Vol. xx, n.° x, pp. xx-xx, xxxx
Revista UIS Ingenierías
Página de la revista: revistas.uis.edu.co/index.php/revistauisingenierias

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.

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

Duban Alfonzo Oviedo Daza[1a], Daniel Nicolas Delgado Gómez [1b], Néstor Raúl Ortiz Pimiento[1c]

[1] Grupo de optimización de sistemas productivos, administrativos y logísticos (OPALO), Escuela de Estudios Industriales y Empresariales, Universidad Industrial de Santander, Colombia. Orcid: [c] 0000-0001-9776-4258, [d] 0000-0003-1778-9768. Correos electrónicos: [a] duban.oviedo@correo.uis.edu.co, [b] idanield01@gmail.com, [c] nortiz@uis.edu.co.

Resumen

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 las actividades de un proyecto cumpliendo las restricciones de precedencias y la disponibilidad de recursos, teniendo como objetivo conseguir 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, este esquema fue utilizado parar evaluar la aptitud de cada individuo, haciendo usa de una lista de actividades (representación del individuo) como regla de prioridad, calculando para n diferentes escenarios el makespan esperado. 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 [32], 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.

Palabras clave: Algoritmo Genético, *Genetic algorithm*, Duración Aleatoria, *Random Duration*, Problema de programación de proyectos, *Project Scheduling Problem*, Recursos Restringidos, *Resource Constrained*.

Abstract

This research addresses the Resource Constrained Project Scheduling Problem (RCPSP) with a random duration of activities, which consists of scheduling the activities of a project meeting the precedence restrictions and the availability of resources. with the objective of achieving a base programming that manages to minimize the execution time of the project under different simulated scenarios. To solve this problem, a genetic algorithm (GA) supported in a sequence generation scheme (SGS) in parallel, this scheme was used to evaluate the aptitude of each individual, making use of a list of activities (Represent the individual) as a priority rule, calculating the expected makespan for different scenarios. 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 [32], 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.

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

ISSN impreso: 1657 - 4583. ISSN en línea: 2145 – 8456, **CC BY-ND 4.0**
Autores (A. Apellido)., "Título en el idioma en que se presenta el artículo," *Rev. UIS Ing.*, vol. xx, no. x, pp. xx-xx, año. doi:

1. 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 [1] 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..

2. Metodología

2.1. Revisión de literatura

Los primeros intentos por dar uniformidad a la notación, clasificación y modelo del RCPSP fue hecho por [2] 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, problemas llamados también SRCPSP "Stochastic Resource Constrained Project Scheduling Problem" en donde la característica estocástica se puede generar por uno o más factores, estos han sido abordados en modelos como los trabajado por: [3] con duración de actividades aleatoria, [4] con aleatoriedad en la disponibilidad de recursos, [5] con curva de aprendizaje LC (learning curve) y un comportamiento variable del consumo de recursos DC (Resource Requirement Drop-Down Curve), [6] con incertidumbre tanto en la duración de las actividades como en el consumo o disponibilidad, [7] en el Multimodo RCPSP o MRCPSP con duración de actividades estocástica, en donde cada actividad puede realizarse de múltiples formas, [8] con el multimodo con retrasos mínimos y máximos MRCPSP/Max. [9] en el multiproyecto RCPSP o RCMPSP con duraciones de actividades aleatorias donde se trabajan con recursos disponible que son compartidos y consumidos por varios proyectos.

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 haciéndolo estocástico mediante la definición de la duración de las actividades con una distribución de probabilidad asociada; algunos autores que han planteado matemáticamente este problema son: [10], [11], [12], [13], [14], [15] ,[16], [17], [18]
[3], [19], [20], [21]
y [22].

2.1.2. Enfoques para resolver el SRCPSP

Por su parte [23] explican que hay tres enfoques principales para resolver SRCPSP: 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 autores como: [1] [24], [10] y [25], estos últimos con el PR-RCPSP (programación de proyectos proactivo y reactivo con limitaciones de recursos).

Según [23] la programación predictiva *i*gnora la estocasticidad del problema y utiliza estimaciones puntuales, generalmente expectativas o mediana, en lugar de variables aleatorias, logrando ser resuelto de forma determinística; la programación reactiva toman decisiones de programación durante el tiempo de ejecución del proyecto; la programación proactiva crea un cronograma de referencia de mayor solidez a

resultados inesperados como una duración de la actividad más larga de lo previsto.

[10], [11], [12] abordan el problema desde un enfoque reactivo, en done la duración final de la actividad se conoce solo cuando se ha completado dicha actividad, por ello, plantean el modelo matemático en donde la solución a la función objetivo está dada en términos de una política de prioridad, la cual define para cada tiempo de decisión $t \geq 0$ las actividades que se inician en t, calculando el valor esperado del makespan que genera dicha política.

2.1.3. SGS.

[10] mencionó que los algoritmos heurísticos principalmente usan dos procedimientos o esquemas de generación de programa (SGS), adaptados de RCPSP determinístico, estos son el SGS paralelo y el SGS en serie; sin embargo, se han diseñados nuevos esquemas de generación de programas adaptados a las características SRCPSP, tal es el caso de [10] que quiso unir las ventajas del SGS en serie y en paralelo, proponiendo el SGS estocástico. [26] trabajaron con el DSGS (esquema de generación de programación dinámica) ya que el SGS en serie tal como está planteado no se puede aplicar durante la ejecución real del proyecto o en un enfoque reactivo, [27] por su parte, trabajaron con el esquema de generación dinámica estocástica SDGS.

2.1.4. Algoritmos genéticos.

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

[28] propusieron un algoritmo llamado búsqueda genética local específica, el cual es un AG estándar que trabaja únicamente en un conjunto de números locales, el algoritmo luego de obtener los hijos hace una búsqueda local generando una solución vecina a estos. [10] para generar la población inicial propuso un muestreo aleatorio sesgado basado en el arrepentimiento, el cual trabajaba con la regla de prioridad del último tiempo de finalización (LFT) para calcular la probabilidad de seleccionar una actividad; este autor calcula la aptitud del individuo en n escenarios tomando el individuo (lista de actividades) como regla de prioridad en un SGS.

[13] trabajaron su modelo matemático con parámetros de tiempo difusos y propusieron un algoritmo genético basado en un esquema de generación de programación paralela difuso. [1] propusieron un algoritmo genético de dos fases con búsqueda local, en donde adaptaron el muestreo aleatorio sesgado basado en el arrepentimiento y el calculo de la aptitud del individuo, propuestos por [10], los autores tambien usan un operador llamado doble justificación, ademas mejoran los resultados del AG con una busqueda local.

[12] propusieron un algoritmo genético biobjectivo, en este aplicaron a los dos mejores cromosomas de cada generación un operador de cambio de subred; también usaron 3 diferentes operadores de mutación (uniforme, incremental, no uniforme) y un operador de desplazamiento a la izquierda. [15] crearon la población inicial del algoritmo mediante una heurística de muestreo de múltiples pasos, estos autores usaron para medir la aptitud del individuo la robustez de este, 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 [29] en su método heurístico.

[30] propusieron un algoritmo evolutivo simple basado en población.
[31] plantearon un modelo biobjectivo, 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.
La mayoría de esto autores probaron sus algoritmos evolutivos con instancias de la librería PSPLIB, con proyectos tipo j30, 160 y/o j120.

2.2. Formulación del modelo matemático

El modelo matemático elegido después de realizar una revisión de los modelos propuestos en la literatura, es el trabajado por [10] y [11], los cuales, al igual que el presente trabajo, manejan un enfoque reactivo-proactivo, en donde el objetivo es conseguir que la solución genere en promedio, el menor makespan en diferentes escenarios posibles de la ejecución del proyecto.

Estos autores inician su planteamiento afirmando que el RCPSP con actividades aleatorias, se puede definir con base en el RCPSP determinístico.

2.2.1. Descripción del problema modelo RCPSP determinístico

El modelo RCPSP determinístico tiene un conjunto de actividades denotado por un subíndice i o j conformado por $N = \{1,2,3, \dots, n+2\}$ donde n representa el número de actividades del proyecto a las cuales se le agrega 2 actividades ficticias para denotar el inicio y el fin del

proyecto, las cuales son respectivamente 1 y n+2, estas actividades tienen duración cero. La duración de la actividad i es d_i, está dada por una distribución 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 recursos $k = 1,2, \dots, K$, los cuales son renovables en cada momento t, la disponibilidad de cada recurso en cualquier momento t es R_k, la actividad i tiene un requerimiento r_{ik} de cada recurso k, las actividades al ser ejecutadas deben 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 horario factible para el proyecto, él cual es representado 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 denotado como C_{max}.

2.2.2. Modelo matemático

Conjuntos

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

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

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

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

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.

2.2.3. Descripción del problema modelo RCPSP determinístico 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 [11] 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.

2.2.4. Modelo matemático

Conjuntos

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

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

Restricciones

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

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

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

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 $S_i^{\lambda}(p)$.

2.3. Diseño del algoritmo genetico

Para darle solución al modelo planteado se desarrolló un algoritmo genético soportado en un esquema generador de secuencias en paralelo (representado en un diagrama general en la figura 1). Este AG fue programado en el 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.

2.3.1. 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

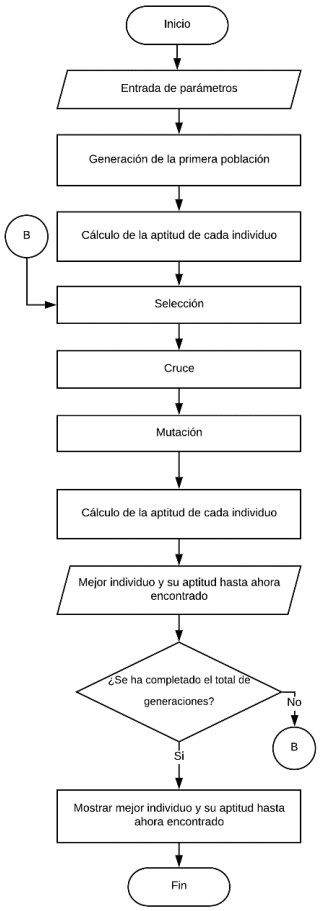

Figura 1. *Diagrama general del Algoritmo Genético.*

2.3.2. Representación de la Solución.

Una de las representaciones encontradas con más frecuencia en la literatura, es la codificación del cromosoma forma de lista de actividades, los autores [30] argumentan que es simple, intuitiva y ayuda a mantener la naturaleza discreta del problema mientras permite una fácil interpretación, basándose en ello se utilizó 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 2 se puede apreciar un cromosoma ejemplo para un proyecto de 8 actividades

reales y dos ficticias, estas últimas con duración cero representando el inicio y fin del proyecto.

1	2	3	5	4	7	6	8	9	10

Figura 2. *Cromosoma*

2.3.3. Generación de la población inicial.

Para generar la población inicial se crea cada uno de sus individuos aplicando un esquema generador de secuencias (SGS)

2.3.3.1. SGS.

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 https://www.python.org/) 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.

2.3.3.2. Regla de prioridad LFT.

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

2.3.3.3. 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

2.3.3.4. 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.

2.3.4. 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.

2.3.4.1 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 3, donde d=duración y r= recursos, con una lista actividades λ = { 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.

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 4, en donde el eje vertical son las unidades de recurso y el horizontal son los tiempo o momentos *t*.

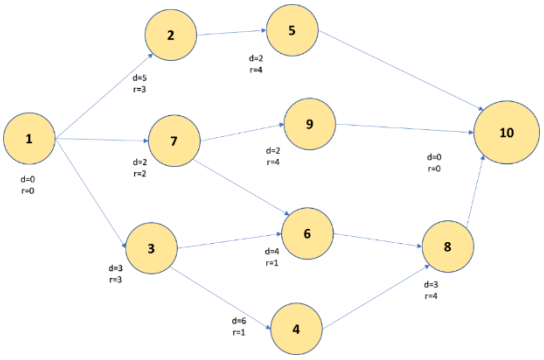

Figura 3. Grafica de la red de proyecto.

Figura 4. 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).

2.3.5. Generación de nuevas poblaciones.

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

2.3.5.1. 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.

2.3.5.2. 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 5, en primer lugar, se genera la ubicación del punto aleatoriamente (en este caso se estableció el punto entre las posiciones 6 y 7).

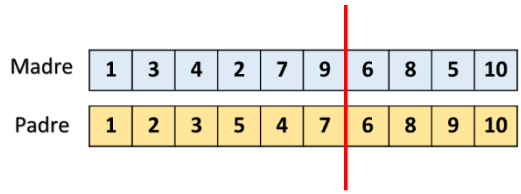

Figura 5. Ubicación de 1 punto de cruce.

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 6

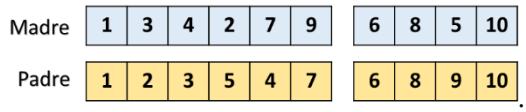

Figura 6. Cromosomas subdivididos.

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 7 y 8.

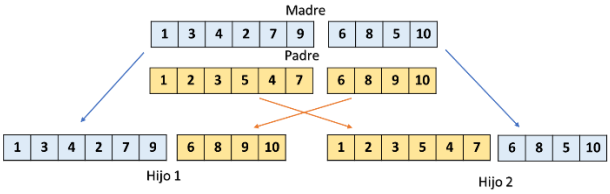

Figura 7. Procedimiento de creación de los hijos.

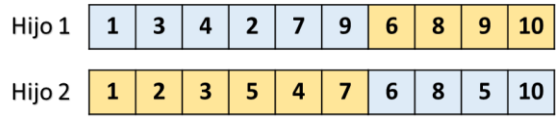

Figura 8. Cromosomas hijos

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 9.

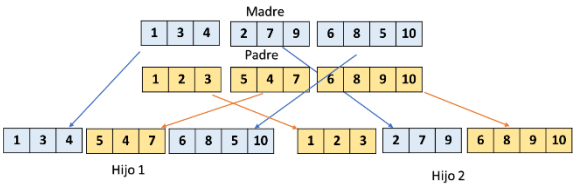

Figura 91. Conformación de hijos con un cruce de 2 puntos.

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

2.3.5.3. 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 el gen posterior, ver figura 10.

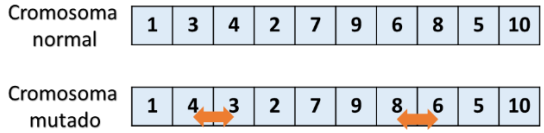

Figura 20. Procedimiento de mutación.

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.

3. 4. 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* [32], 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.

3.4.1. 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:

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

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

3.4.2. 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 [33].

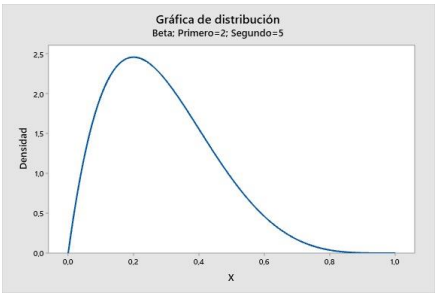

Figura 31. Grafica de la distribución beta con parámetros p=2 y q=5

3.4.3. 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%

3.4.4. 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.

3.4.4.1. 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.

3.4.4.2. 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.

3.4.4.3. 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, … , 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

Variables

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:

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

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

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

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

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

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

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

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.

3.4.5. 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, 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.

3.4.5.1. 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 $\boldsymbol{N} = \{\mathbf{1}, \mathbf{2}, \mathbf{3}, \dots, \boldsymbol{n} + \mathbf{2}\}$ con i como subíndice, un conjunto de escenario E= $\{\mathbf{1}, \mathbf{2}, \mathbf{3}, \dots, \boldsymbol{nscen}\}$ con e como subíndice, el tiempo de inicio esperado de la actividad i representado con $\boldsymbol{s_i}$, el makespan esperado del proyecto ($\boldsymbol{s_{n+2}}$), el tiempo de inicio de la actividad i en el escenario e denotado como $\boldsymbol{s_{ie}}$ y el makespan del proyecto en el escenario e $\left(\boldsymbol{s_{n+2,e}}\right)$.

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

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

4. Experimentación y resultados

Las tablas 1 y 2 muestran los resultados obtenidos por los métodos comparados de cada una de las 20 instancias.

Tabla 1. 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 2. Resultados instancias j60 de los métodos comparados

Se observa que la aptitud del mejor individuo para cada instancia (la cual es un promedio obtenido de 50

	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

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.

En las tablas 3, 4, 5 y 6 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 3.1 Resultados del makespan esperado e índices de robustez para instancias j30 con AG

	AG		
Instancia	**Makespan Esperado**	**Robustez**	
		De la calidad	**De la solución**
j301_2	93,999	22,105	471,265
j3014_3	105,531	23,545	353,453
j3023_9	104,211	21,306	424,974
j3044_9	101,903	29,912	394,446
j3020_7	75,095	24,101	372,712
j301_10	91,279	21,485	462,978
j3019_5	92,105	23,167	388,885
j306_3	98,619	26,636	403,904
j3037_8	141,944	17,374	518,183
j3047_6	99,482	23,507	272,657

Tabla 4.2 Resultados del makespan esperado e índices de robustez para instancias j30 con DR

	Duraciones Redundantes		
Instancia	**Makespan Esperado**	**Robustez**	
		De la calidad	**De la solución**
j301_2	88,185	26,193	315,351
j3014_3	98,493	26,494	332,283
j3023_9	102,099	32,100	327,796
j3044_9	102,484	16,746	284,158
j3020_7	74,944	21,960	276,623
j301_10	88,014	24,052	300,030
j3019_5	91,548	20,694	280,410
j306_3	101,983	20,200	335,413
j3037_8	128,869	30,896	479,193
j3047_6	102,360	27,362	264,527

Tabla 5.3 Resultados del makespan esperado e índices de robustez para instancias j60 con AG

AG			
Instancia	**Makespan Esperado**	**Robustez**	
		De la calidad	**De la solución**
j6014_2	113,877	71,877	653,208
j607_7	153,781	99,781	1.082,918
j601_2	133,496	73,496	1244,918
j6022_10	128,178	66,178	782,782
j6017_6	127,143	74,143	850,138
j603_5	133,554	95,554	1212,911
j604_6	107,862	90,861	836,919
j601_6	100,863	57,863	855,509
j6011_8	126,418	94,418	1093,891
j6012_10	139,037	104,037	1370,824

Tabla 6.4 Resultados del makespan esperado e índices de robustez para instancias j60 con DR

Duraciones Redundantes			
Instancia	**Makespan Esperado**	**Robustez**	
		De la calidad	**De la solución**
j6014_2	123,905	28,920	718,08
j607_7	139,962	41,964	799,588
j601_2	129,813	37,813	1.050,451
j6022_10	126,922	31,923	829,145
j6017_6	132,556	46,556	917,754
j603_5	136,809	32,816	730,647
j604_6	119,262	33,27	1537,342
j601_6	112,924	27,928	748,168
j6011_8	115,600	30,600	727,212
j6012_10	133,543	34,553	721,327

El procedimiento de optimización basado en el método de 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 7).

Tabla 75. 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 las tablas 5 y 6 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 8).

Tabla 8. 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 la instancia j3037_8, ver figura 12 y 13, agrupando por intervalos dichos makespan (eje horizontal) para hallar su frecuencia (eje vertical).

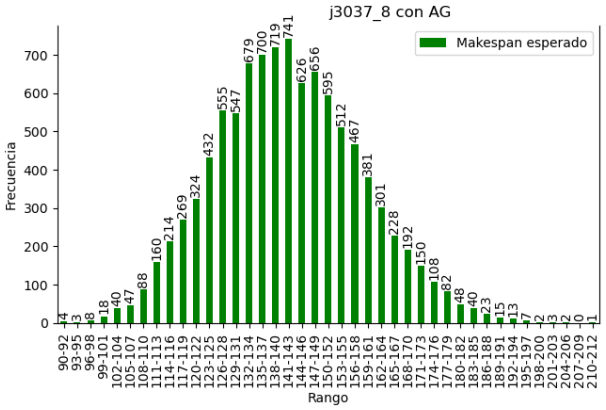

Figura 42. Makespan por simulación instancia j3037_8 con AG

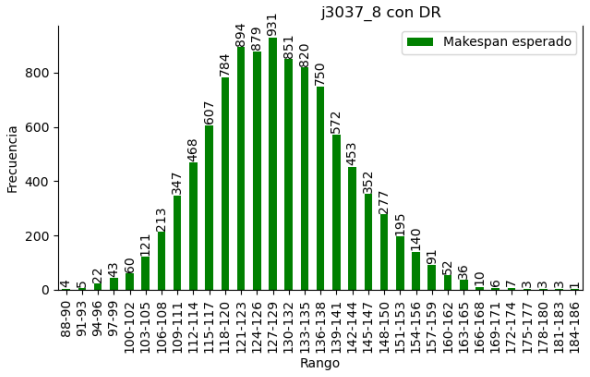

Figura 53. Makespan por simulación instancia j3037_8 con DR.

En la tabla 9 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 9. 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

5. Conclusiones

Según la revisión de literatura, son pocos los trabajos de investigación que tratan del problema de ruteo de vehículos con drones (VRPD), por lo tanto, es un tema de investigación que puede ser considerado nuevo e innovador dentro del área del ruteo de vehículos, con múltiples aplicaciones reales.

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 la 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 (Ortiz 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.

Referencias

[1] B. Ashtiani, R. Leus, and M.-B. Aryanezhad, "New competitive results for the stochastic resource-constrained project scheduling problem: Exploring the benefits of pre-processing," *J. Sched.*, vol. 14, no. 2, pp. 157–171, 2011, doi: 10.1007/s10951-009-0143-7.

[2] P. Brucker, A. Drexl, R. Möhring, K. Neumann, and E. Pesch, "Resource-constrained project scheduling: Notation, classification, models, and methods," *Eur. J. Oper. Res.*, vol. 112, no. 1, pp. 3–41, 1999, doi: 10.1016/S0377-2217(98)00204-5.

[3] S. Baradaran, S. M. T. Fatemi Ghomi, M. Mobini, and S. S. Hashemin, "A hybrid scatter search approach for resource-constrained project scheduling problem in PERT-type networks," *Adv. Eng. Softw.*, vol. 41, no. 7–8, pp. 966–975, 2010, doi: 10.1016/j.advengsoft.2010.05.010.

[4] O. Lambrechts, E. Demeulemeester, and W. Herroelen, "Proactive and reactive strategies for resource-constrained project scheduling with uncertain resource availabilities," *J. Sched.*, vol. 11, no. 2, pp. 121–136, 2008, doi: 10.1007/s10951-007-0021-0.

[5] R. Lu and L. Li, "A heuristic approach for rework-based product design project scheduling problem," in *Chinese Control and Decision Conference, 2008, CCDC 2008*, 2008, pp. 1486–1491, doi: 10.1109/CCDC.2008.4597565.

[6] X. Zhong and Z. Zhang, "Proactive scheduling procedures for RCPSP with beta distributed durations and exponential distributed resources," *Adv. Intell. Syst. Comput.*, vol. 362, pp. 855–867, 2015, doi: 10.1007/978-3-662-47241-5_72.

[7] A. Drexl and J. Gruenewald, "Nonpreemptive multi-mode resource-constrained project scheduling," *IIE Trans. (Institute Ind. Eng.*, vol. 25, no. 5, pp. 74–81, 1993, doi: 10.1080/07408179308964317.

[8] R. Heilmann, "Resource-constrained project scheduling: A heuristic for the multi-mode case," *OR Spektrum*, vol. 23, no. 3, pp. 335–357, 2001, doi: 10.1007/PL00013354.

[9] T. R. Browning and A. A. Yassine, "A random generator of resource-constrained multi-project network problems," *J. Sched.*, vol. 13, no. 2, pp. 143–161, 2010, doi: 10.1007/s10951-009-0131-y.

[10] F. Ballestín, "When it is worthwhile to work with the stochastic RCPSP?," *J. Sched.*, vol. 10, no. 3, pp. 153–166, 2007, doi: 10.1007/s10951-007-0012-1.

[11] A. Tahooneh and K. Ziarati, "Using artificial bee colony to solve stochastic resource constrained project scheduling problem," *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 6728 LNCS, no. PART 1, pp. 293–302, 2011, doi: 10.1007/978-3-642-21515-5_35.

[12] Y. Shou and W. Wang, "Robust optimization-based genetic algorithm for project scheduling with stochastic activity durations," *Information*, vol. 15, no. 10, pp. 4049–4064, 2012, [Online]. Available: https://www.scopus.com/inward/record.uri?eid=2-s2.0-84865446411&partnerID=40&md5=7a75825ea55b3407cdfa83fbd6e6cb21.

[13] Y. Huang, Y. Shou, and L. Zhang, "Genetic algorithm for the project scheduling problem with fuzzy time parameters," in *IEEE International Conference on Industrial Engineering and Engineering Management*, 2011, pp. 689–693, doi: 10.1109/IEEM.2011.6118005.

[14] C. J. Zhong, Y. Yu, and Y. L. Jia, "Project scheduling problem with stochastic resource-dependent activity duration," *Appl. Mech. Mater.*, vol. 483, pp. 607–610, 2014, doi: 10.4028/www.scientific.net/AMM.483.607.

[15] H. Mogaadi and B. F. Chaar, "Scenario-based evolutionary approach for robust RCPSP," *Adv.*

Intell. Syst. Comput., vol. 427, pp. 45–55, 2016, doi: 10.1007/978-3-319-29504-6_6.

[16] R. Leus, S. Rostami, and S. Creemers, "New benchmark results for the stochastic resource-constrained project scheduling problem," in *IEEE International Conference on Industrial Engineering and Engineering Management*, 2016, vol. 2016-Janua, pp. 204–208, doi: 10.1109/IEEM.2015.7385637.

[17] S. Rostami, S. Creemers, and R. Leus, "New strategies for stochastic resource-constrained project scheduling," *J. Sched.*, vol. 21, no. 3, pp. 349–365, 2018, doi: 10.1007/s10951-016-0505-x.

[18] F. Zaman, S. Elsayed, R. Sarker, and D. Essam, "Scenario-Based Solution Approach for Uncertain Resource Constrained Scheduling Problems," 2018, doi: 10.1109/CEC.2018.8477756.

[19] M. Lombardi and M. Milano, "Constraint based scheduling to deal with uncertain durations and self-timed execution," *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 6308 LNCS, pp. 383–397, 2010, doi: 10.1007/978-3-642-15396-9_32.

[20] C. Fang, R. Kolisch, L. Wang, and C. Mu, "An estimation of distribution algorithm and new computational results for the stochastic resource-constrained project scheduling problem," *Flex. Serv. Manuf. J.*, vol. 27, no. 4, pp. 585–605, 2015, doi: 10.1007/s10696-015-9210-x.

[21] S. Creemers, "Minimizing the expected makespan of a project with stochastic activity durations under resource constraints," *J. Sched.*, vol. 18, no. 3, pp. 263–273, 2015, doi: 10.1007/s10951-015-0421-5.

[22] R. K. Chakrabortty, R. A. Sarker, and D. L. Essam, "Resource constrained project scheduling with uncertain activity durations," *Comput. Ind. Eng.*, vol. 112, pp. 537–550, 2017, doi: 10.1016/j.cie.2016.12.040.

[23] M. Brcic, D. Kalpic, and K. Fertalj, "Resource Constrained Project Scheduling under Uncertainty: A Survey," *23rd Cent. Eur. Conf. Inf. Intell. Syst.*, pp. 401–409, 2012, [Online]. Available: http://bib.irb.hr/datoteka/590574.2012_Resource_Constrained_Project_Scheduling_under_Uncertainty-_A_Survey.pdf.

[24] M. E. Bruni, P. Beraldi, and F. Guerriero, *The stochastic resource-constrained project scheduling problem*. Springer International Publishing, 2015.

[25] M. Davari and E. Demeulemeester, "Important classes of reactions for the proactive and reactive resource-constrained project scheduling problem," *Ann. Oper. Res.*, vol. 274, no. 1–2, pp. 187–210, 2019, doi: 10.1007/s10479-018-2899-7.

[26] R. K. Chakrabortty, H. F. Rahman, and M. J. Ryan, "Efficient priority rules for project scheduling under dynamic environments: A heuristic approach," *Comput. Ind. Eng.*, vol. 140, 2020, doi: 10.1016/j.cie.2020.106287.

[27] M. E. Bruni, P. Beraldi, F. Guerriero, and E. Pinto, "A heuristic approach for resource constrained project scheduling with uncertain activity durations," *Comput. Oper. Res.*, vol. 38, no. 9, pp. 1305–1318, 2011, doi: 10.1016/j.cor.2010.12.004.

[28] S. Liu, K. L. Yung, and W. H. Ip, "Genetic local search for resource-constrained project scheduling under uncertainty," *Int. J. Inf. Manag. Sci.*, vol. 18, no. 4, pp. 347–363, 2007, [Online]. Available: https://www.scopus.com/inward/record.uri?eid=2-s2.0-37149032066&partnerID=40&md5=21db86d5354ecb8f83ac929c71d20201.

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

[30] S. Chand, H. K. Singh, and T. Ray, "Finding robust solutions for resource constrained project scheduling problems involving uncertainties," in *2016 IEEE Congress on Evolutionary Computation, CEC 2016*, 2016, pp. 225–232, doi: 10.1109/CEC.2016.7743799.

[31] J. Wang, X. Hu, E. Demeulemeester, and Y. Zhao, "A bi-objective robust resource allocation model for the RCPSP considering resource transfer costs," *Int. J. Prod. Res.*, 2019, doi: 10.1080/00207543.2019.1695168.

[32] N. R. Ortíz Pimiento and F. J. Diaz Serna, "An optimization model to solve the resource constrained project scheduling problem RCPSP in new product development projects•," *Dyna*, vol. 87, no. 212, pp. 179–188, 2020, doi: 10.15446/dyna.v87n212.81269.

[33] Z. Chen, E. Demeulemeester, S. Bai, and Y. Guo, "Efficient priority rules for the stochastic

resource-constrained project scheduling problem," *Eur. J. Oper. Res.*, vol. 270, no. 3, pp. 957–967, 2018, doi: 10.1016/j.ejor.2018.04.025.

[34] A. Zafra-Cabeza, M. A. Ridao, and E. F. Camacho, "Using a risk-based approach to project scheduling: A case illustration from semiconductor manufacturing," *Eur. J. Oper. Res.*, vol. 190, no. 3, pp. 708–723, Nov. 2008, doi: 10.1016/j.ejor.2007.06.021.