuristic approach for resource constrained project scheduling with uncertain activity durations. *Computers and Operations Research*, *38*(9), 1305–1318. https://doi.org/10.1016/j.cor.2010.12.004

Bruni, M. E., Di Puglia Pugliese, L., Beraldi, P., & Guerriero, F. (2018). A computational study of exact approaches for the adjustable robust resource-constrained project scheduling problem. *Computers and Operations Research*, *99*, 178–190. https://doi.org/10.1016/j.cor.2018.06.016

Chakrabortty, R. K., Abbasi, A., & Ryan, M. J. (2019). A Risk Assessment Framework for Scheduling Projects With Resource and Duration Uncertainties. *IEEE Transactions on Engineering Management*. https://doi.org/10.1109/TEM.2019.2943161

Chakrabortty, R. K., Rahman, H. F., & Ryan, M. J. (2020). Efficient priority rules for project scheduling under dynamic environments: A heuristic approach. *Computers and Industrial*

Engineering, *140*. https://doi.org/10.1016/j.cie.2020.106287

Chakrabortty, R. K., & Ryan, M. J. (2019). An uncertainty tolerant approach for stochastic resource constrained project scheduling problems. *2019 IEEE Technology and Engineering Management Conference, TEMSCON 2019*. https://doi.org/10.1109/TEMSCON.2019.8813613

Chakrabortty, R. K., Sarker, R. A., & Essam, D. L. (2017). Resource constrained project scheduling with uncertain activity durations. *Computers and Industrial Engineering*, *112*, 537–550. https://doi.org/10.1016/j.cie.2016.12.040

Chand, S, Singh, H. K., & Ray, T. (2016). Finding robust solutions for resource constrained project scheduling problems involving uncertainties. *2016 IEEE Congress on Evolutionary Computation, CEC 2016*, 225–232. https://doi.org/10.1109/CEC.2016.7743799

Chand, Shelvin, Singh, H. K., & Ray, T. (2017). A heuristic algorithm for solving resource constrained project scheduling problems. *2017 IEEE Congress on Evolutionary Computation, CEC 2017 - Proceedings*, 225–232. https://doi.org/10.1109/CEC.2017.7969317

Chen, L., & Zhang, Z. (2016). Preemption resource-constrained project scheduling problems with fuzzy random duration and resource availabilities. *Journal of Industrial and Production Engineering*, *33*(6), 373–382. https://doi.org/10.1080/21681015.2016.1140089

Chen, Z., Demeulemeester, E., Bai, S., & Guo, Y. (2018). Efficient priority rules for the stochastic resource-constrained project scheduling problem. *European Journal of Operational Research*, *270*(3), 957–967. https://doi.org/10.1016/j.ejor.2018.04.025

Chicano García, J. F. (2007). *Metaheurísticas e Ingeniería del Software*. 281. http://dialnet.unirioja.es/servlet/tesis?codigo=18354

Choi, J., Realff, M. J., & Lee, J. H. (2004). Dynamic programming in a heuristically confined state space: A stochastic resource-constrained project scheduling application. *Computers and Chemical Engineering*, *28*(6–7), 1039–1058. https://doi.org/10.1016/j.compchemeng.2003.09.024

Creemers, S. (2015). Minimizing the expected makespan of a project with stochastic activity durations under resource constraints. *Journal of Scheduling*, *18*(3), 263–273. https://doi.org/10.1007/s10951-015-0421-5

Creemers, S. (2014). The resource-constrained project scheduling problem with stochastic activity durations. *IEEE International Conference on Industrial Engineering and Engineering Management*, *2015-Janua*, 453–457. https://doi.org/10.1109/IEEM.2014.7058679

Csébfalvi, A., Csébfalvi, G., & Danka, S. (2011). Fuzzification of the resource-constrained project scheduling problem: A fight against nature. *ECTA 2011 FCTA 2011 - Proceedings of the International Conference on Evolutionary Computation Theory and Applications and International Conference on Fuzzy Computation Theory and Applications*, 286–291. https://www.scopus.com/inward/record.uri?eid=2-s2.0-84862212207&partnerID=40&md5=180ad2a462052abb51c67f322678c9b2

Danka, S. (2014). Sounds of silence: A sampling-based bi-criteria harmony search metaheuristic for the resource constrained project scheduling problem with uncertain activity durations and cash flows. *Periodica Polytechnica Civil Engineering*, *58*(2), 93–104. https://doi.org/10.3311/PPci.7051

Davari, M., & Demeulemeester, E. (2019a). Important classes of reactions for the proactive and reactive resource-constrained project scheduling problem. *Annals of Operations Research*, *274*(1–2), 187–210. https://doi.org/10.1007/s10479-018-2899-7

Davari, M., & Demeulemeester, E. (2019b). The proactive and reactive resource-constrained project scheduling problem. *Journal of Scheduling*, *22*(2), 211–237. https://doi.org/10.1007/s10951-017-0553-x

Fang, C., Kolisch, R., Wang, L., & Mu, C. (2015). An estimation of distribution algorithm and new computational results for the stochastic resource-constrained project scheduling problem. *Flexible Services and Manufacturing Journal*, *27*(4), 585–605. https://doi.org/10.1007/s10696-015-9210-x

Faria, J. M. P., Araújo, M. M. T., & Tereso, A. P. (2015). Project management under uncertainty: A study on solution methods. In S. K.S. (Ed.), *Proceedings of the 26th International Business Information Management Association Conference - Innovation Management and Sustainable Economic Competitive Advantage: From Regional Development to Global Growth, IBIMA 2015* (pp. 1–10). International Business Information Management Association, IBIMA. https://www.scopus.com/inward/record.uri?eid=2-s2.0-84976406974&partnerID=40&md5=ab8db9f8ff66e347f031674063f31360

Fernandez, A. A., Armacost, R. L., & Pet-Edwards, J. J. (1998). Understanding simulation solutions to resource constrained project scheduling problems with stochastic task durations. *EMJ - Engineering Management Journal*, *10*(4), 5–13. https://doi.org/10.1080/10429247.1998.11415002

García Sánchez, Á. (2013). *Técnicas metaheurísticas*. 47. http://www.iol.etsii.upm.es/arch/metaheuristicas.pdf

González C, J., Galvis S, D., & Hurtado T, L. (2014). La distribución Beta Generalizada como un modelo de sobrevivencia para analizar la evasión universitaria. *Estudios Pedagógicos (Valdivia)*, *40*(1), 133–144. https://doi.org/10.4067/s0718-07052014000100008

Huang, Y., Shou, Y., & Zhang, L. (2011). Genetic algorithm for the project scheduling problem with fuzzy time parameters. *IEEE International Conference on Industrial Engineering and Engineering Management*, 689–693. https://doi.org/10.1109/IEEM.2011.6118005

Kassandra, T., Rojali, & Suhartono, D. (2018). Resource-Constrained Project Scheduling Problem using Firefly Algorithm. In B. W. W. L. A. S. R. F. G. A. A. S. W. S. D. Meiliana Arifin Y. (Ed.), *Procedia Computer Science* (Vol. 135, pp. 534–543). Elsevier B.V. https://doi.org/10.1016/j.procs.2018.08.206

Knyazeva, M., Bozhenyuk, A., & Rozenberg, I. (2015). Resource-constrained Project Scheduling Approach under Fuzzy Conditions. In S. M. Ginters E. (Ed.), *Procedia Computer Science* (Vol. 77, pp. 56–64). Elsevier B.V. https://doi.org/10.1016/j.procs.2015.12.359

Kolisch, R., & Sprecher, A. (1997). PSPLIB - A project scheduling problem library. *European Journal of Operational Research*, *96*(1), 205–216. https://doi.org/10.1016/S0377-2217(96)00170-1

Kutsch, E., & Hall, M. (2005). Intervening conditions on the management of project risk: Dealing with uncertainty in information technology projects. *International Journal of Project Management*, *23*(8), 591–599. https://doi.org/10.1016/j.ijproman.2005.06.009

Lamas, P., & Demeulemeester, E. (2016). A purely proactive scheduling procedure for the resource-constrained project scheduling problem with stochastic activity durations. *Journal of Scheduling*, *19*(4), 409–428. https://doi.org/10.1007/s10951-015-0423-3

Lambrechts, O., Demeulemeester, E., & Herroelen, W. (2008). Proactive and reactive strategies for resource-constrained project scheduling with uncertain resource availabilities. *Journal of Scheduling*, *11*(2), 121–136. https://doi.org/10.1007/s10951-007-0021-0

Leus, R., Rostami, S., & Creemers, S. (2016). New benchmark results for the stochastic resource-

constrained project scheduling problem. *IEEE International Conference on Industrial Engineering and Engineering Management*, *2016-Janua*, 204–208. https://doi.org/10.1109/IEEM.2015.7385637

Li, H., & Womer, N. K. (2015). Solving stochastic resource-constrained project scheduling problems by closed-loop approximate dynamic programming. *European Journal of Operational Research*, *246*(1), 20–33. https://doi.org/10.1016/j.ejor.2015.04.015

Li, S., Jia, Y., & Wang, J. (2012). A discrete-event simulation approach with multiple-comparison procedure for stochastic resource-constrained project scheduling. *International Journal of Advanced Manufacturing Technology*, *63*(1–4), 65–76. https://doi.org/10.1007/s00170-011-3885-2

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

Lombardi, M., & Milano, M. (2010). Constraint based scheduling to deal with uncertain durations and self-timed execution. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *6308 LNCS*, 383–397. https://doi.org/10.1007/978-3-642-15396-9_32

Lombardi, M., & Milano, M. (2012). A min-flow algorithm for Minimal Critical Set detection in Resource Constrained Project Scheduling. *Artificial Intelligence*, *182–183*, 58–67. https://doi.org/10.1016/j.artint.2011.12.001

Lu, R., & Li, L. (2008). A heuristic approach for rework-based product design project scheduling problem. *Chinese Control and Decision Conference, 2008, CCDC 2008*, 1486–1491.

https://doi.org/10.1109/CCDC.2008.4597565

Ma, Z., Demeulemeester, E., He, Z., & Wang, N. (2019). A computational experiment to explore better robustness measures for project scheduling under two types of uncertain environments. *Computers and Industrial Engineering*, *131*, 382–390. https://doi.org/10.1016/j.cie.2019.04.014

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

Masmoudi, M., & Haït, A. (2012). Fuzzy uncertainty modelling for project planning: Application to helicopter maintenance. *International Journal of Production Research*, *50*(13), 3594–3611. https://doi.org/10.1080/00207543.2012.670925

Mogaadi, H., & Chaar, B. F. (2016). Scenario-based evolutionary approach for robust RCPSP. *Advances in Intelligent Systems and Computing*, *427*, 45–55. https://doi.org/10.1007/978-3-319-29504-6_6

Moreno Díaz, P., Huecas Fernández-Toribio, G., Sánchez Allende, J., & García Manso, A. (2007). Metaheurísticas de optimización combinatoria: uso de Simulated Annealing para un problema de calendarización. *Tecnología y Desarrollo*, *5*, 301.

MorilloTorres, D. (2017). *Eficiencia Energética en la Programación de Tareas con Recursos Restringidos*.

Ortiz Pimiento, N. R., & Díaz Serna, F. J. (2019). Relative Average Deviation as Measure of Robustness in the Stochastic Project Scheduling Problem. *Revista Facultad de Ingeniería*, *28*(52), 77–97. https://doi.org/10.19053/01211129.v28.n52.2019.9756

Ortíz Pimiento, N. R., & Diaz Serna, F. J. (2020). An optimization model to solve the resource constrained project scheduling problem RCPSP in new product development projects•. *Dyna*,

87(212), 179–188. https://doi.org/10.15446/dyna.v87n212.81269

Palace T, G., & Postrand G, M. (2018). *ARTÍCULO DE REFLEXIÓN C OMPUTATIONAL C OMPLEXITY : T IME AND C OMPLEJIDAD C OMPUTACIONAL : T IEMPO Y*. 12–18.

Pet-Edwards, J., & Mollaghasemi, M. (1996). Simulation and genetic algorithm approach to stochastic research constrained project scheduling. *Southcon Conference Record*, 333–338. https://www.scopus.com/inward/record.uri?eid=2-s2.0-0029719571&partnerID=40&md5=645b295578a7dcf1420e62349b0a708e

Python Software Foundation. (s. f.). *Functions Defined*. Python.org. https://www.python.org/

Roghanian, E., Alipour, M., & Rezaei, M. (2018). An improved fuzzy critical chain approach in order to face uncertainty in project scheduling. *International Journal of Construction Management*, *18*(1), 1–13. https://doi.org/10.1080/15623599.2016.1225327

Rostami, S., Creemers, S., & Leus, R. (2018). New strategies for stochastic resource-constrained project scheduling. *Journal of Scheduling*, *21*(3), 349–365. https://doi.org/10.1007/s10951-016-0505-x

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

Spiegel, M. R., & Stephens, L. J. (2009). *Estadística* (McGRAW-HIL).

Tahooneh, A., & Ziarati, K. (2011). Using artificial bee colony to solve stochastic resource constrained project scheduling problem. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *6728 LNCS*(PART 1), 293–302. https://doi.org/10.1007/978-3-642-21515-5_35

Tseng, C.-C., & Ko, P.-W. (2016). Measuring schedule uncertainty for a stochastic resource-constrained project using scenario-based approach with utility-entropy decision model. *Journal of Industrial and Production Engineering*, *33*(8), 558–567. https://doi.org/10.1080/21681015.2016.1172522

Wang, J., Hu, X., Demeulemeester, E., & Zhao, Y. (2019). A bi-objective robust resource allocation model for the RCPSP considering resource transfer costs. *International Journal of Production Research*. https://doi.org/10.1080/00207543.2019.1695168

William Wallace. (2014). Gestión de Proyectos - Definición de Proyectos. *Edinburgh Business School*, *2014*(1106), 68.

Yu, Y., Wang, C., & Zhu, J. (2012). An agent-based dynamic scheduling solution for resource-constrained project scheduling. *Proceedings of International Conference on Computational Intelligence, Modelling and Simulation*, 120–123. https://doi.org/10.1109/CIMSim.2012.44

Zafra-Cabeza, A., Ridao, M. A., & Camacho, E. F. (2008). Using a risk-based approach to project scheduling: A case illustration from semiconductor manufacturing. *European Journal of Operational Research*, *190*(3), 708–723. https://doi.org/10.1016/j.ejor.2007.06.021

Zaman, F., Elsayed, S., Sarker, R., & Essam, D. (2018). Scenario-Based Solution Approach for Uncertain Resource Constrained Scheduling Problems. *2018 IEEE Congress on Evolutionary Computation, CEC 2018 - Proceedings*. https://doi.org/10.1109/CEC.2018.8477756

Zhang, X., Cui, N., Bie, L., & Chai, Y. (2011). Timely project completion probability and stability cost on the interaction among uncertainty of random duration, service level and feeding buffer in a RCPSP environment. *International Conference on Management and Service Science, MASS 2011*. https://doi.org/10.1109/ICMSS.2011.05998239

Zhong, C. J., Yu, Y., & Jia, Y. L. (2014). Project scheduling problem with stochastic resource-

dependent activity duration. *Applied Mechanics and Materials*, *483*, 607–610. https://doi.org/10.4028/www.scientific.net/AMM.483.607

Zhong, X., & Zhang, Z. (2015). Proactive scheduling procedures for RCPSP with beta distributed durations and exponential distributed resources. *Advances in Intelligent Systems and Computing*, *362*, 855–867. https://doi.org/10.1007/978-3-662-47241-5_72