Pontificia Universidad Javeriana Cali

Facultad de Ingeniería y Ciencias

Solución Robusta al Problema de Programación de Actividades con Recursos Restringidos y Duraciones Estocásticas

Autor: María Victoria Gutiérrez Padilla

Director: Daniel Morillo-Torres, Ph.D.

Cali, Colombia

2025

Abstract

En esta investigación se aborda el Uncertain Resource-Constrained Project Scheduling Problem (URCPSP); su versión determinista (el RCPSP) es considerada como el problema NP-hard más importante del área de programación. El RCPSP tiene como objetivo secuenciar las actividades que conforman un proyecto, con el fin de minimizar su tiempo de ejecución, teniendo en cuenta las duraciones de las actividades, sus relaciones de precedencia y el uso de recursos escasos. El URCPSP considera un entorno más realista, donde las duraciones de las actividades pueden variar debido a improvistos. En la literatura se han definido diversos enfoques para su solución, entre ellos destaca la programación robusta, que consiste en encontrar una programación base que sea capaz de absorber posibles perturbaciones mediante una amortiguación temporal. Para conocer qué tan robusta es una solución se han desarrollado diferentes medidas de robustez. Sin embargo, la eficacia de estas medidas es limitada, y ninguna considera simultáneamente la estructura y el estado operativo del problema. Así, la contribución de este proyecto es triple: a) se define una nueva medida de robustez mediante la adaptación de este concepto en sistemas eléctricos, teniendo en cuenta una función lineal por partes para cada actividad que evalúa la importancia de las holguras, considerando el grado en el que se cumple la holgura esperada de cada actividad; b) se diseña una metodología general de solución para el URCPSP que considera el componente estocástico de las duraciones de las actividades y la nueva medida de robustez, e incluye la propuesta de una metaheurística evolutiva; y c) se adapta una librería estándar de casos de prueba incluyendo las duraciones estocásticas basados en perfiles realistas de actividades bajo riesgo. Los resultados obtenidos validan la eficiencia de la propuesta y muestran cómo la distribución inteligente de las holguras permite alcanzar una mayor robustez.

Keywords: SRCPSP; incertidumbre; Robustez; holguras; Algoritmo Genético.

Contents

1 Introduction

Los problemas de programación de actividades son considerados de los más relevantes y entre los primeros en ser estudiados en el área de la Investigación Operativa. Esto se debe a la complejidad y alta aplicabilidad de estos problemas, abarcando múltiples sistemas que requieran la gestión de recursos escasos. Así, además de encontrarse en la industria, manufactura o construcción, existen aplicaciones en otras áreas como medicina, aeronáutica, educación o ergonomía [1, 2, 3, 4]. Estos problemas tienen como objetivo secuenciar actividades (o tareas) en el tiempo, haciendo uso óptimo de los recursos disponibles para el procesamiento de cada actividad [5]. Los primeros estudios enfocados en esta problemática se publicaron a mediados de 1 950, donde tenían en cuenta únicamente restricciones de precedencia entre tareas y las soluciones buscaban secuenciar dichas tareas de forma que se minimice el tiempo de finalización de un proyecto [6, 7].

En las últimas décadas, se ha prestado especial atención en la programación de tareas con restricciones de uso de recursos limitados [8]. Este problema se conoce en la literatura como el Resource-Constrained Project Scheduling Problem (RCPSP) y es de gran interés tanto en la academia como en la industria, debido a que es considerado como el problema base más importante en el área de Programación de Tareas [9]. Diversos sistemas de secuenciación como el Job Shop, el Flow Shop, o el Parallel Machine Scheduling, son casos particulares del RCPSP. Además, el RCPSP tiene aplicaciones en diferentes campos como la planeación de producción, la distribución, el transporte, la gestión de proyectos, la optimización de la cadena de suministro, entre otros [10]. El RCPSP tiene como objetivo determinar el tiempo de inicio de las actividades que conforman un proyecto (y así su secuencia), con el fin de cumplir un objetivo determinado. Este normalmente está relacionado con la minimización del tiempo de ejecución total del proyecto, teniendo en cuenta las actividades, sus tiempos de procesamiento, y el consumo de recursos para su realización. Así, de forma simultánea considera dos tipos de restricciones: 1) el cumplimiento de las relaciones de precedencia entre actividades, y 2) no sobrepasar el máximo uso de recursos restringidos en cada instante del tiempo [9].

En un entorno real, las duraciones de las actividades pueden variar debido a improvistos como una avería en las máquinas, errores humanos, condiciones meteorológicas inciertas, entre otros. Esto puede causar un aumento en el tiempo de finalización de las actividades (y del proyecto), lo que a su vez aumenta los costos de ejecución y podría causar el incumplimiento a los clientes. Considerar estas disrupciones en un sistema de secuenciación da lugar al RCPSP no determinista o bajo incertidumbre, conocido como el Uncertain RCPSP (URCPSP). Este será el problema objeto de estudio en esta investigación. En la literatura existen varios enfoques para hacer frente al URCPSP, donde destaca el denominado enfoque robusto, cuyo objetivo es desarrollar una programación base capaz de absorber las variaciones en la duración de las actividades, minimizando el impacto potencial [11]. La programación base se refiere a una secuenciación fija establecida desde el inicio, que, ante imprevistos, no se modifica, sino que puede gestionar las alteraciones afectando lo menos posible la duración total del proyecto. Durante la última década, la optimización robusta ha sido un frecuente objeto de estudio debido a su buen rendimiento, ya que genera soluciones resistentes a los peores escenarios [12]. Es por esto que esta investigación se centrará en el enfoque robusto para el URCPSP. Sin embargo, se han encontrado tres GAPs de investigación principales en relación

a la solución robusta del URCPSP:

- Se han propuesto muchas medidas de robustez en los últimos años. Sin embargo, la eficacia de estas medidas sigue siendo discutible, ya que aún se desconoce cuál es la medida de robustez más eficaz para la programación robusta en condiciones de incertidumbre. Además, hasta nuestro conocimiento, no existe una medida de robustez para el URCPSP que abarque integralmente las holguras resultantes, actividades sucesoras y el consumo de recursos, considerando también que una mayor holgura no siempre implica una mayor robustez.
- No existe una metodología estándar para resolver el URCPSP que integre su componente estocástico, robustez y al mismo tiempo reduzca su makespan.
- Hasta el momento no se cuenta con una librería de casos de prueba oficial para el URCPSP que considere un enfoque bajo riesgos realista y que permita a los investigadores comparar el desempeño de los algoritmos propuestos.

Para esto, se propone:

- **Una nueva medida de robustez**: la propuesta se basa en una métrica de entropía utilizada en sistemas eléctricos, que, al adaptarla al URCPSP, considera simultáneamente (1) la estructura de la red junto con las actividades sucesoras, el consumo de recursos escasos y las holguras generadas por la programación, y (2) una función de holgura lineal que permite distribuir inteligentemente las holguras.
- **Una metodología de solución general que considera el componente bajo incertidumbre**: esta incluye el diseño de un Algoritmo Genético para la solución del URCPSP que incorpora los operadores más eficientes documentados en la literatura. La metodología propuesta parte de la resolución de múltiples escenarios de cada instancia. Cada solución obtenida se considera una candidata a solución del sistema bajo incertidumbre; posteriormente se evalúa cada solución con base en la nueva medida de robustez, y se selecciona aquella que obtiene el mayor valor.
- **La adaptación y generación de casos de prueba para el URCPSP con duraciones estocásticas**: en la creación de estas instancias se consideró un enfoque bajo riesgos que representa un comportamiento realista. Basado en casos aplicados, se establece que no todas las actividades están sujetas a riesgos, e incluso las que sí lo hacen, pueden tener riesgos distintos, asociados a la naturaleza propia del proceso o actividad.

El resto de este documento se organiza de la siguiente manera: en la Sección 2 se presenta una definición formal del RCPSP en sus dos variantes: determinista y bajo incertidumbre. En la Sección 3 se realiza una revisión de literatura incluyendo ambas variantes del RCPSP, pero enfocándose principalmente en aquellos trabajos que consideren elementos bajo incertidumbre, incluyendo sus enfoques de solución. La Sección 4 presenta la nueva medida de robustez propuesta. En la Sección 5 se propone un Algoritmo Genético y se establece una metodología de solución robusta para el RCPSP bajo incertidumbre. La Sección 6 proporciona los resultados de los experimentos computacionales en diferentes instancias adaptadas

de la PSPLIB para el RCPSP bajo incertidumbre. Finalmente, se presentan las conclusiones de la investigación en la Sección 7.

2 Problem description

Si bien este paper se enfoca en encontrar una solución robusta al URCPSP, en esta sección primero se describe formalmente el RCPSP en su versión determinista para tener una base teórica sobre este problema (Sección 2.1), para luego definir el RCPSP bajo incertidumbre (Sección 2.2).

2.1 Description of the deterministic problem

El RCPSP puede definirse formalmente de la siguiente manera: sea un proyecto con un conjunto de actividades o tareas $I := \{0, \ldots, i, \ldots, (I' + 1)\}$, donde cada actividad i tiene un tiempo de procesamiento P_i, de forma que, una vez iniciada su ejecución, esta no puede ser interrumpida y debe finalizarse. Además, las actividades tienen predecesores, es decir, una actividad no puede llevarse a cabo hasta haberse finalizado todas las actividades que la preceden. Esta relación de precedencias entre actividades está dada por un conjunto de parejas de actividades $PRE := \{(i, j) : i, j \in I\}$, donde j representa una actividad predecesora de la actividad i. Adicionalmente, se tiene un conjunto de recursos renovables $K := \{1, \ldots, K'\}$ que tienen una disponibilidad máxima W_k constante en el tiempo. Cada actividad i requiere durante su duración consumir una cantidad constante R_{ik} del recurso k para llevarse a cabo. Para establecer la formulación matemática, se denota un conjunto $T := \{1, \ldots, T_{\max}\}$, que representa el horizonte de tiempo en el que se realiza el proyecto. Una forma de estimar la cota superior del tiempo máximo de finalización del proyecto ($T_{\max}$) es sumar la duración de todas las actividades ($\sum_{i \in I} P_i$). Finalmente, por convención el conjunto de actividades se dividen en dos grupos: $\{1, \ldots, I'\}$ y $\{0, (I' + 1)\}$, siendo el primero las actividades reales que componen un proyecto, y el segundo son dos actividades adicionales que representan el inicio y final del proyecto, respectivamente. Las actividades adicionales son ficticias, por esto no consumen recursos para ejecutarse ni tienen un tiempo de procesamiento.

Con estas definiciones, el RCPSP puede expresarse mediante el siguiente modelo Mixed-Integer Linear Programming basado en las *On/off formulations*, pertenecientes a las denominadas *Time-indexed formulations* para el RCPSP [13].

Conjuntos

- I : conjunto de actividades $\{0, \ldots, (I' + 1)\}$.
- T : conjunto que representa al horizonte de tiempo $\{1, \ldots, T_{\max}\}$.
- K : conjunto de recursos renovables $\{1, \ldots, K'\}$
- PRE : conjunto de bidimensional (i, j) que determina la relación de precedencia entre actividades i y j.

Parámetros

- P_i : tiempo de procesamiento de la actividad $i \in I$.

- W_k : cantidad total disponible del recurso $k \in K$.
- R_{ik} : consumo del recurso $k \in K$ requerido para llevar a cabo la actividad $i \in I$.
- M : representa un valor constante suficientemente grande, también se le conoce como en la literatura como the Big M.

Variables de decisión

- A_{it}: $\begin{cases} 1, & \text{si la actividad } i \in I \text{ está activa en el tiempo } t \in T \\ 0, & \text{en otro caso} \end{cases}$
- ST_i : tiempo de inicio de la actividad $i \in I$.

Mixed-Integer Linear Programming Model

$$\text{Minimizar } Makespan = ST_{(I'+1)}, \tag{1}$$

Subject to

$$ST_j \geq ST_i + P_i, \qquad \forall (i,j) \in PRE \tag{2}$$

$$ST_i + P_i - 1 \geq t \cdot A_{it}, \qquad \forall i \in I, \forall t \in T \tag{3}$$

$$ST_i \leq t + (1 - A_{it}) \cdot M, \qquad \forall i \in I, \forall t \in T \tag{4}$$

$$\sum_{t \in T} A_{it} = P_i, \qquad \forall i \in I \tag{5}$$

$$\sum_{i \in I} R_{ik} \cdot A_{it} \leq W_k, \qquad \forall k \in K, \forall t \in T \tag{6}$$

$$A_{it} \in \{0,1\}, \qquad \forall i \in I, \forall t \in T \tag{7}$$

$$ST_i \geq 0, \qquad \forall i \in I \tag{8}$$

La expresión (1) es la función objetivo que busca minimizar el tiempo de inicio de la última actividad, es decir la ficticia que representa la finalización de todo el proyecto (*makespan*). Las restricciones (2) garantizan que una actividad no pueda comenzar hasta que sus predecesores hayan finalizado. Las restricciones (3) a (5) aseguran la dinámica de la variable A_{it}, de forma que solo tenga valor de 1 cuando la actividad i se encuentre activa durante su tiempo de inicio hasta su tiempo de finalización. Así, las restricciones (3) obligan a A_{it} a tomar el valor de 0 desde el momento en el que la actividad i finalice hasta T; las restricciones (4) obligan a A_{it} a ser 0 desde $t = 0$ hasta el momento en el que inicie la actividad i; por su parte las restricciones (5) obligan a que existan tantas A_{it} con valor de 1 igual a su duración; obligando a que tome valores de 1 solo desde el tiempo de inicio de la actividad i hasta su finalización. Las restricciones (6) garantizan que durante cada periodo el consumo de recursos de las actividades activas no excedan la disponibilidad máxima de los recursos. Finalmente, las ecuaciones (7) y (8) representan las restricciones de dominio de las variables de decisión.

La Fig. 1 muestra un ejemplo del RCPSP mediante un grafo Activity-On-Node (AON). Este consta de 11 actividades representadas por nodos, incluidas las ficticias (0.10), y dos

tipos de recursos ($W1, W2$) con 5 unidades disponibles de recursos para cada tipo. En la parte superior de cada nodo se encuentra el tiempo de ejecución de cada actividad y en la parte inferior el consumo de recursos necesarios para llevarse a cabo. Finalmente, cada arco dirigido representa las relaciones de precedencia entre actividades.

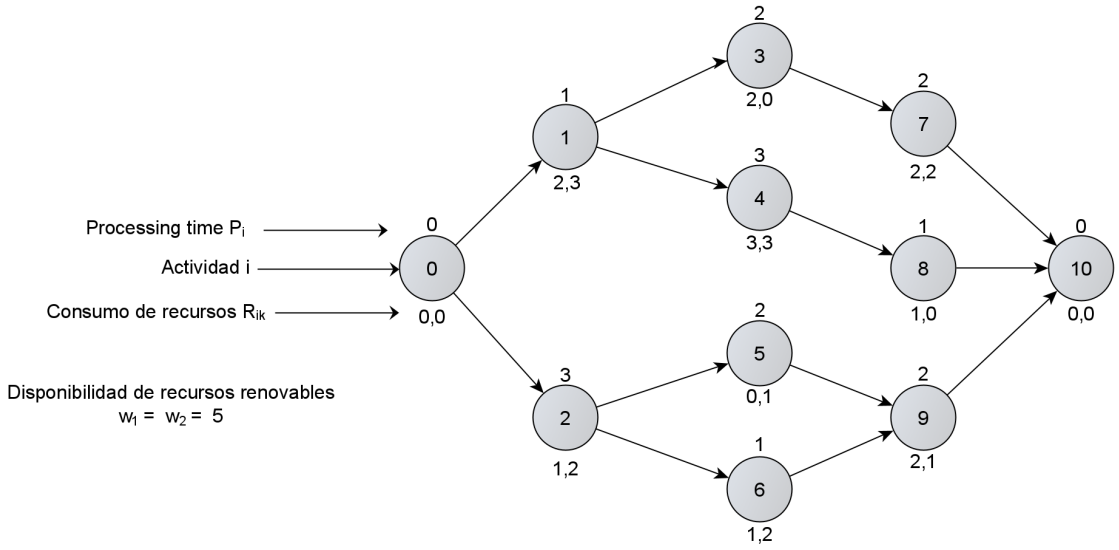

Figure 1: Grafo Activity-On-Node de un ejemplo del RCPSP.

Las soluciones a este problema corresponden a los tiempos de inicio de cada actividad de modo que se haga el proyecto en el menor tiempo posible. Una representación de una solución óptima, mediante un diagrama de Gantt, para el problema presentado en la Fig. 1 se muestra en la Fig. 2. Es importante destacar que pueden existir múltiples soluciones óptimas, por ejemplo, podría obtenerse otra solución óptima cambiando el tiempo de inicio de la actividad 8 de $ST_8 = 4$ a $ST_8 = 5$, ya que la disponibilidad de recursos y las restricciones de precedencia lo permiten.

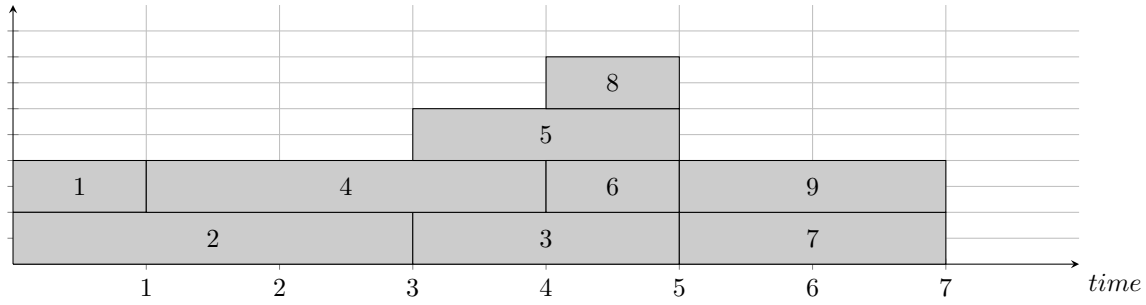

Figure 2: Una solución óptima del problema propuesto en la figura 1.

2.2 Description of the under uncertainty problem

En un sistema realista, algunos parámetros pueden ser inciertos, como la duración de las actividades o la cantidad de recursos disponibles [8]. Esto se debe a la naturaleza dinámica de un entorno real, que está sujeta a disrupciones o riesgos, por lo que pueden haber retrasos en la ejecución de las actividades, los recursos pueden dejar de estar disponibles, las máquinas pueden presentar averías o los empleados pueden incapacitarse [14]. Esto genera efectos secundarios indeseables, por ejemplo, el incumplimiento a los clientes por retrasos en la finalización de los proyectos. Dada esta naturaleza dinámica, el URCPSP ha sido abordado desde diferentes enfoques en la literatura, destacándose la secuenciación robusta, que es la utilizada en esta investigación. Una revisión completa y detallada de este enfoque, así como de las principales contribuciones relacionadas con estas metodologías, se presenta en la Sección 3.2.

3 Literature review

En esta revisión de la literatura se exploran diversos aspectos relacionados con el objeto de estudio de este paper. En la Sección 3.1, se abordan las variantes y métodos de solución para el RCPSP. En la Sección 3.2 se describen los enfoques para resolver el URCPSP. En la Sección 3.3.1 se presenta una revisión sobre medidas de robustez. Finalmente, la Sección 3.3.2 detalla la medida de robustez en sistemas eléctricos que será adaptada en esta investigación.

3.1 RCPSP

El problema base en la Programación de Proyectos es el RCPSP, este cumple las dos interdependencias básicas en los problemas de esta área: la limitación de recursos y las restricciones de precedencia entre actividades [15]. El primer modelo matemático formulado para el RCPSP se desarrolló en 1 969 [16] y en 1 983 se demostró que el RCPSP pertenece a los problemas denominados NP-Hard [17]. Estos son problemas que, a medida que su tamaño aumenta, su resolución exige tiempos de cómputo con crecimiento exponencial, es decir, en el peor de los casos, es inviable resolverlos en tiempo polinomial. A través del tiempo se han estudiado múltiples variantes o clasificaciones del RCPSP. La Tabla 1 resume las principales modificaciones sobre los supuestos para el RCPSP que originan nuevos problemas, categorizados en cuatro dimensiones de acuerdo con Habibi et al. [9]: recursos, concepto de actividades, función objetivo y nivel de información disponible.

En relación con los métodos de solución, estos se pueden clasificar en dos grupos: exactos y de búsqueda aproximdada. Los métodos exactos se caracterizan por resolver los problemas de forma óptima. Sin embargo, solo son eficientes para problemas pequeños. En el caso del RCPSP, la literatura ha mostrado que se pueden resolver en tiempo razonable un máximo de 60 actividades para aquellos casos de prueba altamente restringidos [18]. Esto se debe a la naturaleza NP-Hard del RCPSP [19, 20, 21, 22]. Por otro lado, dentro de los algoritmos aproximados se encuentran los algoritmos heurísticos, que se caracterizan por su flexibilidad y eficiencia computacional para resolver problemas de gran tamaño [18]. No obstante, pueden estancarse en óptimos locales, ya que no cuentan con estrategias para una mejor exploración del espacio factible. Es por esto que se recurre a algoritmos metaheurísticos para resolver problemas de optimización, estos algoritmos describen métodos de solución generales basados

Table 1: Clasificación del RCPSP (adapted from [9]).

Clasificación		Descripción
Por recursos	Renovables	Son aquellos recursos que son accesibles en cada periodo de tiempo con toda su capacidad [15]. Es decir, al finalizar una actividad, pueden ser usados nuevamente para llevar a cabo otra actividad (u otro conjunto de actividades). Este tipo de recursos son limitados por periodo y cada unidad de recurso solo puede ser usada por una actividad simultáneamente. Ejemplo: operarios, máquinas, entre otros.
	No renovables	Son aquellos recursos que, al finalizar una actividad, son consumidos y no pueden volver a ser usados. Este tipo de recursos se limitan por una capacidad total a lo largo del proyecto. Generalmente se consideran en problemas cuyas actividades pueden realizarse de diferentes maneras (o modos) y cada una consume cantidades de recursos diferente. Esto se debe a que, en el caso del RCPSP unimodal, siempre habría que garantizar una cantidad suficiente para procesar todas las actividades. Algunos ejemplos de recursos no renovables son materia prima o presupuesto.
	Recursos doblemente restringidos	Este tipo de recursos se limitan para cada periodo y para todo el proyecto. Ejemplo: el dinero puede ser un recurso doblemente restringido si se limita un presupuesto para cada periodo de tiempo (por ejemplo, para cada día, semana o mes) y al mismo tiempo, se limita un presupuesto total del proyecto (es decir, un presupuesto para el tiempo total de la duración del proyecto).
Por concepto de actividades	Preemptive	Se permite la interrupción de actividades, para ser retomadas en otro instante de tiempo sin costo adicional. El consumo de recursos permanece invariable mientras se encuentra activa la actividad.
	Execution mode	Cada actividad puede asociarse a varios modos de ejecución, y a su vez, cada uno de los modos puede tener diferentes requisitos de recursos y tiempo de procesamiento. Pero, basta con ejecutarse de un solo modo para considerar que la actividad ha sido finalizada.
	Resource request varying with time	El uso de recursos para la ejecución de las actividades varía con el tiempo, haciendo que una actividad pueda consumir mayor o menor cantidad de un recurso para su ejecución dependiendo del periodo de inicio; también puede requerir recursos que inicialmente no consideraba o dejar de necesitar otros que previamente requería.

Table 1: Clasificación del RCPSP (adapted from [9]), continued.

Por función objetivo	Basadas en tiempo	Objetivos relacionados con la finalización de las actividades, uno de los más usados es la minimización del tiempo de ejecución total del proyecto o makespan, pero también se pueden encontrar objetivos como la minimización de la tardanza o de retraso en las actividades.
	Económicas	Minimiza los costos relacionados con la ejecución del proyecto, por ejemplo, costos asociados al uso de recursos, horas extras, retrasos, entre otros. También se puede minimizar el Valor Presente Neto.
	Multiobjetivo	Combinación de dos o más funciones objetivo. Por ejemplo, Laszczyk and Myszkowski [26] abordaron un RCPSP con múltiples habilidades minimizando la duración total del proyecto y el costo total del proyecto. Al-Fawzan and Haouari [27] propusieron un modelo bi-objetivo para el RCPSP robusto, considerando la minimización de la duración del proyecto y maximizando la robustez. Maghsoudlou et al. [28] resuelven el RCPSP multimodal con múltiples habilidades que minimiza la duración del proyecto, minimiza el costo total de la asignación de trabajadores y maximiza la calidad total de las actividades.
Por nivel de información disponible	Determinista	Se asume que se conoce y son constantes los valores de todos los parámetros del problema.
	No determinista (bajo incertidumbre)	El valor de uno o más parámetros es incierto. En la Sección 3.2 se profundiza esta variante del RCPSP.

en heurísticas, pero con estrategias de nivel superior que le permiten escapar de óptimos locales [23, 24]. Por ello, las metaheurísticas tienden a producir soluciones razonablemente cercanas al óptimo cuando sus diseños son apropiados [25].

En las últimas décadas ha habido un creciente interés en métodos que no se enfocan en una única metaheurística tradicional, por el contrario, combinan componentes algorítmicos que dan lugar a las llamadas metaheurísticas híbridas. Estos métodos se dividen en dos categorías: la primera es la mezcla de varias metaheurísticas, y la segunda combina una metaheurística con otras técnicas de investigación operativa, como la programación con restricciones o los métodos de búsqueda de árboles [29]. Pellerin et al. [30] determinó mediante un análisis comparativo las 36 metaheurísticas híbridas más eficientes para resolver el RCPSP. Se concluyó que 18 de los 36 métodos fueron híbridos basados en algoritmos poblacionales, y entre estos, 12 fueron basados en Algoritmos Genéticos (GA), convirtiéndolos en los métodos de solución más populares y eficientes.

3.2 RCPSP bajo incertidumbre

Para abordar el RCPSP bajo incertidumbre la literatura ha definido cinco enfoques principales: Programación Reactiva, Programación Estocástica, Programación Difusa, Programación Proactiva (Robusta) y Análisis de Sensibilidad [31]. Sin embargo, algunos autores como Brcic et al. [25], tienen en cuenta otro enfoque conocido como la programación predictiva, que ignora la incertidumbre y aproxima el RCPSP bajo incertidumbre a su versión determinística, lo que resulta poco efectivo ya que el proyecto puede retrasarse, sobrepasar el presupuesto o incluso resultar inviable. Los cinco enfoques principales para abordar el RCPSP bajo incertidumbre se describen brevemente a continuación:

- **Programación Reactiva**: este enfoque está basado en acciones de reparación a posteriori. La incertidumbre no interviene en la programación base del proyecto y, en caso de que se produzca un acontecimiento inesperado, se reprograman las actividades restantes [9]. Una de las estrategias para la programación reactiva es la *reactive effort*, que se basa en técnicas sencillas para reparar rápidamente la programación [31]. Un ejemplo de esta técnica es *right shift rule*, en la que las actividades afectadas por una disrupción se adelantan en el tiempo sin alterar su posición. Por otro lado, otras técnicas implican la reorganización completa de las actividades, a estos enfoques se les denomina *full rescheduling*.

  Algunos autores han desarrollado medidas para cuantificar la diferencia entre la programación del proyecto inicial y la programación generada posterior a un acontecimiento inesperado, con el fin de verificar la estabilidad de la programación frente a imprevistos. Una medida de estabilidad para el enfoque de programación reactiva es la *minimum perturbation strategy*, que se basa en métodos exactos que minimizan la suma ponderada de las diferencias absolutas entre los tiempos de inicio de cada actividad en la programación reparada y la programación inicial [32]. En otros trabajos, la medida de estabilidad es minimizar la cantidad de actividades modificadas [33].

- **Programación Estocástica**: este enfoque considera la naturaleza estocástica de las variables y tiene como objetivo minimizar la duración total esperada del proyecto, donde las duraciones de las actividades están dadas por un vector aleatorio o distribuciones

de probabilidad [8, 31]. Su principal desventaja es el alto costo computacional que requiere para resolver los modelos resultantes y la necesidad de suficientes datos para las pruebas de bondad de ajuste.

- **Programación Difusa**: una forma de convertir un problema incierto en un problema determinista es reemplazando los parámetros inciertos por *fuzzy numbers*. Haciendo uso de conjuntos difusos, donde utiliza una función de pertenencia que, generalmente, cuantifica el grado de incertidumbre en una escala normalizada de 0 a 1, donde 0 indica la confianza mínima y 1 la confianza máxima que se tiene sobre un valor de parámetro incierto [34]. Una de las principales ventajas de este enfoque es que los parámetros inciertos no deben seguir funciones de distribución estadística, lo que es útil cuando los datos históricos son escasos, y que el tamaño del modelo resultante es viable incluso con el aumento del número de parámetros inciertos [34, 35]. Sin embargo, una desventaja es que las reglas de combinación de funciones de pertenencia se basan en un enfoque min-max, que no es robusto, y por otro lado, estas reglas asignan la misma importancia a todos los valores que se van a combinar [36].

- **Programación Proactiva (o Robusta)**: las técnicas de optimización robusta han sido muy estudiadas en los últimos años por su eficacia y aplicabilidad, pues han resultado exitosas para abordar el URCPSP [37, 38]. Por ello, será foco de estudio en esta investigación. El enfoque robusto se centra en realizar desde el inicio una programación base que sea capaz de absorber posibles perturbaciones mediante una holgura de tiempo y recursos [39]. Debido a que las principales perturbaciones consideradas en la programación de proyectos surgen del cambio en la duración de las actividades, la literatura ha identificado dos enfoques principales de robustez: la robustez de calidad, que se refiere a la insensibilidad de la duración del proyecto ante las perturbaciones, y la robustez de solución, que se refiere a la insensibilidad de los tiempos de inicio de las actividades ante las perturbaciones [40, 41]. Con el fin de evaluar qué tan robusta es una solución, se han desarrollado diversos indicadores conocidos como medidas de robustez. Este tema se profundiza en la siguiente subsección.

3.3 Medidas de robustez

En la sección 3.3.1 se analizarán las principales métricas de robustez propuestas en la literatura para el URCPSP, donde se incluyen diversos enfoques para cuantificar la capacidad de resistir perturbaciones, además de identificar los vacíos existentes en la literatura. Posteriormente, en la sección 3.3.2, se explorará la aplicación de la robustez en sistemas eléctricos (base para la propuesta de este paper), detallando la métrica basada en la entropía propuesta por Koç et al. [42], y se presentarán las similitudes estructurales y dinámicas de los sistemas eléctricos con el URCPSP.

3.3.1 Robustez en el URCPSP

Las medidas de robustez permiten conocer qué tan capaz es una solución de absorber la variabilidad o perturbaciones de un sistema. La métrica más usada para el URCPSP es la función de coste de estabilidad, definida como la suma ponderada de las desviaciones absolutas esperadas entre los tiempos de inicio previstos de las actividades en la programación base y

sus tiempos de inicio realizados durante la ejecución real del programa [43]. Sin embargo, esta medida no tiene en cuenta características claves del URCPSP, como por ejemplo la estructura de la solución. Como esta, se han propuesto diferentes métricas en la literatura, a continuación se resumen las más relevantes:

- Ma et al. [39] abordan el Project Scheduling Problem with Flexible Resources (PSPFR), también llamado Multi-Skill Resource-Constrained Project Scheduling Problem (MSRCPSP). Este considera que los recursos cuentan con un conjunto de habilidades requeridas por las actividades. Cada recurso solo puede ejecutar una habilidad a la vez pero pueden cambiar la habilidad seleccionada en momentos discretos. Los autores miden la robustez como la suma ponderada de los Time Buffer (TB) de todas las actividades. Un TB se refiere al tiempo de reserva asignado con respecto a la holgura de una actividad. Dicha holgura se puede definir como Free Slack (FS) o Total Slack (TS). La primera se refiere a la cantidad de tiempo en el que una actividad puede moverse sin retrasar el tiempo de inicio más temprano de sus sucesores inmediatos; la segunda se refiere al tiempo en el que una actividad puede moverse sin retrasar el tiempo de finalización de todo proyecto [44]. En el caso de los autores, los TB se asignan con respecto a las FS, y su medida de robustez se muestra en la ecuación (9).

$$\sum_{n=1}^{n} w_i \cdot TB_i \tag{9}$$

  Donde w_i denota el coste marginal de desviar el tiempo de finalización de la actividad i de su tiempo de finalización previsto en el calendario base. Este coste puede considerarse como el impacto negativo del retraso en los sucesores inmediatos y transitivos de la actividad i. TB_i denota el TB, que puede absorber las perturbaciones y evitar que el impacto causado por estas se propague por la programación base.

- Drezet and Billaut [45] resuelven un URCPSP en el que las actividades del proyecto requieren únicamente recursos humanos, con una o varias habilidades, y los requerimientos de las actividades cambian a lo largo del tiempo. Se proponen dos medidas de robustez combinadas linealmente entre sí para obtener una robustez total. La primera medida (ρ_S^P) considera información sobre los empleados. Para ello, se define el $workload_i$ como la carga laboral del empleado i, p_{ind_i} como la probabilidad de ausencia del empleado i, y d_{ind_i} como la duración máxima de ausencia del empleado i; este indicador se calcula con la expresión (10).

$$\rho_S^P = \min_{I \in I} -(workload_i + d_{ind_i}) \times p_{ind_i} \tag{10}$$

  Por otro lado, la segunda medida de robustez (ρ_S^A) considera la información de las actividades. Se define $slack_j$ como la holgura de la actividad j, p_{all_j} como la probabilidad de ocurrencia de perturbación de la actividad j, y d_{all_j} como la duración máxima de la perturbación de la actividad j; este indicador se calcula con la expresión (11).

$$\rho_S^A = \min_{j \in J}(slack_j - d_{all_j}) \times p_{all_j} \tag{11}$$

En ambos casos, se considera como medida de robustez aquella actividad (empleado) que presenta la mayor disrupción, definida por tres elementos: la probabilidad, el máximo tiempo de la perturbación (ausencia) y el valor nominal de la holgura (carga laboral). Finalmente, la expresión para calcular la robustez total se muestra en la ecuación (12). Donde el parámetro α se puede interpretar como la importancia entre la robustez entre empleados y actividades.

$$\rho_S = -(\alpha \cdot \rho_S^P + (1-\alpha) \cdot \rho_S^A), \qquad \text{This function has to be minimized} \tag{12}$$

- Wang et al. [14] abordan un Resource-Constrained Multi-Project Scheduling Problem (RCMPSP) en el que un conjunto de proyectos, cada uno compuestos por un conjunto de actividades a realizar, debe ejecutarse simultáneamente, compitiendo por el uso de los recursos renovables. Además, cada proyecto define una fecha de lanzamiento (fecha de inicio del proyecto) y una fecha de finalización deseada [46]. Los autores introducen dos medidas de robustez en las que comparan el rendimiento determinista (utilizando el valor del parámetro c de la distribución triangular como duración de las actividades) con el rendimiento observado al simular las duraciones de las actividades estocásticas. Las medidas de robustez se presentan en las ecuaciones (13) y (14):

$$R1 = \frac{1}{|L|} \frac{\sum_{l \in L} SAD_l - AD_l}{AD_l} \tag{13}$$

$$R2 = \frac{\max_{l \in L} SAD_l - \max_{l \in L} AD_l}{\max_{l \in L} AD_l} \tag{14}$$

  Donde AD_l es la duración determinista del proyecto l, SAD_l es la duración estocástica del proyecto l, y L es el conjunto de subproyectos. La primera métrica ($R1$) mide la desviación relativa media de la duración del subproyecto simulado en comparación con la duración prevista del proyecto determinista. Por otro lado, la segunda métrica de robustez ($R2$) se centra en la robustez del multiproyecto completo, considerando únicamente los valores de los extremos.

- Hazır et al. [44] abordan el Discrete Time/Cost Trade-off Problem, que es un problema de programación de actividades multimodal. Los autores proponen nueve medidas de robustez basadas en holguras y las comparan mediante experimentos computacionales y medidas de rendimiento. Para el análisis experimental, utilizan simulación Monte Carlo y la distribución lognormal para generar un conjunto aleatorio de duraciones de las actividades. Una vez simulados los proyectos, se seleccionan las medidas de robustez con mayor correlación con las medidas de rendimiento. En este caso, se tuvieron en cuenta dos medidas de rendimiento: la proporción de réplicas en las que el proyecto finaliza antes del plazo establecido y el retraso medio en el plazo de finalización del proyecto.

  Se encontró que la medida de robustez con mayor correlación fue la RM_9, seguida por la RM_3, concluyendo que estas fueron las mejores medidas de robustez para lograr buenas estimaciones de una programación robusta. Las medidas de robustez se muestran en las ecuaciones (15) y (16).

$$RM_9 = 100 \cdot \frac{\delta - C_{(n+1)}}{\delta} \tag{15}$$

Donde $C_{(n+1)}$ denota la fecha de finalización del proyecto y δ el plazo de finalización del mismo. En esta medida de robustez el tamaño de la holgura se determina por la cantidad de tiempo en la que se puede retrasar la fecha de finalización del proyecto sin superar el plazo δ. De este modo, no se asignan holguras entre actividades individuales, sino que se agrega un factor de seguridad al final de proyecto.

$$RM_3 = \sum_{i=1}^{n} NS_i \cdot TS_i \tag{16}$$

Donde NS_i es el número total de sucesores de la actividad i y TS_i es la holgura total de la actividad i. Esta medida de robustez es una adaptación de la propuesta por Chtourou and Haouari [47]. El aporte de los autores radica en la utilización de la holgura total y el número total de sucesores de una actividad, en lugar del número de sucesores inmediatos. Esta medida de robustez busca asignar mayores holguras a las actividades con un mayor número de sucesores, ya que se supone que son las actividades con mayor probabilidad de retrasar la finalización del proyecto.

- Ma et al. [41] proponen un marco de trabajo dividido en dos partes. La primera parte consiste en la propuesta de una función de utilidad de holgura, en la que se adopta suavizamiento exponencial, cuyo propósito es otorgar mayor importancia a los primeros instantes de tiempo de la holgura de cada actividad. Ellos sugiren que la primera holgura es la más beneficiosa para la solidez de la programación del proyecto. La medida de robustez se expresa en la ecuación (17)

$$Robu = \sum_{i=1}^{n} W_i \sum_{j=1}^{FS_i} e^{\lambda \cdot j} \tag{17}$$

Donde W_i es la importancia de la actividad i, n es el número total de actividades, FS_i es la cantidad la holgura libre de la actividad i, λ es un parámetro no positivo y j es la unidad de tiempo de la holgura.

La segunda parte consiste en la introducción de tres parámetros para calcular la importancia de las actividades (W_i). Los autores proponen dos versiones (ecuaciones (18) y (19)).

$$\text{Versión 1} : W_i = w_i + \varphi_i \sum_{j \in S_i^*} w_j, \qquad 0 \leq \varphi_i \leq 1 \tag{18}$$

$$\begin{aligned} \text{Versión 2} : W_i = \varphi_i \sum_{j \in S_i^*} w_j, \qquad & 0 \leq \varphi_i \leq 1 \text{ para } (i,n) \in A, \\ & \varphi_i = 1 \text{ para } (i,n) \in A \end{aligned} \tag{19}$$

Donde S_i^* es el conjunto total de sucesores (directos e indirectos) de la actividad i, φ_i es el impacto medio del retraso de la actividad i, que se refiere al porcentaje promedio del número de sucesores totales que se verán afectados en caso de que la actividad i sufra un retraso, y w_j es la importancia inicial de los sucesores que pertenecen al conjunto S_i^*.

- Tian et al. [48] investigaron cómo generar una programación robusta para el discrete time/resource trade-off problem con una fecha de entrega bajo incertidumbre. Para ello, adoptan nueve medidas de robustez, que se consideran como función objetivo en un modelo de programación lineal cuyo valor debe ser maximizado. Posteriormente, se evalúa y compara la eficiencia de las medidas de robustez mediante las siguientes medidas de rendimiento:

  – Average Project Length (APL): es el promedio de tiempo requerido para finalizar el proyecto cumpliendo las restricciones de precedencia y recursos (ecuación (20)).

  $$APL = \frac{\sum_{t=1}^{N} S_{nt}}{N} \tag{20}$$

  Donde S_{nt} indica la hora de inicio de la actividad final ficticia en la simulación t y N es el número de simulaciones.

  – The Timely Project Completion Probability (TPCP): es la probabilidad de terminar el proyecto dentro del plazo previsto (ecuación (21)).

  $$TPCP = P(S_{nt} < duedate) \tag{21}$$

  – The Stability Cost (SC): es el costo de penalización por desviarse de las horas de inicio previstas (ecuación (22)).

  $$SC = \sum_{t=1}^{N} \sum_{i=1}^{n} w_i |S_{it} - s_i| \tag{22}$$

  Donde s_i es la hora de inicio prevista de la actividad i, S_{it} es la hora de inicio de la actividad i en la simulación t y w_i es el peso de la penalización de la actividad i, que se determina aleatoriamente a partir de una distribución uniforme entre 1 y 10. El peso de la actividad final n se establece en tres niveles: 50, 75 y 100.

Comparando estas métricas de rendimiento, los autores identificaron como las tres medidas de robustez más eficientes aquellas presentadas en las ecuaciones (23), (24) y (25).

$$RM6 = \sum_{i=1}^{n} \sum_{j=1}^{FS_i} \mathrm{e}^{-j} \tag{23}$$

$$RM7 = \sum_{i=1}^{n} CIW_i \sum_{j=1}^{FS_i} \mathrm{e}^{-j} \text{ donde } CIW_i = w_i + \sum_{j \in Succ_i^*} w_j \tag{24}$$

$$RM8 = \sum_{i=1}^{n} CIW_i \sum_{j=1}^{FS_i/Dur_i} \mathrm{e}^{-j} \text{ donde } CIW_i = w_i + \sum_{j \in Succ_i^*} w_j \tag{25}$$

La ecuación (23) es una adaptación de la medida propuesta por Hazır et al. [44] y es una alternativa que evita la inflación innecesaria de las holguras de algunas actividades. En lugar de asumir el mismo rendimiento para cada unidad de holgura, utiliza una función que asigna rendimientos decrecientes por unidad adicional de holgura libre FS_i de la actividad i. La ecuación (24) también es una adaptación de la medida propuesta por Lambrechts et al. [49], representa la utilidad de la holgura libre decreciente ponderada por el peso de inestabilidad acumulada CIW_i, donde $Succ_i^*$ son los sucesores, y w_i es generado aleatoriamente a partir de una distribución uniforme entre 0 y 1. Finalmente, la ecuación (25) es otra adaptación de la medida propuesta por Hazır et al. [44], donde $Dur_i = workcontent_i/resource_i$; esta expresión evalúa la holgura de las actividades con respecto a su duración; a mayor relación holgura/duración, mayor será su capacidad para evitar un retraso.

En la revisión de la literatura se han identificado varios vacíos de investigación sobre la robustez en la programación de proyectos. Ma et al. [39] asignan mayores buffers de tiempo a las actividades con mayores costos marginales, lo que no siempre es eficiente, debido a que las actividades más costosas no necesariamente tienen mayor variabilidad. Drezet and Billaut [45] miden la robustez basándose en la actividad más vulnerable ante las perturbaciones; sin embargo, no es adecuado evaluar la robustez de un sistema completo considerando solo una actividad, y su escala de valoración podría asignar una robustez igual a actividades con implicaciones diferentes, por ejemplo, a una actividad con una perturbación de 2 unidades de tiempo y 50% de probabilidad de ocurrencia y a otra con una perturbación de 1 unidad de tiempo y 100% de probabilidad de ocurrencia. Wang et al. [14] consideran solo el makespan, omitiendo factores como holguras, sucesores y recursos, lo que limita la precisión de su medida. Por otro lado, la medida $RM9$ propuesta por Hazır et al. [44] aunque efectiva, es similar a la de Wang et al. [14], y la medida $RM3$ considera aspectos importantes como los sucesores y la holgura, pero otorga la misma importancia a todas las unidades de tiempo de la holgura y asume que mayor holgura implica mayor robustez, sin considerar un límite máximo de una holgura necesaria. Finalmente, Ma et al. [41] otorgan mayor importancia a los primeros instantes de tiempo de las holguras, lo que puede no ser eficiente ya que si una actividad presenta perturbaciones largas, todas las unidades de tiempo de la holgura serán igualmente importantes hasta que el fallo se repare. En resumen, se han propuesto numerosas medidas de robustez en los últimos años. Sin embargo, la eficacia de estas medidas sigue siendo discutible, ya que aún no se ha determinado cuál de ellas es la más efectiva para la programación proactiva bajo incertidumbre. Además, estos vacíos indican la necesidad de desarrollar medidas de robustez más integrales. Este proyecto propone una medida de robustez holística, que agrupa los principales componentes del URCPSP: la estructura de las precedencias, el uso de recursos renovables por instantes de tiempo, y las holguras de las actividades generadas por una programación factible.

3.3.2 Robustez en sistemas eléctricos

La robustez no es un concepto exclusivo para problemas de programación de tareas. Este ha sido utilizado en diferentes áreas de investigación, como la microbiología [50], la ingeniería estructural [51], el diseño y desarrollo de turbinas de corrientes marinas [52], la aerodinámica [53], entre otros. Dentro de estas áreas destacan los sistemas eléctricos, debido a que el alto riesgo de disrupciones aleatorias ha llevado a la propuesta de indicadores de robustez. Una de las fallas más comunes son las denominadas fallas en cascada, que han sido ampliamente estudiadas por diversos autores [54, 55, 56]. Estas fallas ocurren cuando una sobrecarga o un componente defectuoso provoca fallos sucesivos en otros componentes conectados. De manera análoga, esta situación se presenta en el URCPSP, donde una actividad que demora más tiempo del estimado puede afectar en cadena el tiempo de inicio de sus sucesores, impactando así el makespan del proyecto.

Koç et al. [42] propuso una medida de robustez para sistemas eléctricos basada en la entropía. Esta medida sugiere que, a mayor entropía en una red eléctrica, los flujos sobre las líneas se distribuirán de manera más homogénea. Esto aumenta la robustez del sistema, ya que incrementa su capacidad para soportar flujos adicionales en las líneas en caso de que un nodo falle, permitiendo que el sistema no colapse. Esta métrica de robustez considera tanto las características topológicas (la interconexión de componentes, el número y tipo de nodos, y enlaces) como la dinámica de flujos (la distribución y nivel de carga). De manera similar, en el URCPSP, las características topológicas corresponden a los nodos (actividades y sus características como duración y requerimiento de recursos) y arcos (relaciones de precedencia), mientras que la dinámica de flujos podría referirse a las decisiones inherentes a la secuenciación de las actividades (como la asignación de recursos, establecer el tiempo de inicio y fin de cada actividad, etc). A continuación se detallan los dos elementos principales para calcular esta métrica de robustez en sistemas eléctricos:

1. **Robustez nodal**: estima la capacidad de un nodo para resistir fallos por sobrecargas. Para calcular la robustez nodal se recurre a los conceptos de entropía, que involucra que tan homogénea es la distribución de los flujos en las líneas, y tolerancia de la red, que involucra la capacidad de cada uno de los nodos. La entropía de la distribución de flujos de un nodo se presenta en la ecuación (26).

$$H_j = \sum_{i=1}^{L} p_i \log_{10}(p_i), \qquad \forall j \in \{1, \ldots, N\} \tag{26}$$

   Donde L es el número de flujos de salida de un nodo j, N es número total de nodos del sistema, y p_i es la normalización de la magnitud de los flujos de salida f_i. Estos últimos se calculan mediante la ecuación (27).

$$p_i = \frac{f_i}{\sum_{k=1}^{L} f_k}, \qquad \forall i \in \{1, \ldots, N\} \tag{27}$$

   Por otro lado, la ecuación (28) representa el cálculo de la tolerancia de la red denominado α_i.

$$\alpha_i = \frac{1}{LL_i}, \qquad \forall i \in \{1, \ldots, N\} \tag{28}$$

Donde LL_i es el nivel de carga de una línea de flujo i (nodo), que se calcula como se muestra en la ecuación (29).

$$LL_i = \frac{C_i}{L_i} \qquad \forall i \in \{1, \ldots, N\} \tag{29}$$

Donde C_i es la capacidad de carga de la línea i y L_i es la carga de la línea i.

Finalmente, Koç et al. [42] proponen la ecuación (30) para estimar la robustez nodal, mediante la combinación de la entropía nodal y la tolerancia de la red. El signo menos $(-)$ se añade para conservar la magnitud de la entropía, que resulta negativa debido a la aplicación del logaritmo al rango de valores de la normalización de los flujos P_i.

$$R_j = -\sum_{i=1}^{L} \alpha_i p_i \log_{10}(p_i), \qquad \forall j \in \{1, \ldots, N\} \tag{30}$$

2. **Importancia nodal**: es una medida de centralidad que permite determinar el peso relativo de cada nodo y por tanto la criticidad de cada uno. La importancia nodal depende de los flujos eléctricos distribuidos por cada nodo y se calcula como se presenta en la ecuación (31).

$$\delta_j = \frac{P_j}{\sum_{k=1}^{N} P_k}, \qquad \forall j \in \{1, \ldots, N\} \tag{31}$$

Donde δ_j se define como la importancia del nodo j, y P_j representa el parámetro de la potencia total distribuida por el nodo j.

Posterior al cálculo de la robustez nodal (R_j) y de la importancia nodal (δ_j), se establece que la robustez del sistema eléctrico se representa mediante la ecuación (32).

$$R = \sum_{j=1}^{N} R_j \delta_j \tag{32}$$

Dado que esta medida de robustez ya ha sido adaptada con éxito por Maya-Rodríguez et al. [57] en sistemas logísticos debido a su analogía topológica, y considerando la similitud en términos de características topológicas y dinámica de flujos entre los sistemas eléctricos y el URCPSP, en este paper se propone adaptar la métrica de robustez propuesta por Koç et al. [42] para el URCPSP.

4 New robustness measure proposed for the RCPSP

Uno de los principales aportes de esta investigación consiste proponer una nueva medida de robustez para el URCPSP, para esto se toman los conceptos de la robustez nodal e importancia nodal de las medidas de robustez en sistemas eléctricos [42]. A continuación se detalla cada elemento de esta nueva medida, desde el símil que tienen los sistemas de programación de tareas, hasta un ejemplo numérico.

En el URCPSP, la robustez está estrechamente relacionada a las holguras, ya que, en caso de un imprevisto, las actividades podrían desplazarse en el tiempo utilizando dichas holguras sin afectar el makespan (o lo menos posible), lo que incrementa la robustez del sistema. No obstante, una mayor holgura no siempre implica mayor robustez, ya que una asignación excesiva de holguras podría no ser requerida dada la naturaleza del riesgo o afectación de una actividad, y solo se incrementaría el makespan del proyecto. Por ello, y a diferencia de otros trabajos, se definirá lo que en esta investigación denominaremos Expected Slack (ES), esta representa la cantidad de tiempo adicional que se espera tener en un actividad para garantizar que, ante riesgos imprevistos, no se altere la programación del proyecto considerando un determinado Service Level (SL). Sin perdida de generalidad, se considera un $SL = 95\%$ en esta investigación.

Para adaptar la métrica de robustez utilizada en sistemas eléctricos al URCPSP, se parte del supuesto de que una mayor entropía en un sistema está asociada a una mayor robustez [42]. En consecuencia, se sugiere que, a mayor entropía, mayor podría ser el cumplimiento de la holgura esperada en caso de un evento fortuito para cada actividad y, por ende, mayor es la robustez.

Con este contexto, a continuación se describe la adaptación de la métrica de robustez en sistemas eléctricos al URCPSP dividida en tres subsecciones: primero se define la robustez nodal (Sección 4.1), luego se introduce la importancia nodal (Sección 4.2) para, finalmente, concluir con la robustez del sistema (Sección 4.3).

4.1 Robustez nodal

Retomando lo mencionado en la Sección 3.3.2, en los sistemas eléctricos, la robustez nodal mide la capacidad de un nodo para resistir fallos por sobrecargas. De manera análoga, en el URCPSP, la robustez nodal evalúa la capacidad de una actividad para resistir perturbaciones. Tanto en sistemas eléctricos como en el URCPSP, la robustez nodal se basa en los conceptos de entropía y tolerancia nodal que se presentan en las secciones 4.1.1 y 4.1.2, respectivamente. Posterior al calculo de estos, se presenta la robustez nodal en la ecuación (41). Estos cálculos son realizados unicamente para las actividades sujetas a riesgo (subconjunto denominado $IR : IR \subset I$), pues no tendría lógica establecer el cumplimiento de una holgura esperada a una actividad que no está sujeta a riesgos.

4.1.1 Entropía

En los sistemas eléctricos, la entropía mide la distribución de los flujos de energía entre los nodos del sistema, reflejando cuán uniformemente se reparte la energía. En lugar de medir el desorden en términos de caos o irregularidad, la entropía en este contexto se centra en la dispersión de la energía. Cuando los flujos de energía se distribuyen uniformemente entre

todos los nodos, la dispersión es mayor, lo que se traduce en una mayor entropía y, por ende, en un sistema más robusto [42]. En otras palabras, una mayor entropía indica que los flujos están distribuidos de manera homogénea, lo que es beneficioso ya que permite al sistema manejar fallos de manera más eficaz sin sobrecargar otros nodos, aumentando así su robustez. De forma análoga, en el URCPSP, la entropía no mide el grado de caos, sino la distribución de las holguras en las actividades sujetas a riesgo a lo largo de la programación del proyecto. Sin embargo, en este contexto no se busca una distribución homogénea, sino una distribución inteligente de las holguras. Es decir, las holguras deben asignarse según las necesidades de cada actividad: aquellas que no están sujetas a riesgos no requieren holguras, mientras que las actividades con mayor riesgo necesitan holguras en proporción a su variabilidad (Expected Slack). En este caso, un valor alto de entropía refleja una distribución más inteligente de las holguras, lo que brinda al proyecto mayor flexibilidad para adaptarse a cambios y retrasos sin afectar significativamente la programación. Esto resulta en un sistema más robusto.

En la ecuación (33) se propone la expresión para el cálculo de la entropía nodal. A diferencia de los sistemas eléctricos, en el URCPSP no se considera la sumatoria sobre el conjunto de arcos (flujos) que salen de cada nodo, pues no hay explícitamente un flujo, por tanto la entropía se calcula para cada nodo (actividad) individualmente.

$$H_i = p_i \log_{10}(p_i), \qquad \forall i \in IR, \quad IR \subset I \tag{33}$$

En la robustez de sistemas eléctricos, p_i corresponde a la normalización de los flujos de salida, que representan la cantidad de energía transportada a través de una línea o componente de la red desde un nodo a otro. En el URCPSP, p_i se define como la normalización de las holguras. En ambos contextos, la robustez se mide según la capacidad de mantener la operación frente a perturbaciones. En redes eléctricas, la normalización de los flujos evalúa cómo se distribuye la energía de manera eficiente y equilibrada en el sistema, considerando la capacidad de las líneas. En el RCPSP, la normalización de las holguras evalúa cómo se distribuye el tiempo extra disponible (holguras) entre actividades para absorber retrasos sin afectar el makespan. En ambos casos, esta normalización permite medir la distribución de estos márgenes (energía o tiempo) y su impacto en la estabilidad del sistema. Adicionalmente, se propone una normalización de holguras dependiente del grado de importancia que esta tenga en la secuencia evaluada.

Así, para calcular p_i primero debemos calcular la importancia de la holgura de cada actividad i, para ello, se propone la función de holgura lineal por partes $SLF(i, S_i)$ para cada actividad i, donde S_i representa la holgura obtenida por la secuencia de dicha actividad. La ecuación completa se muestra en la ecuación (34).

$$SLF(i, S_i) = \begin{cases} m_{1i} * S_i, & S_i \leq ES_i \quad \forall i \in IR, \quad IR \subset I \\ m_{2i} * S_i + b_{2i}, & S_i > ES_i \quad \forall i \in IR, \quad IR \subset I \end{cases} \tag{34}$$

$SLF(i, S_i)$ se define como una función por partes que evalúa la importancia de la holgura de la actividad i, en una escala de 0 a 100, considerando el grado en que cumple su holgura esperada (ES_i), o más de 100 una vez se supere. Esta función se caracteriza por tomar la forma $m_{1i} * S_i$ cuando la holgura de la actividad i no alcanza el valor de la ES_i y adopta la

forma $m_2 * S_i + b_2$ en caso contrario. Esta función se define por partes debido a que, una vez se alcance ES_i, será posible reparar la disrupción sin alterar la programación propuesta, y en este caso, la importancia de la holgura alcanzará un valor del 100% (y cada unidad de tiempo de la holgura toma el mismo valor de importancia). En caso de que la S_i exceda a ES_i, solo denotará un tiempo adicional que, basados en el comportamiento esperado de los riesgos y el nivel de servicio, probablemente no será requerido o significativo. Por lo tanto, la importancia de holgura, una vez alcanzada la ES_i, crecerá con una pendiente menos pronunciada. En esta investigación, se considerará un crecimiento del 0.001% por holgura excedente, valor que puede ser ajustado según la naturaleza del problema a resolver. En resumen, el valor de la holgura en una actividad i (generada por una solución completa al problema) implica una importancia significativa que crece considerablemente cuando el valor se encuentra entre el intervalo $[0, ES_i]$, a partir del cual su importancia continúa incrementándose, pero con mucha menos intensidad (menor pendiente). En la ecuación (34) m_{1i} y m_{2i} representan las pendientes de las líneas, y b_{2i} corresponde a la ordenada al origen, que es el punto de intersección con el eje y.

Para determinar ES_i se debe considerar la naturaleza de los riesgos a los que está sometido la actividad. Así, asumiendo normalidad en los tiempos extra que generan estos riegos al tiempo de procesamiento a una actividad i, se estima para cada riesgo el ES con base en un nivel de confianza dado; en esta investigación se usará el 95%. Posteriormente, se elige la ES más alta entre los distintos riesgos evaluados. Esto se hace con el fin de que, en caso de que uno o más riesgos se materialicen, la holgura sea suficiente para reparar el fallo que causó el mayor retraso de la actividad sin afectar la programación en el 95% de las veces, asumiendo simultaneidad en las reparaciones.

Example 1. Supongamos un fragmento de una solución de un URCPSP como se aprecia en la Fig. 3. Esta muestra una programación de actividades factible, donde se tiene en cuenta las restricciones de precedencia y la disponibilidad de los recursos, también se muestra el consumo del recurso 1 de cada actividad en la Tabla 2, y se quiere conocer la función de holgura lineal para la actividad $i = 6$. Para este ejemplo se usará la notación Act_i, para referirse a una actividad específica i.

Primero, se calcula la holgura de Act_6. Suponiendo que esta tiene como sucesora a Act_8, y no puede programarse en simultáneo con Act_9 y Act_{10}, porque excede la disponibilidad del recurso 1 (que es $W_1 = 8$), Act_6 podría desplazarse en una holgura de dos instantes de tiempo, en el intervalo $t = 3$ a $t = 5$. Supongamos además que Act_6 está sujeta a un riesgo que sigue una distribución normal con $\mu_6 = 2$ y $\sigma_6 = 1$. Para calcular $SLF(6, S_6)$ con un 95% de confianza, primero debemos determinar su holgura esperada $ES_6 = z_{0.95} * \sigma_6 + \mu_6 = 1.645 * 1 + 2 = 3.645 \approx 4$.

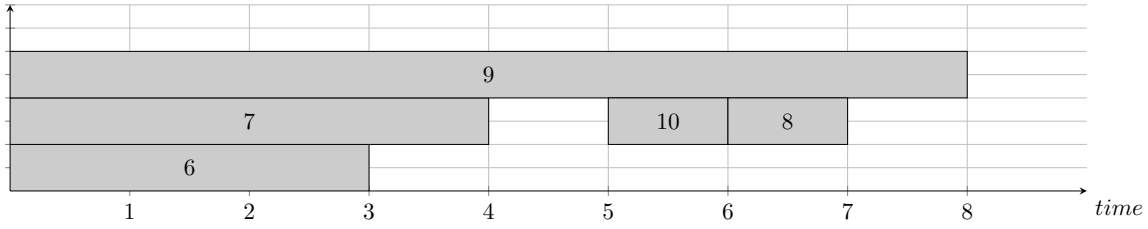

Figure 3: Diagrama de Gantt de un fragmento de solución factible para un ejemplo de un URCPSP.

$W_1 = 8$	
i	R_{i1}
6	3
7	2
8	1
9	3
10	5

Table 2: Disponibilidad del recurso 1 para un ejemplo de un URCPSP.

Resumiendo, se espera que Act_6, sujeta a un riesgo aleatorio, cuente con una holgura de 4 unidades de tiempo para que, en el 95% de los casos, no afecte la programación del proyecto. A continuación, se procede a construir la función lineal por tramos ($SLF(6, S_6)$) utilizando los puntos: (0.0%), que indica que si no hay holgura la importancia es cero; $(4, 100\%)$ que indica el cumplimiento de ES; y $(5, 100.001\%)$ que indica un exceso de holgura. La función completa se muestra en la Fig. 4.

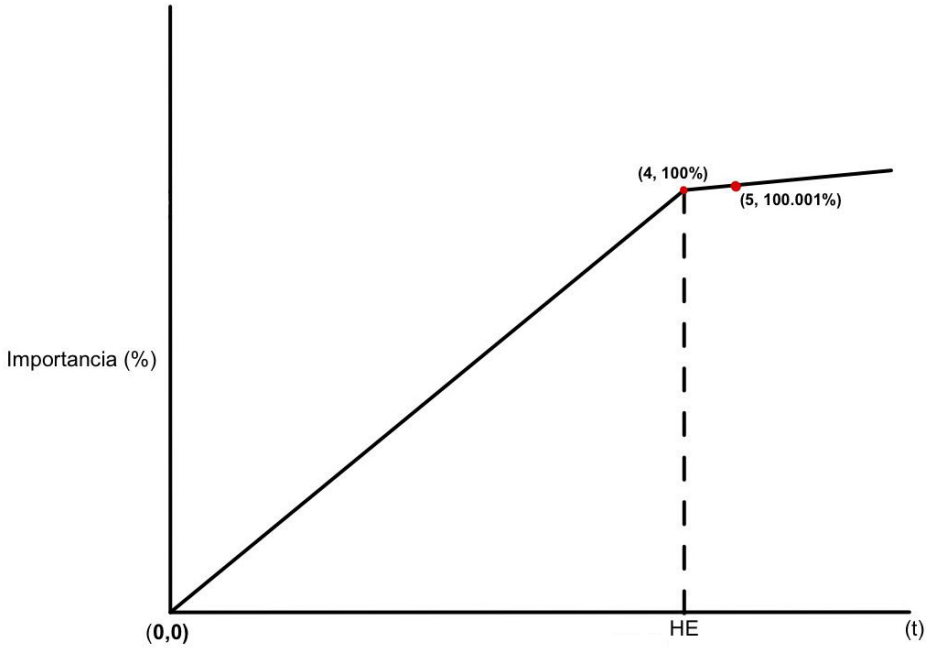

Figure 4: Ejemplo de la función por partes $SLF(6, S_6)$ que muestra la importancia de la holgura de la actividad 6.

Para el primer tramo de la ecuación $SLF(6, S_6)$ se describe una línea cuya pendiente es $m_{1,6} = 100/4 = 25$. En cuanto al segundo tramo de la función, la pendiente es $m_{2,6} = (100.001 - 100)/(5 - 4) = 0.001$, y el intercepto es $b_{2,6} = 100 - 0.001 * 4 = 99.996\%$. La ecuación (35) muestra la función completa.

$$SLF(6, S_6) = \begin{cases} 25 * S_6, & S_6 \leq 4 \\ 0.001 * S_6 + 99.996, & S_6 > 4 \end{cases} \tag{35}$$

Con lo anterior, calculamos la importancia de la holgura de Act_6 utilizando el primer tramo de la función por tramos, dado que en este ejemplo la Act_6 tiene una holgura de $S_6 = 2$. Así, la importancia es $SLF(6, S_6 = 2) = 25 * 2 = 50\%$.

Una vez definida la ecuación de la importancia de holgura $SLF(i, S_i)$, se procede a ajustar todos los valores a una escala común por medio de la normalización de las holguras p_i, que se define matemáticamente en la ecuación (36).

$$p_i = \frac{SLF(i, S_i)}{\sum_{i \in IR} SLF(i, S_i)}, \qquad \forall i \in I, \quad IR \subset I \tag{36}$$

Example 2. Continuando con el Ejemplo 1 de la Fig. 3, y asumiendo que se ha calculado la importancia de la holgura de todas las actividades sujetas a riesgos. Usando la ecuación (36), se normaliza la importancia de las holguras de cada actividad, y finalmente, se calcula la entropía nodal con la ecuación (33). Los cálculos de este ejemplo se muestran en la Tabla 3.

	SLF	P_i	H_i
$SLF(i=1, S_i=3)$	100,05	0,570	-0,139
$SLF(i=6, S_i=2)$	50	0,285	-0,155
$SLF(i=8, S_i=5)$	25,56	0,146	-0,122

Table 3: Normalización de la importancia de la holgura de las actividades y cálculo de entropía para actividades sujetos a riesgo.

4.1.2 Tolerancia nodal

En los sistemas eléctricos, el parámetro de tolerancia nodal está relacionado con la capacidad, que se refiere a la carga máxima que una línea puede soportar. En el URCPSP, la capacidad se asocia con la disponibilidad de recursos. Como se ha mencionado en otras secciones, la holgura representa el tiempo adicional disponible para una actividad sin afectar el makespan o a sus sucesores. Sin embargo, si los recursos no están disponibles durante ese tiempo adicional, la holgura pierde su valor práctico, ya que la actividad no podría programarse, aunque el tiempo lo permitiera. Asignar holgura a una actividad sin considerar la capacidad de los recursos puede llevar a que estos recursos se vean comprometidos en otras actividades en el futuro, lo que podría ocasionar retrasos imprevistos o la necesidad de reprogramar. Por lo tanto, un URCPSP es más robusto cuando cuenta con mayor disponibilidad de recursos. Así, la tolerancia nodal en el URCPSP se relaciona con la disponibilidad de los recursos y se propone su estimación en la ecuación (37).

$$\alpha_i = \frac{1}{LL_i}, \qquad \forall i \in IR, \quad IR \subset I \tag{37}$$

Donde LL_i se define como el promedio de la utilización de recursos por cada actividad i y se calcula a partir de dos conceptos: RU_{kij} y PRU_{ik}, que se detallan a continuación.

- RU_{kij} es la utilización del recurso k de todas las actividades $m \in I_j^{act}$, donde I_j^{act} representa el conjunto de actividades activas durante el instante de tiempo $t = j$ de la holgura S_i. Este valor se calcula con base en R_{mk} (el consumo del recurso k que la actividad m requiere para ejecutarse) y W_k (disponibilidad del recurso k), siguiendo con la nomenclatura definida en la Sección 3.2. La ecuación (38) muestra la función completa.

$$RU_{kij} = \frac{\sum_{m \in I_j^{act}} R_{mk}}{W_k}, \qquad \forall k \in K, \forall i \in IR, \forall j \in \{1, \ldots, S_i\}, \quad IR \subset I \tag{38}$$

- PRU_{ik} es el promedio del uso del recurso k en S_i (holgura de la actividad i). Se calcula como se muestra en la ecuación (39).

$$PRU_{ik} = \frac{\sum_{j \in \{1,\ldots,S_i\}} RU_{kij}}{S_i}, \qquad \forall i \in IR, \forall k \in K, \quad IR \subset I \tag{39}$$

Así, LL_i se calcula según la ecuación (40), que se interpreta como el promedio de los usos promedios de los recursos durante la holgura S_i.

$$LL_i = \frac{\sum_{k \in K} PRU_{ik}}{|K|} \qquad \forall i \in IR, \quad IR \subset I \tag{40}$$

Finalmente, en la ecuación (41) se muestra el calculo de la robustez nodal mediante la combinación de la entropía nodal y la tolerancia nodal.

$$R_i = -\alpha_i p_i \log_{10}(p_i), \qquad \forall i \in IR, \quad IR \subset I \tag{41}$$

Example 3. Continuando con el Ejemplo 2 de la Fig. 3, la tolerancia nodal se calcula de la siguiente manera: primero, determinamos la utilización del recurso $k = 1$ para las actividades activas en el primer instante de tiempo dentro de la holgura de Act_6 (entre $t = 3$ y $t = 4$), utilizando la Ecuación (38). Luego, repetimos el procedimiento para el segundo instante de tiempo dentro de la holgura, que va de $t = 4$ a $t = 5$. El procedimiento y los resultados obtenidos se presentan en la Tabla 4.

$RU_{k,i,j}$	**Cálculo**	**Resultado**
$RU_{1,6,1}$	$\frac{R_{9,1}+R_{7,1}}{W_1} = \frac{3+2}{8}$	0.625
$RU_{1,6,2}$	$\frac{R_{9,1}}{W_1} = \frac{3}{8}$	0.375

Table 4: Cálculo de $RU_{1,6,j}$ para el ejemplo de la Fig. 3

Esto indica que la utilización del recurso $k = 1$ es del 62.5% durante el primer instante de tiempo de la holgura de Act_6, y del 37.5% en el segundo instante de tiempo. A continuación, calculamos el promedio del uso del recurso $k = 1$ en la holgura de Act_6 (PRU_{ik}) con la ecuación (39). El cálculo se muestra en la ecuación (42).

$$PRU_{6,1} = \frac{RU_{1,6,1} + RU_{1,6,2}}{S_i} = \frac{\frac{5}{8} + \frac{3}{8}}{2} = 0.5 \tag{42}$$

Lo anterior se interpreta como que, en promedio, la utilización del recurso 1 en la holgura de Act_6 es del 50%. Ahora, para calcular LL_i, que es el promedio de la utilización de todos los recursos en la holgura de Act_6, debemos aplicar el procedimiento anterior para todos los recursos. En este ejemplo, se asume que existen dos recursos y que la utilización del recurso 2 es conocida con un valor de $PRU_{62} = 0.6$. El calculo de LL_6 correspondería a $(PRU_{6,1} + PRU_{6,2})/|K| = 0.55$.

Finalmente, se calcula la tolerancia nodal $\alpha_6 = 1/LL_6 = 1.82$ y la robustez nodal $R_6 = -\alpha_6 p_6 \log_{10}(p_6) = -1.82 * 0.285 \log_{10}(0.285) = 0.282$ con las ecuaciones (37) y (41), respectivamente.

4.2 Importancia nodal

En un proyecto, algunas actividades tienen un impacto significativo en el cronograma porque afectan a una gran cantidad de actividades sucesoras. Cuando una de estas actividades sufre un retraso, este retraso puede propagarse y afectar a todas las actividades sucesoras, lo que

podría comprometer el plazo final del proyecto, de manera similar a un fallo en cascada en un sistema eléctrico. Por otra parte, si una actividad con pocos o ningún sucesor se retrasa, su impacto en el cronograma global es menor, ya que no afectará significativamente a otras partes del proyecto en relación con actividades sucesoras pero sí podría impactar en el uso de los recursos. Así, el impacto de cada actividad en el URCPSP depende del número de sucesores que tenga, de la misma manera en que el impacto de un nodo en una red eléctrica depende de la cantidad de energía que distribuye a través de la red. Por lo anterior, la importancia nodal en el URCPSP (δ_i) se define en función de la cantidad de sucesores totales (TS_i) de cada actividad i. Así, se propone la ecuación (43) que determina el cálculo de la importancia nodal.

$$\delta_i = \frac{TS_i}{\sum_{m \in I} TS_m} \qquad \forall i \in IR, \quad IR \subset I \tag{43}$$

Example 4. Finalmente, continuando con el Ejemplo 3 de la Fig. 3 si se asume que Act_6 tiene como sucesor a Act_6, y que el numero de sucesores totales del sistema es 5, la importancia nodal de Act_6 (δ_6) es igual a $TS_6 / \sum_{m \in I} TS_m = 1/5 = 0.2$.

4.3 Robustez del sistema

Posterior al cálculo de la robustez nodal y de la importancia nodal, se calcula la robustez del sistema mediante la ecuación (44).

$$R = \sum_{i=1}^{IR} R_i \delta_i \tag{44}$$

5 Proposed model

Una vez definida la nueva medida de robustez (Sección 4), se requiere una metodología de solución que permita tanto encontrar soluciones factibles de alta calidad en relación al makespan, así como secuencias de programación de tareas base robustas. Para esto, en esta sección se proponen dos componentes: 1) un algoritmo de solución que integra esta nueva medida de robustez (Subsección 5.1), y 2) una metodología general que permite abordar el componente bajo incertidumbre del URCPSP (Subsección 5.2). La parametrización del algoritmo se encuentra en la subsección 6.3.

5.1 Proposed Genetic Algorithm

El Genetic Algorith (GA) es una metaheurística poblacional inspirada en la teoría de la evolución de Darwin de la supervivencia del más apto y fue introducido por Holland [58] en 1975. De forma general, un GA funciona de la siguiente manera: comienza con una población de soluciones codificadas (cromosomas) que tienen características específicas (genes). Estos individuos son evaluados mediante una función de aptitud para determinar los más aptos, es decir, qué tan buenos son según un indicador definido para un problema determinado. Los individuos con mejor función de aptitud tendrán mayores probabilidades de seleccionarse como padres. Estos últimos contribuirán para la generación de nuevas soluciones (descendientes)

mediante operadores de cruce, que permite combinar genes deseables de cada padre, y mutación, que introduce variabilidad a la población para evitar convergencia prematura. Este proceso es iterativo a lo largo de varias generaciones hasta alcanzar un criterio de parada. De esta manera, las mejores soluciones tendrán más probabilidades de reproducirse y transmitir sus características a las generaciones futuras, simulando el proceso de selección natural, donde las soluciones más adecuadas sobreviven y las menos adecuadas se eliminan. Los GA son bien conocidos en la literatura y han sido ampliamente usados en campos como producción [59], ruteo [60], predicción de datos meteorológicos [61], cadena de suministro [62], diseño estructural [63], salud [64], entre otros. Uno de los primeros autores en aplicar GA para el RCPSP fue Hartmann [65], desde entonces se ha perfeccionado su diseño y, como se mencionó en la Sección 3, el GA es uno de métodos de solución más populares y eficientes para resolver el RCPSP [30, 66]. En las subsecciones 5.1.1 a 5.1.8 se describe la metodología para el GA propuesto en esta investigación.

5.1.1 Encoding scheme

Un Encoding Scheme o Schedule representation es la codificación utilizada para representar una posible solución de un problema, garantizando que pueda ser interpretada y procesada por el algoritmo. En un GA, se transforma las soluciones originales, como calendarios o programaciones de tareas, en estructuras codificadas llamadas cromosomas.

En la literatura, las codificaciones más eficientes para el RCPSP son los esquemas de Activity List (AL) y Random Key (RK) [67]. Aunque no existen estudios recientes que comparen exhaustivamente estas codificaciones, Hartmann [65] determinó que la representación AL es la más adecuada para resolver el RCPSP. Por esta razón, se adopta esta representación en esta investigación. Una AL se compone por una secuencia de actividades factibles por precedencia, donde la posición establece el orden de prioridad de programación.

La Fig. 5 muestra un ejemplo de codificación bajo el esquema AL, donde las actividades se organizan según sus relaciones de precedencia. En este caso, las actividades se enumeran de $i = 1$ a $i = 10$, representando los extremos como actividades ficticias, y se ubican en posiciones de 0 a 9, respectivamente.

Figure 5: Ejemplo de una Activity List.

En el GA propuesto cada individuo (cromosoma) de la población está compuesto por una AL y un gen binario de dirección. Este gen puede adoptar dos valores: Forward cuando su valor es 0 y Backward cuando su valor es 1. La decodificación Forward consiste en utilizar directamente el orden de prioridad definido por la AL. Por otro lado, la decodificación Backward invierte el orden de la AL e intercambia los sucesores y predecesores de cada actividad, de manera que la primera actividad corresponde a la posición $I' + 1$ y la última a la posición

0. Este gen de dirección está basado en el conocido Forward-Backward Iteration (FBI), que ha demostrado ser eficaz y adecuado para mejorar la calidad de las soluciones en el RCPSP [68].

5.1.2 Decoding based on Serial Schedule Generation Scheme

Los Schedule Generation Schemes (SGS) son algoritmos que permiten construir un cronograma factible a partir de una representación codificada de una solución, como la AL. Esto implica transformar dicha representación en una programación válida, procesando cada actividad (o un grupo de ellas) en el orden establecido por la lista y respetando las restricciones de precedencia y disponibilidad de recursos.

Existen dos tipos de SGS principales para el RCPSP: esquema en serie y esquema en paralelo. El espacio de búsqueda de este último produce programaciones non-delay ajustadas, pero no siempre contiene la solución óptima, mientras que el esquema en serie produce programaciones activas, conjunto que siempre contiene la solución óptima [69]. En esta investigación se emplea el SGS serial, que programa las actividades de la AL lo más pronto posible según el orden de la lista, teniendo en cuenta las restricciones de precedencia y de consumo de recursos. El proceso se repite hasta que no quedan actividades por programar. Dado que las programaciones generadas por el SGS serial, en general, pueden ser no tan ajustadas como en el esquema paralelo, esto podría resultar en una mayor holgura en la planificación, y por tanto, en una programación más robusta frente a incertidumbres.

5.1.3 Initial Population

Un GA comienza con una población inicial, es decir, la primera generación de soluciones. El objetivo de esta es tener individuos suficientemente diversos para garantizar una fase de exploración en la búsqueda. Pellerin et al. [30] realizó una comparación de 111 investigaciones que implementaban metaheurísticas híbridas para resolver el RCPSP comprendidas entre los años 1 995 y 2 018, uno de los hallazgos fue que el método más usado para construir la solución inicial fue la generación aleatoria pero factible. Por ello, se optó por la generación de individuos completamente aleatorios para una población de tamaño POP.

El proceso de creación debe repetirse tantas veces como individuos se requieran. Para construir la AL de cada individuo, se inicia determinando un conjunto de actividades elegibles (Ele), teniendo en cuenta las restricciones de precedencia. Posteriormente, se asigna una probabilidad uniforme para cada actividad $i \in Ele$ y se selecciona aleatoriamente una. Así, de manera iterativa se seleccionan actividades, actualizando el conjunto Ele, debido a que por precedencias, cada actividad programada podría liberar nuevas actividades elegibles que se adicionan al conjunto Ele, cambiando la probabilidades de selección de las actividades (pero manteniéndolas uniformes). Este proceso iterativo se repite hasta que todas las actividades hayan sido programadas.

5.1.4 Fitness function

La función de aptitud cuantifica la calidad de una solución, permitiendo compararla con otras y determinar cuál es mejor. Esta función se usa en cada generación para evaluar cada cromosoma (individuo), y así proveer de mayor probabilidad de ser padres a aquellas con los mejores valores. La función de aptitud del GA propuesto es el makespan, la inclusión de la

robustez se detalla en la Sección 5.2.

5.1.5 Selection Operator

La selección es el proceso en el que se eligen los padres de una población, estos se cruzarán para crear descendencia y construir la siguiente generación. Si bien no existe una metodología de selección que supere a otra en todos los casos, la literatura recomienda tener un enfoque semielitista. En la metodología exitosa propuesta por Mendes et al. [70], uno de los padres se elige aleatoriamente entre los mejores individuos de la población (denominada TOP), mientras que el otro se elige aleatoriamente entre toda la población actual (incluidos los TOP). Se adoptó esta metodología de selección para el GA propuesto.

5.1.6 Crossover Operator

En el operador de cruce se producen uno o más descendientes utilizando el material genético de los progenitores, la idea es que los descendientes estén compuestos de información genética de los padres. En la literatura, los operadores de cruce de mayor relevancia para el RCPSP son one-point, two-point, uniform, peak, y magnet-based [67]. Sin embargo, el one-point crossover propuesto por Davis [71] y el two-point crossover propuesto por Hartmann [65] son los más usados [72]. En particular, Hartmann [65] demostró que el two-point crossover supera al one-point crossover en términos de desempeño para el RCPSP, siendo ampliamente utilizado en investigaciones recientes como las de He and Zhang [73], Li et al. [74], Liu et al. [75]; además, este operador ha mostrado ser eficiente en otros problemas de secuenciación, como el Job Shop [76, 77]. Por lo tanto, este será el operador adoptado en el GA de esta investigación.

En el Two-point crossover se seleccionan dos puntos de cruce a lo largo del cromosoma para la generación de dos descendientes [78]. Para aplicar este operador en el RCPSP, donde las soluciones suelen representarse como permutaciones (similares a la AL), es necesario un manejo especial para garantizar la factibilidad de las soluciones. El proceso inicia seleccionando aleatoriamente dos posiciones (puntos de cruce): $k_1, k_2 : k_1 < k_2$. Posteriormente, se toman las secciones definidas por estos puntos del primer padre, y luego se asigna al descendiente los genes respetando el orden en que aparecen en la madre. Es decir, el hijo hereda los genes desde la posición $\{0, \ldots, k_1\}$ y $\{k_2, .., I' + 1\}$ del padre, y los restantes, $\{k_1 + 1, ..., k_2 - 1\}$, son tomados en el orden de la madre pero sin repetir aquellos que ya se heredaron. Luego, se repite el procedimiento pero iniciando con la madre y usando los mismos puntos de cruce, generando así otro descendiente diferente. Este enfoque asegura que las soluciones resultantes sean válidas y factibles dentro del contexto del problema. En esta investigación se excluyen todas las posiciones de cruce que, aunque con valores diferentes, resulten descendientes iguales a algún padre. Adicionalmente, para evitar un resultado equivalente al one-point crossover se debe eliminar para k_2 la actividad correspondiente a la posición seleccionada k_1 y las actividades contiguas a ella $\{k_1 - 1, k_1 + 1\}$.

Example 5. La Fig. 6 muestra un caso no deseado en la aplicación del operador Two-Point Crossover. Se considera un proyecto con 10 actividades, donde las actividades 1 y 10 son ficticias, con posiciones de cruce $k_1 = 1$ y $k_2 = 3$. En este ejemplo, el cruce no genera variabilidad, ya que los descendientes resultan idénticos a sus respectivos progenitores: el primer descendiente (Children 1) es igual al padre y el segundo (Children 2) a la madre. Los

genes pintados de morado corresponden a los seleccionados del “Padre”, mientras que los de azul representan los genes tomados de la “Madre”.

A continuación se detalla la generación de los hijos: para el primer descendiente se conservan en orden los genes del “Padre” antes de $k_1 = 1$ y después de $k_2 = 3$, ambos incluidos. Luego, para el segmento restante (entre $k_1 = 1$ y $k_2 = 3$), se seleccionan los genes de la “Madre”. Sin embargo, para mantener la factibilidad del problema, solo se toman aquellos valores que no coincidan con actividades ya presentes en el descendiente. Dejando solo como actividad heredable la Act_5, generando un descendiente igual al padre. Para el segundo descendiente, se sigue el mismo procedimiento, pero intercambiando el orden de prioridad para heredar los genes: primero la madre y luego el padre. El descendiente generado es igual a la madre.

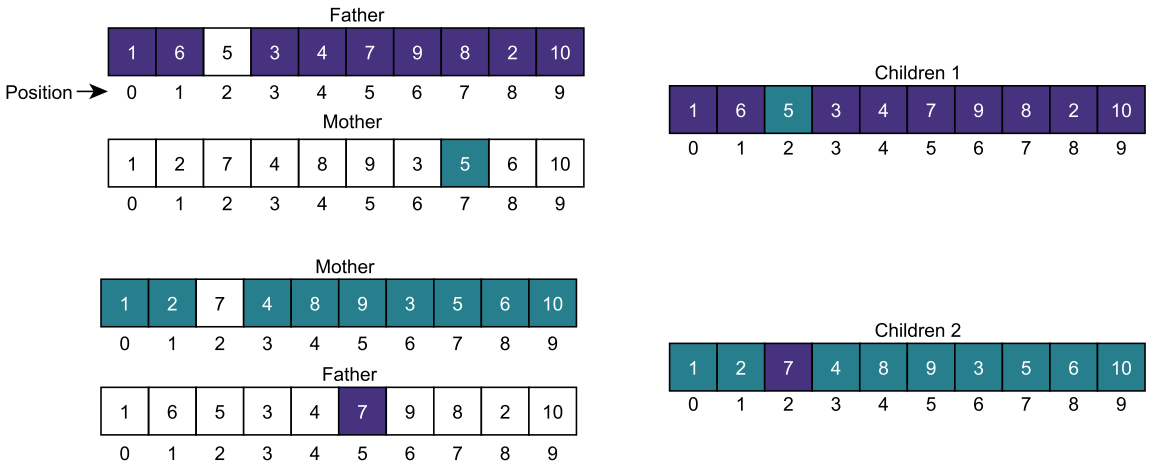

Figure 6: Ejemplo de posiciones de cruce par el Two-Point Crossover que generan descendencia igual a los padres.

Example 6. Retomando el ejemplo anterior, la Fig. 7 ilustra un caso completo de aplicación del operador Two-Point Crossover en el que se generan descendientes diferentes a los padres. En este caso, el primer punto de cruce se ubica en la posición $k_1 = 1$ y el segundo en la posición $k_2 = 6$.

Para la construcción del primer descendiente, los genes pintados de morado corresponden a los valores heredados del padre, seleccionados antes y después de los puntos de cruce. Luego, entre estos puntos (posiciones $[2, \ldots, 5]$), se heredan los genes en el orden de la madre, representados en azul, omitiendo aquellos valores que ya fueron asignados desde el padre, asegurando así la factibilidad de la solución. Para el segundo descendiente, se sigue el mismo procedimiento, pero en orden inverso: primero se asignan los genes de la madre y, posteriormente, los del padre.

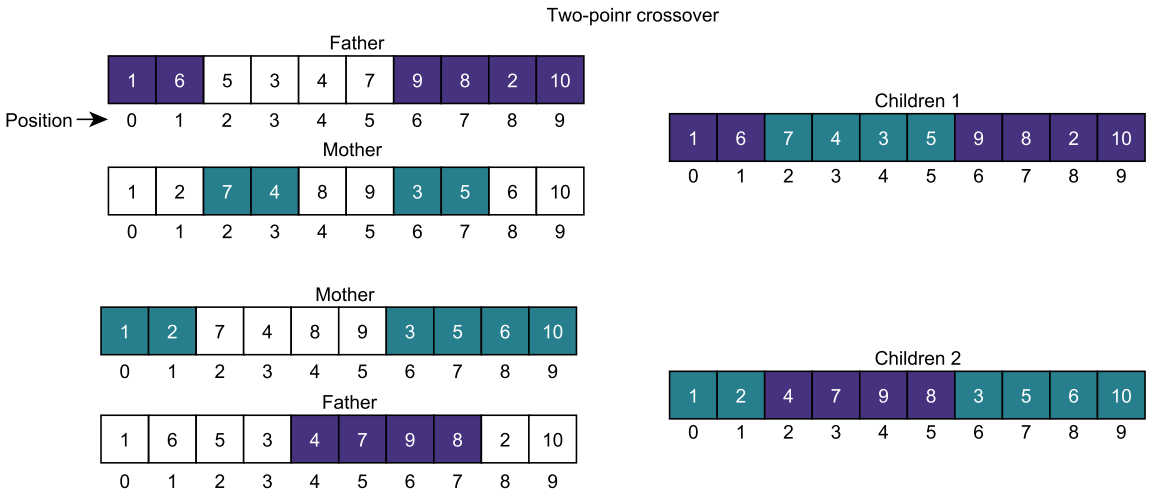

Figure 7: Ejemplo completo de la generación de dos hijos con el operador Two-Point Crossover.

5.1.7 Mutation Operator

El propósito del operador de mutación es introducir variabilidad en la población, esto permite aumentar la diversidad genética en las siguientes generaciones, evitando la convergencia prematura y fomentando la exploración. El operador de mutación aplicado en esta investigación está basado en la inserción de Boctor [79], puesto que es uno de los operadores de mutación más eficientes para el resolver el RCPSP [69, 66]. El operador consiste en seleccionar aleatoriamente una actividad de la AL e insertarla en una posición aleatoria, diferente a la actual, sin incumplir las restricciones de precedencia, manteniendo la factibilidad.

Example 7. La Fig. 8 ilustra un ejemplo del operador de inserción, donde se tiene una AL con 10 actividades y la actividad seleccionada para insertar es la Act_3 que está en la posición 4 de la AL (contando desde la posición 0), y se inserta en la posición 1. Desplazando las actividades 2, 5 y 6 a la derecha de la lista.

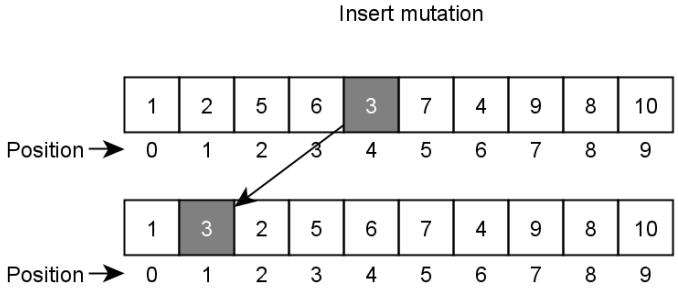

Figure 8: Ejemplo de operador de mutación de inserción

Finalmente, se fija la probabilidad de mutación relativamente alta (en torno al 90%) en comparación con un AG habitual, también se aplican tres inserciones por mutación. Esto se debe a que una probabilidad cercana al 90%-95%, junto con la aplicación de tres inserciones, permite obtener soluciones de alta calidad como lo demuestra Morillo et al. [69]. Para más detalles, véase la Subsección 6.3.

5.1.8 Replacement Operator

El operador de reemplazo determina cuales individuos deben mantenerse en la población (incluyendo la descendencia) para la siguiente generación. Para ello, se tomó como base la estrategia de reemplazo de Mendes et al. [70]. Las nuevas generaciones de la población se conforman de tres subpoblaciones: 1) los mejores individuos (población TOP), 2) los mejores individuos resultantes en las operaciones de cruce y mutación (población $Middle$), y finalmente, 3) los individuos restantes se generan aleatoriamente para introducir mayor diversidad en la población (población BOT). La Fig. 9 ilustra gráficamente la estrategia de reemplazo implementada.

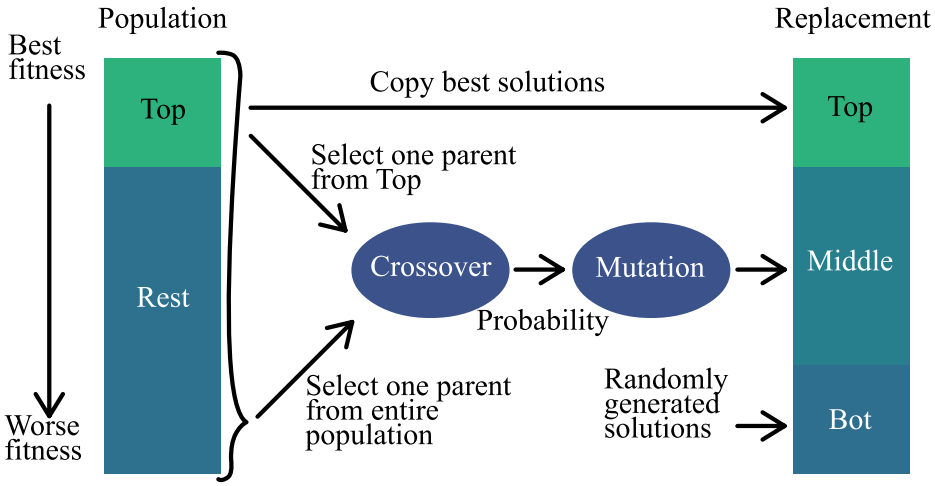

Figure 9: Estrategia de reemplazo adaptada de Mendes et al. [70]

5.2 Metodología de solución robusta para el URCPSP

En la literatura se han propuesto diversas estrategias para abordar la incertidumbre en el RCPSP. Dado que el problema tradicional busca minimizar el makespan (programaciones ajustadas) y el problema robusto busca maximizar la estabilidad del sistema frente a perturbaciones (programaciones holgadas), nos enfrentamos a un problema de naturaliza bi-objetivo. Sin embargo, al tener objetivos contrarios, no hay una solución que optimice ambos simultáneamente. Esto implica que un enfoque de solución basado en trade-offs, incrementa la complejidad del proceso en la toma de decisiones al tener que determinar cuánto sacrificar un objetivo para mejorar otro [80, 81]. Por otra parte, Ma et al. [39] proponen maximizar la robustez dentro del plazo de entrega del proyecto (fecha límite), de esta manera evitan el enfoque bi-objetivo, pero garantizando que el makespan esté dentro del intervalo de tiempo esperado por el cliente. Sin embargo, este enfoque podría permitir obtener más holguras de

las necesarias en el proyecto, siempre que se encuentren dentro del plazo de entrega establecido. Lo que podría impactar negativamente en sobrecostes y un uso poco eficiente de los recursos. Ma et al. [41] proponen otra metodología, en la que inicialmente resuelven el problema determinista por medio de un modelo de programación lineal, maximizando la robustez dentro de los plazos de entrega del proyecto, obteniendo la programación esperada. Después generan 150 escenarios donde varían el consumo de recursos y los tiempos de ejecución de las actividades para simular una programación real. Finalmente, resuelven un modelo de programación lineal teniendo en cuenta el calendario esperado y el calendario real de forma reactiva para validar su propuesta.

En este paper se propone una nueva metodología basada en dos etapas para resolver el URCPSP de forma que se evita simultáneamente la dicotomía de diferentes objetivos, así como la generación de holguras excesivas. La metodología propuesta busca identificar una AL en el espacio de soluciones que optimice tanto la robustez como el makespan. Cada etapa se describe a continuación:

1. En la primera etapa, el RCPSP se resuelve mediante el GA propuesto, cuyo objetivo es minimizar el makespan. Para ello, se modela el comportamiento aleatorio de las actividades con riesgo, utilizando funciones de distribución de probabilidad y se generan una cantidad significativa de escenarios. Estos corresponden a simulaciones de cada instancia para representar la variabilidad en la duración de las actividades. Cada escenario es resuelto de forma individual por el GA, utilizando parámetros deterministas, obtenidos a partir de las distribuciones de probabilidad para las duraciones bajo riesgo. Como resultado, para cada escenario se obtiene una lista de actividades candidatas que producen el mejor makespan posibles dentro de las condiciones bajo incertidumbre definidas, minimizando el impacto de los riesgos sobre la programación del proyecto.

2. En la segunda etapa, las ALs de cada escenario, obtenidas en la primera etapa, se decodifican con las duraciones originales del proyecto. Posteriormente, estas soluciones se evalúan mediante la medida de robustez propuesta (ecuación (44)). Es decir, la ALs candidatas se encuentran mediante la solución de los escenarios bajo incertidumbre, pero son usadas para obtener las secuencias (soluciones) con las duraciones originales del problema (sin incertidumbre). Esto se debe a que no es apropiado medir la robustez una vez se han materializado los riesgos y, con ellos, las duraciones de las actividades afectadas, ya que en la realidad no se puede conocer de antemano qué riesgos o eventos realmente ocurrirán en el proyecto. El objetivo de este paso es evaluar la robustez de cada AL sin depender de un escenario particular, dado que en la práctica no se conoce de antemano cuál se materializará. Pero se puede asegurar que cada lista candidata fue la que mejor makespan encontró para su escenario. Finalmente, se calculas las holguras de cada AL y su robustez correspondiente. Aquella más robusta entre todos los escenarios es seleccionada como la mejor solución al sistema completo.

Por lo tanto, la propuesta consiste en encontrar la programación base más robusta del proyecto, es decir, aquella que puede definirse de manera anticipada a la ocurrencia de los eventos, considerando la naturaleza aleatoria de las actividades pero sin conocer con certeza sus duraciones. Para esto, se selecciona, de la lista de mejores soluciones obtenidas en la

primera etapa, aquella solución que presenta mayor robustez en el problema original. Esta solución representa la secuencia que maximiza la robustez posible, minimizando, al máximo, el makespan, incluso frente a las disrupciones aleatorias en los tiempos de ejecución de las actividades. Cabe resaltar que se requiere de la simulación de un número significativo de escenarios para modelar el comportamiento aleatorio de las actividades.

Para validar la metodología propuesta, se adaptaron las instancias de la conocida PSPLIB mediante un enfoque realista que incorpora riesgos, como se detalla en Sección 6.1. Con el propósito de cuantificar la efectividad de este enfoque, se compara la solución encontrada con la denominada solución *vanilla*. Esta se define como la solución obtenida al resolver la instancia, mediante el GA propuesto, pero sin incorporar incertidumbre (ni escenarios) y midiendo posteriormente su robustez. El la siguiente sección se muestra cómo la solución vanilla no logra encontrar una lista igual de robusta o más robusta que la obtenida con la metodología propuesta, dando consistencia al hipótesis planteada en esta sección.

6 Computational results

En esta sección se detallan los resultados obtenidos en la experimentación computacional divididos en cuatro partes: la Subsección 6.1 describe el conjunto de casos de prueba utilizado y su adaptación al URCPSP; la Subsección 6.2 describe la estimación de la denominada robustez ideal, que se usa como medida de comparación de la medida de robustez propuesta; en la Subsección 6.3 de detalla la parametrización del GA propuesto y, finalmente se exponen los principales resultados obtenidos en la Subsección 6.4.

6.1 Set of instances

Para validar la eficiencia de modelo propuesto, este trabajo también propone la adaptación y generación de casos de prueba con diversos niveles de complejidad para el URCPSP, incluyendo duraciones estocásticas. Actualmente, la librería de casos de prueba más usada para resolver problemas determinísticos para el RCPSP es la PSPLIB propuesta por Kolisch and Sprecher [82] (disponible en `https://www.om-db.wi.tum.de/psplib/`). Esta librería fue generada para el RCPSP uni-modal y multimodal (MRPCPS). En relación con el RCPSP uni-modal, la librería está compuesta por 4 conjuntos de problemas llamados $j30$, $j60$, $j90$ y $j120$, cada uno con 30, 60, 90 y 120 actividades, respectivamente. Los conjuntos $j30$, $j60$ y $j90$ cuentan con 48 subconjuntos de 10 problemas cada uno, y el conjunto $j120$ cuenta con 60 subconjuntos de 10 problemas cada uno, para un total de 1 080 casos de prueba. Cada caso de prueba define un número de actividades, modos de ejecución, disponibilidad y consumo de recursos renovables (RCPSP) y no renovables (MRCPSP), relaciones de precedencia y plazos máximos de ejecución.

Sin embargo, para el RCPSP no determinístico o estocástico no se cuenta con una librería oficial. Además, en la literatura se encuentran adaptaciones donde solo se limita a asignar una función de distribución parametrizada a las actividades, para luego generar escenarios. Esta metodología puede no ser la mejor forma de representar un comportamiento realista; como lo muestra Ortíz-Pimiento and Diaz-Serna [83], quienes realizan un estudio de un caso real del RCPSP. Los autores caracterizan el sistema y se encuentran que no todas las actividades están sujetas a incertidumbre. Algunas de las actividades son altamente precisas o automatizadas, otras, por el contrario, por su naturaleza o por exposición a elementos externos, están sujetas

a cambiar su duración debido a los factores de incertidumbre a las que están expuestas. Además, estos cambios no son causados por una sola fuente de incertidumbre, en su lugar, cada actividad puede ser susceptible a sufrir cambios por diferentes riesgos externos (que se refieren a aquellos acontecimientos que interfieren en el desarrollo normal de las actividades como fallos en la maquinaria, accidentes laborales, entre otros). Los autores determinan que el número de riegos a los que se puede estar asociada una actividad varían dependiendo de las características propias de la actividad. Así, en este trabajo se propone una nueva librería basada en la PSPLIB y en esta metodología basada en riesgos.

Ortíz-Pimiento and Diaz-Serna [83], en su caso de estudio, concluyeron que en el sistema evaluado, las actividades asociadas a riesgos se aproximan al 30% del total de las actividades, y cada una de ellas puede sufrir entre uno o dos riesgos. Finalmente, el impacto de ocurrencia de cada riesgo genera un aumento en la duración de las actividades, y estas se representan por medio de una distribución de probabilidad. De presentarse más de un riesgo, el tiempo adicional se puede asociar con aquel riesgo que genere mayor impacto (mayor duración), asumiendo que el daño o el retraso causado por cada riesgo es independiente entre sí. Se resalta que, bajo el comportamiento aleatorio, no siempre el riesgo con mayor variabilidad será el que genere mayores tiempos de procesamiento.

Teniendo en cuenta lo anterior, para la adaptación de los casos de prueba de la PSPLIB en esta investigación se establecerá como actividades sujetas a riesgos al 30% del total de las actividades del proyecto de forma aleatoria y cada una de ellas estará sujeta a máximo dos riesgos. Posteriormente, se seleccionarán los tipos de riesgos asociados a cada actividad, de entre un total de seis tipos. Este valor se ajustó de forma que se obtengan riesgos suficientemente diversos en los casos de prueba. Para esto se define su comportamiento aleatorio mediante una distribución normal con parámetros μ y σ definidos en las ecuaciones (45) y (46).

$$\mu = \max(P_i) \cdot LR_r \tag{45}$$

$$\sigma = \mu \cdot VL, \qquad VL \in \{0.05, 0.1, 0.2\} \tag{46}$$

Donde P_i es el tiempo de procesamiento de la actividad i y LR_r es el Limit Risk, que se obtiene del intervalo superior del riesgo correspondiente según se muestra en la Tabla 5. Finalmente, VL es el Variability Level que puede tomar valores de 5%, 10% y 20%. De esta forma se pueden generar riesgos de bajo impacto cuyo incremento en la duración será un porcentaje pequeño de la duración máxima, mientras que otros riesgos serán de un alto impacto, cuyo incremento en el tiempo de procesamiento será significativo; en ambos casos, los riegos pueden tener una alta o baja variabilidad. Además de modelar la variabilidad en la duración, es necesario determinar la probabilidad de ocurrencia de cada evento. La literatura reporta casos en los que estas probabilidades varían significativamente, desde valores bajos cercanos al 2% hasta valores elevados de hasta el 50%, dependiendo de la naturaleza de la actividad. Para la adaptación, las probabilidades de ocurrencia de cada actividad se generaron a partir de una distribución uniforme en el intervalo $[0\%, 50\%)$.

Riesgo (r)	Intervalo (LR_r)
1	$[0 - 0.125]$
2	$[0.125 - 0.25]$
3	$[0.25 - 0.375]$
4	$[0.375 - 0.50]$
5	$[0.50 - 0.625]$
6	$[0.625 - 0.75]$
7	$[0.75 - 0.875]$
8	$[0.875 - 1.0]$

Table 5: Intervalo para determinar el tiempo extra de cada actividad asociada a un riesgo.

De esta forma, se tomó la PSPLIB y se adaptaron los conjuntos $j30$, $j60$, $j90$ y $j120$, asignado algunas actividades bajo riesgo y determinando el tipo de riesgo y su variabilidad. La librería propuesta está disponible en https://github.com/dmorill/Robust_PSPLIB.

Para validar la propuesta, se generó una cantidad suficiente de escenarios basados en una prueba computacional piloto. En particular, se crearon 100 escenarios para el conjunto $j30$, 50 para el $j60$, y 30 para el $j120$. En cada uno de estos escenarios, los tiempos de ejecución de las actividades varían debido al comportamiento aleatorio de los riesgos a los que están expuestas. Es decir, dentro de un mismo caso de prueba, las actividades sujetas a riesgos experimentarán diferentes tiempos de ejecución según el escenario en el que se encuentren. En el mejor escenario posible, los riesgos no se materializarán y, por ende, no afectarán los tiempos de ejecución; sin embargo, en el peor de los casos, las actividades afectadas por riesgos verán incrementados sus tiempos de ejecución inicialmente previstos. En aquellos escenarios en los que una actividad se vea impactada por dos riesgos simultáneamente, se asume una reparación paralela de las disrupciones. Por consiguiente, el tiempo de ejecución de la actividad será, en última instancia, el correspondiente al riesgo de mayor impacto.

6.2 Cálculo de robustez ideal

En esta sección se define la denominada Ideal Robustness ($IdRob$) de un URCPSP, así se establece un punto de referencia teórico que permite evaluar la efectividad de los métodos aplicados. Combiertiendose en un punto de comparación para los resultados obtenidos por la metodología propuesta. Este cálculo facilita la cuantificación del desempeño en términos porcentuales, identifica brechas entre la solución encontrada y la establecida como ideal, contextualizando el desempeño de las soluciones dentro de un marco óptimo (o cercano a este), y ofreciendo una perspectiva más completa sobre su comportamiento ante incertidumbres.

A continuación se presenta un esquema general para calcular la robustez ideal:

1. Construcción de la Programación Base Ideal: para ello, se toma la AL ganadora (Sección 5.2), es decir, de todas las ALs candidatas que minimizan el makespan, se toma aquella que tiene mayor robustez; posteriormente se le adiciona la ES (Sección 4.1.1) a las duraciones originales de las actividades sujetas a riesgos. De esta manera, se está garantizando el cumplimiento al 100% de la SFL, así la programación tendría una holgura suficiente para absorber en un 95% las disrupciones aleatorias. Este nuevo problema, con las duraciones ajustadas, se secuencia mediante el SGS Serial para obtener los

tiempos de inicio y fin de la solución correspondiente. Al tratarse de una programación idealmente robusta, se sacrifica en gran medida el makespan a favor de aumentar lo máximo posible la robustez.

La adición de la ES a las duraciones de las actividades bajo riesgo, tiene el propósito de reorganizar la programación, es decir, obtener los tiempos de inicio y fin de cada actividad dejando suficiente holgura para almacenar la ES. Sin embargo, las duraciones originales realmente no se incrementan; dejando espacios en la programación para los eventos inesperados. Un ejemplo se ilustra en la Fig. 10: la parte a) muestra la decodificación de la AL, donde a la duración original de las actividades sujetas a riesgo se les suma el ES; la parte b) presenta la programación con los espacios reservados para las holguras de cada actividad sujeta a riesgo; la parte c) ilustra las holguras asignadas a cada actividad con un nivel de servicio del 95%.

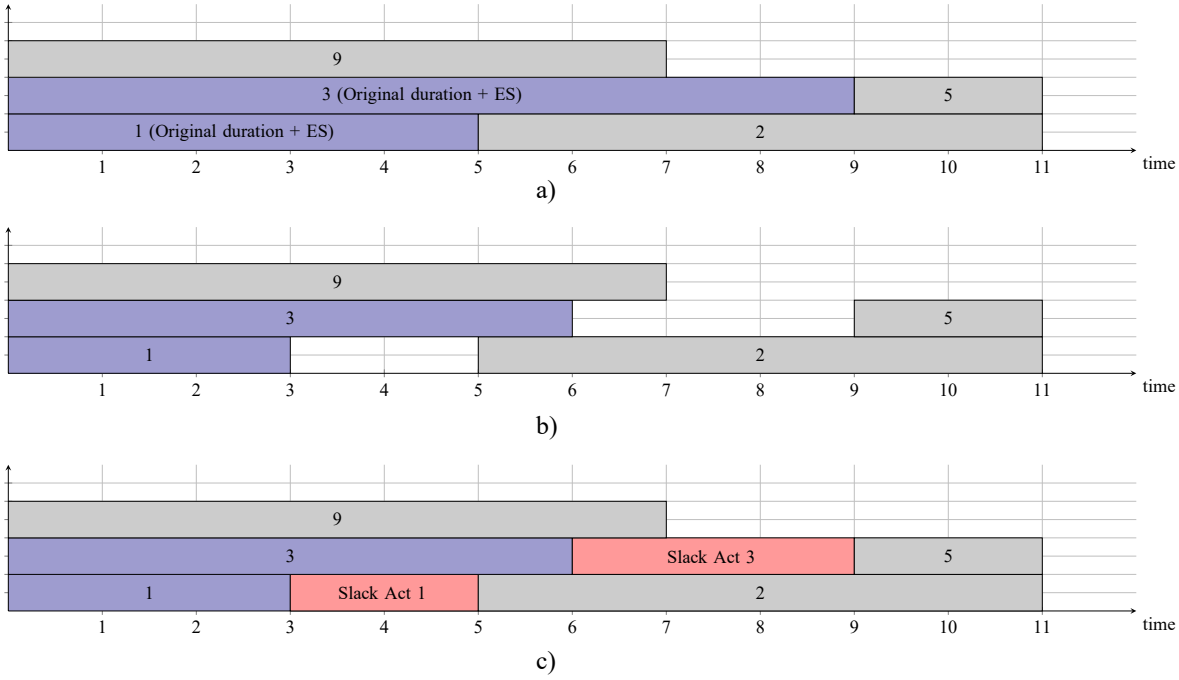

Figure 10: Ejemplo de construcción de la programación base ideal.

2. Medición de robustez ideal: con programación base ideal se evalúa la robustez con la nueva medida propuesta, considerando la holgura ideal de cada actividad sujeta a riesgo. No obstante, las holguras pueden superar las ideales según el resultado del SGS, sin que esto represente un impacto significativo (gracias a la función SFL). Además, se tiene en cuenta el uso de recursos y la relación entre sucesores, de la misma manera que en la medición de robustez en escenarios no ideales. Finalmente, se compara la robustez obtenida mediante la metodología propuesta con la robustez ideal y se calcula el porcentaje de cumplimiento, considerando la robustez ideal como el 100%.

6.3 Genetic Algorithm Parameter Tuning

La parametrización de una metaheurística es un factor determinante en su rendimiento, ya que afecta directamente en el equilibrio entre exploración y explotación del espacio de búsqueda, la eficiencia computacional, y maximiza la capacidad del algoritmo para encontrar soluciones de alta calidad. En el GA, parámetros como el tamaño de la población, la probabilidad de mutación y el número de generaciones afectan la capacidad del algoritmo para evitar la convergencia prematura hacia óptimos locales y mantener la diversidad genética necesaria para explorar soluciones novedosas.

En la literatura existen varios métodos para la configuración de parámetros la mayoría basados en Design of Experiments (DOE). Entre ellos, el Método Taguchi ha mostrado resultados exitosos en investigaciones sobre problemas de programación de tareas, como lo muetran los trabajos de Rabet et al. [84], Hosseinian and Baradaran [85], Asadujjaman et al. [86], Saad et al. [87]. Este método evalúa el comportamiento de distintas combinaciones de parámetros con el objetivo de identificar el conjunto óptimo teniendo en cuenta los factores de ruido, especialmente cuando no se puede hacer un diseño completo, debido al alto costo computacional. Por esto, el Método Taguchi se adoptará en este estudio.

Para este diseño se definieron 5 factores diferentes en 3 niveles. La matriz ortogonal empleada en este estudio es $L27(36)$; por tanto, hay 27 combinaciones en el DOE. Cada configuración de parámetros fue evaluada en los conjuntos adaptados de la PSPLIB: $j30$, $j60$ y $j120$, utilizando 20 instancias seleccionadas aleatoriamente por conjunto interno, es decir, de cada conjunto completo ($j30$, $j60$, y $j120$) se tomaron las muestras de cada subconjuntos que lo conforman (40, 40 y 60 subconjuntos para el $j30$, $j60$, y $j120$, respectivamente). la variable de respuesta para cada combinación corresponde al promedio de los makespan obtenidos al resolver las 20 instancias. Las configuraciones se muestran en la Tabla 6.

	Niveles		
Parámetros	**1**	**2**	**3**
Population	50	100	150
TOP	0.1	0.15	0.2
BOT	0.15	0.2	0.25
ProbMut	0.05	0.5	0.9
Generations	20	50	70

Table 6: Combinación de los valores de los parámetros para aplicación Taguchi.

Para la población se establecieron los niveles con lo observado en la literatura, y con un trade off aceptable de calidad y esfuerzo computacional. Los parámetros TOP y BOT se definieron dentro de los intervalos propuestos por Mendes et al. [70]. Para el parámetro BOT, se redujo el porcentaje de la población BOT, ya que los autores lo emplearon como operador de mutación. Sin embargo, en nuestro enfoque, incluimos este operador dentro del GA, lo que ya contribuye a la diversificación de la población. Para la probabilidad de mutación, autores como Liu et al. [88] y Mahmud et al. [89] definen la probabilidad de mutación como 5%. Sin embargo, Mendes et al. [70] genera soluciones aleatorias en cada generación, lo que implica una probabilidad de mutación del 100% para esos individuos.

Por su parte, Morillo et al. [69], a partir de pruebas con la PSPLIB, determinó que en el RCPSP una probabilidad de mutación cercana al $[90\%, 95\%]$ y la aplicación de 3 inserciones por mutación permiten obtener soluciones de alta calidad en comparación con otros métodos, reduciendo además la generación de soluciones redundantes. Por ello, se definieron los valores extremos para la parametrización de la mutación en 0.05 y 0.9, con un valor medio de 0.5. Finalmente, para el número de generaciones se tuvieron en cuenta estudios previos, donde se encontró que aumentar las generaciones mejora la calidad de la solución, pero incrementa el tiempo de cómputo. En particular, se tomaron como referencia los valores de 20 y 50 generaciones utilizados por Luo et al. [90], y se incluyó 70 generaciones para evaluar si mayores valores pueden generar mejoras adicionales. No se consideraron valores más altos para evitar comprometer la eficiencia computacional.

Las Figuras. 11, 12 y 13 presentan los resultados obtenidos para los conjuntos $j30$, $j60$ y $j120$, respectivamente, mientras que la Tabla 7 resume los valores óptimos por conjunto de instancias determinados a partir de la relación señal/ruido y la minimización del makespan.

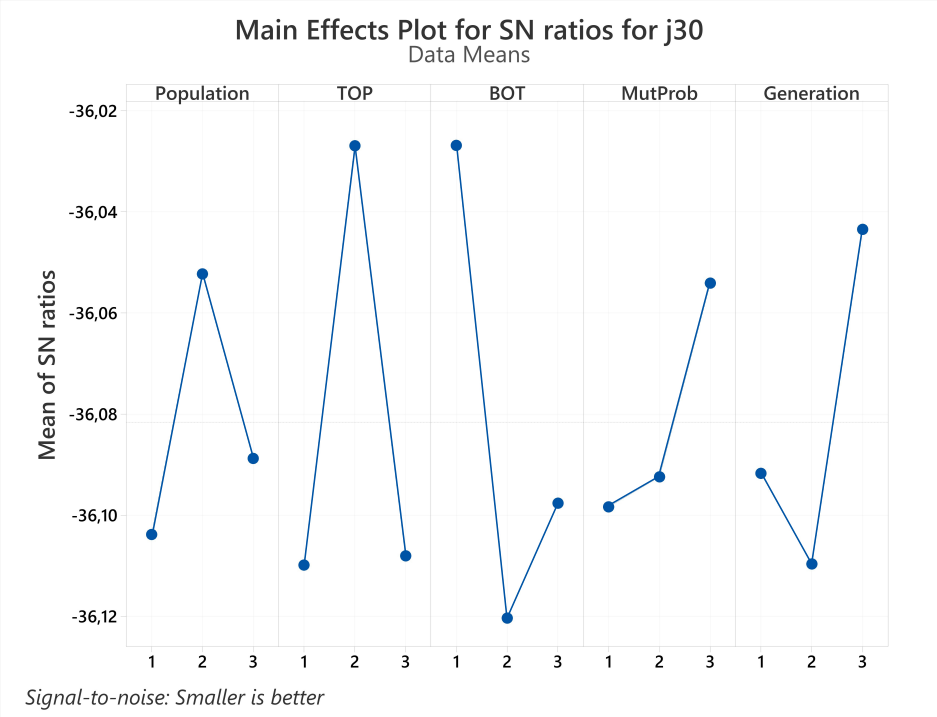

Figure 11: Results of the Taguchi design for the $j30$ set.

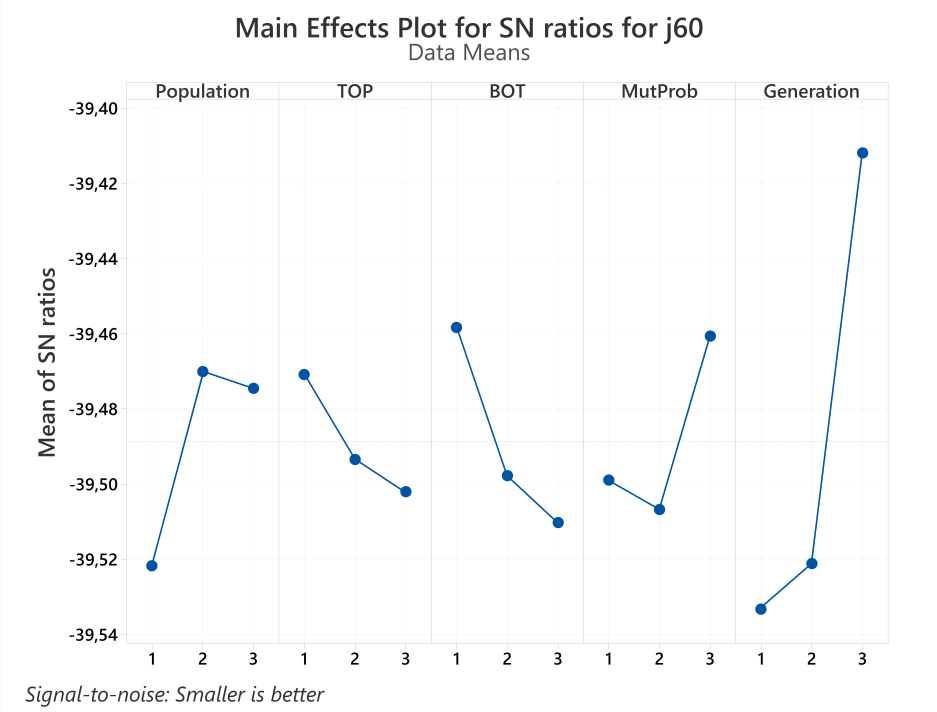

Figure 12: Results of the Taguchi design for the $j60$ set.

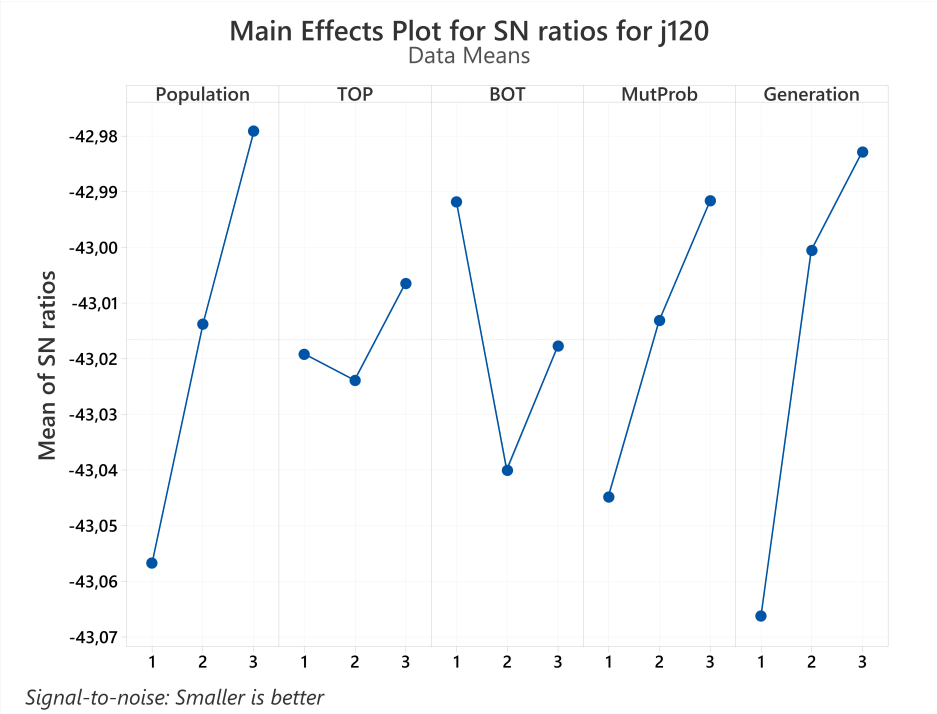

Figure 13: Results of the Taguchi design for the $j120$ set.

	Instancias		
Parámetros	**j30**	**j60**	**j120**
Population	100	100	150
TOP	0.15	0.1	0.2
BOT	0.15	0.15	0.15
ProbMut	0.9	0.9	0.9
Generations	70	70	70

Table 7: Tuned GA parameters for each instance type.

A partir de estos resultados, se concluye que los parámetros BOT, ProbMut y Generations parecen ser independientes del número de actividades. Adicionalmente, el resultado de ProbMut de 0.9 demuestra que una mayor probabilidad de mutación aumenta la calidad de soluciones. Estos parámetros se fijan en el GA con los valores óptimos de la experimentación Taguchi (Tabla 7). Por otra parte, el tamaño de la población mostró un comportamiento esperado, aumentando su valor con el número de actividades debido a la expansión del espacio de búsqueda, sugiriendo que se requiere una mayor diversidad para mejorar la exploración de soluciones. Sin embargo, con el fin de mantener la eficiencia del algoritmo genético (GA), se decidió conservar una población de 100 para el conjunto $j120$.

Por último, el parámetro TOP varía en función del tamaño de la instancia, lo que indica que su ajuste es crucial para equilibrar exploración y explotación en diferentes escalas del problema, pero los resultados de Taguchi no fueron concluyente al respecto. Para validar su parametrización, se llevó a cabo un análisis de varianza (ANOVA), con el objetivo de determinar su impacto en el rendimiento del algoritmo y verificar si las diferencias observadas entre instancias son estadísticamente significativas. Se realizaron 9 experimentos, donde cada uno corresponde al promedio del makespan obtenido al resolver 20 instancias. Para evitar generar ruido y que los resultados sean comparables, se emplearon las mismas instancias seleccionadas aleatoriamente por conjunto para la experimentación Taguchi, y se fijaron los demás parámetros (Population, BOT, ProbMut y Generations) con los valores óptimos previamente obtenidos. Los resultados se muestran en la Fig 14.

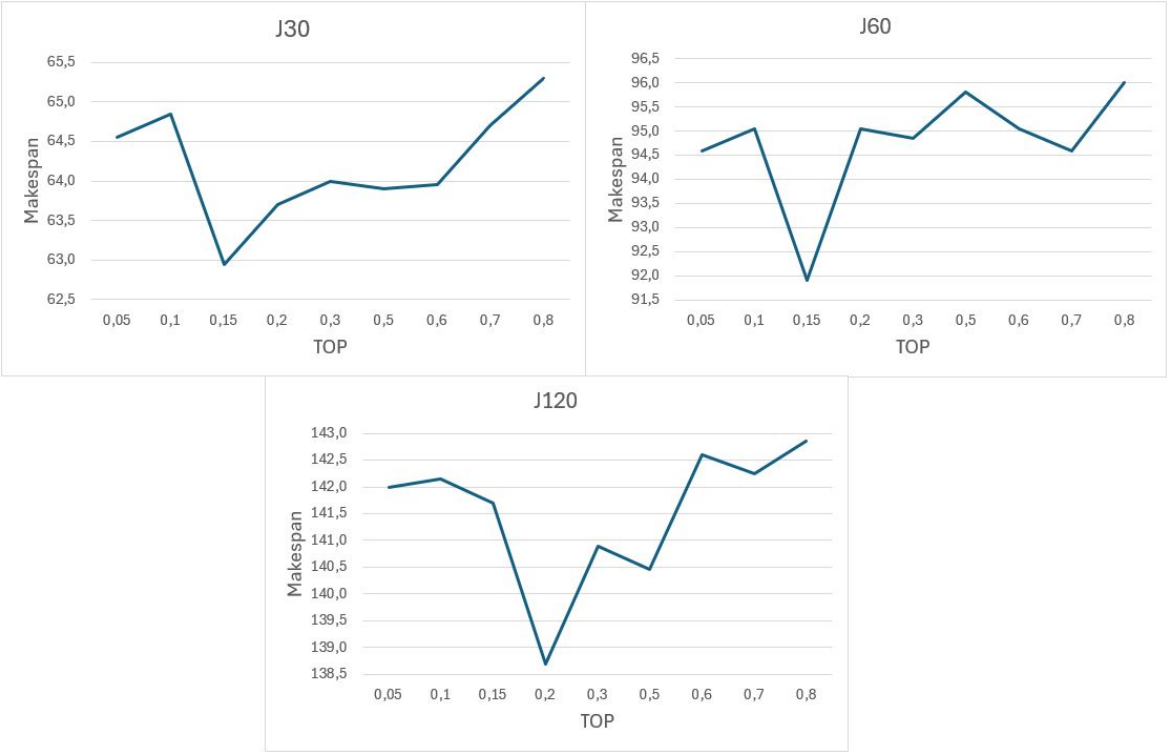

Figure 14: ANOVA de TOP para los conjuntos $j30$, $j60$ y $j120$.

Estos resultados indican que una población TOP cercana al $[15\%, 20\%]$ es apropiada independiente del tamaño de la población (al menos hasta 120 actividades). Esto puede deberse a que ya cuenta con suficiente diversidad (por la mutación y la población nueva BOT) y es relevante mantente un conjunto de soluciones de élite. Conforme a los resultados del ANOVA, se parametriza la población TOP con los valores de 0.15 para los conjuntos de instancias $j30$ y $j60$, y 0.2 para el conjunto de los $j120$.

6.4 Computational results

Esta subsección se divide en cuatro partes. Primero, en la Sección 6.4.1 se presentan los resultados obtenidos a partir de las pruebas computacionales realizadas, organizados en tablas para facilitar su visualización. Segundo, en la Sección 6.4.2 se realiza un análisis comparativo entre los resultados obtenidos mediante la metodología propuesta y los de la programación base ideal. Tercero, en la Sección 6.4.3 se compara la solución obtenida mediante la metodología propuesta con la solución vanilla (definida al final de la Sección 5.2). Finalmente, en la Sección 6.4.4 se analiza la relación entre holgura y robustez, así como entre entropía y robustez.

6.4.1 Presentación general de resultados

En esta subsección se presentan los resultados obtenidos a partir de las pruebas computacionales realizadas de la metodología propuesta en los casos de prueba considerados. Estos resultados se organizan en tablas, que servirán de base para los análisis comparativos en las

siguientes subsecciones. Por lo tanto, esta sección se limita a la presentación de los datos, mientras que su análisis se desarrolla en las subsecciones posteriores.

El algoritmo se implementó en C++ y las pruebas se realizaron en un ordenador con CPU Intel(R) Core(TM) i7-6700 CPU 3.40GHz y 16 gb de memoria RAM. La Tabla 8 muestra la cantidad de instancias resueltas y la cantidad de escenarios generados por cada instancia de cada conjunto de problemas. Por ejemplo, para el conjunto $j30$ se resolvieron 40 instancias seleccionadas aleatoriamente, cada una con 100 escenarios, lo que equivale a un total de 4 000 problemas. Las tablas 10, 11 y 12 muestran los resultados obtenidos para el conjunto $j30$, $j60$ y $j120$, respectivamente. La nomenclatura usada se muestra en la Tabla 9.

Conjunto	**j30**	**j60**	**j120**
Instancias	40	20	10
Escenarios	100	50	30
Problemas resueltos	4 000	1 000	300

Table 8: Resumen de instancias, escenarios y problemas resueltos.

Símbolo	**Descripción**
Ma	Makespan de la solución obtenida mediante la metodología propuesta.
Ro	Robustez de la solución obtenida mediante la metodología propuesta.
En	Entropía de la solución obtenida mediante la metodología propuesta.
T	Tiempo computacional necesario para obtener la solución.
TS	Holgura total obtenida, que es la suma de las holguras en la programación.
RoI	Robustez de la programación base ideal.
MaI	Makespan asociado a la solución con robustez ideal.
MaV	Makespan de la solución vanilla (sin técnicas de robustez).
RoV	Robustez de la solución vanilla.
TSv	Holgura total de la solución vanilla, que es la suma de holguras en la programación de la solución vanilla.
Sacr	Sacrificio del makespan, que es el incremento en el makespan de la solución obtenida al mejorar la robustez respecto a la solución vanilla.
mRo	Mejora de la robustez, que es el aumento porcentual de la robustez en la solución obtenida en comparación con la solución vanilla.

Table 9: Nomenclatura de las tablas de resultados.

6.4.2 Análisis de resultados: metodología propuesta vs programación base ideal

En esta sección se comparan los resultados obtenidos mediante la metodología propuesta con la robustez ideal definida en la Sección 6.2. Con base en los resultados de la Tabla 10, se observa que maximizar la robustez implica un aumento significativo en el makespan, lo que impacta directamente los tiempos de ejecución del proyecto, los costos y el cumplimiento de plazos. Por ejemplo, en la instancia $j3019\_5Robu\_e\_39.sm$, la solución con robustez ideal alcanza un makespan de 119 unidades de tiempo, mientras que la estrategia propuesta logra reducirlo a 56 unidades, priorizando un makespan más corto con la máxima robustez posible

J30												
Solución obtenida						**Solución Ideal**		**Solución Vanilla**			**Ob vs Vani**	
Instance	Ma	Ro	En	T	TS	RoI	MaI	MaV	RoV	TSv	Sacr	mRo
j301_7Robu_e_20	60	0,060	-0,593	7,361	112	0,188	91	60	0,027	78	0,00%	121,32%
j302_2Robu_e_57	56	0,075	-0,820	7,407	133	0,123	68	51	0,033	63	9,80%	129,30%
j303_10Robu_e_67	59	0,112	-0,631	6,658	110	0,242	86	59	0,044	135	0,00%	152,67%
j304_5Robu_e_71	59	0,052	-0,423	7,876	82	0,257	90	59	0,044	123	0,00%	17,87%
j305_8Robu_e_14	70	0,046	-0,471	12,415	36	0,220	122	67	0,000	18	4,48%	∞
j306_10Robu_e_89	68	0,032	-0,596	9,127	94	0,324	96	61	0,020	64	11,48%	59,93%
j307_3Robu_e_43	50	0,027	-0,297	6,016	59	0,209	75	42	0,000	38	19,05%	∞
j308_9Robu_e_16	44	0,067	-0,657	6,094	110	0,334	68	39	0,044	82	12,82%	52,47%
j309_9Robu_e_1	68	0,082	-0,525	9,167	21	0,297	113	63	0,000	8	7,94%	∞
j3010_2Robu_e_22	57	0,091	-0,556	8,274	20	0,187	91	56	0,000	30	1,79%	∞
j3011_9Robu_e_4	67	0,060	-0,566	7,548	61	0,205	94	67	0,046	61	0,00%	32,04%
j3012_10Robu_e_7	57	0,050	-0,511	7,083	125	0,114	72	57	0,028	94	0,00%	79,04%
j3013_8Robu_e_15	113	0,044	-0,571	14,423	49	0,075	154	109	0,013	23	3,67%	254,95%
j3014_4Robu_e_97	51	0,045	-0,445	8,193	14	0,127	85	50	0,000	7	2,00%	∞
j3015_3Robu_e_9	48	0,030	-0,476	7,418	30	0,113	68	48	0,010	33	0,00%	190,45%
j3016_5Robu_e_2	51	0,054	-0,444	6,341	145	0,119	69	51	0,054	145	0,00%	0,00%
j3017_9Robu_e_38	54	0,076	-0,244	6,340	115	1,374	92	48	0,000	106	12,50%	∞
j3019_5Robu_e_39	56	0,132	-0,599	7,272	75	0,410	119	48	0,000	32	16,67%	∞
j3020_10Robu_e_1	37	0,068	-0,614	5,979	51	0,292	69	37	0,068	51	0,00%	0,00%
j3021_9Robu_e_66	81	0,139	-0,447	12,034	42	0,280	112	69	0,031	31	17,39%	350,93%
j3023_5Robu_e_77	55	0,063	-0,489	6,835	79	0,238	95	52	0,041	51	5,77%	55,15%
j3024_2Robu_e_1	58	0,039	-0,524	5,788	115	0,103	89	58	0,000	110	0,00%	∞
j3025_7Robu_e_39	98	0,048	-0,577	10,698	29	0,278	155	96	0,043	23	2,08%	10,55%
j3026_7Robu_e_5	58	0,226	-0,292	7,734	26	0,585	104	56	0,017	19	3,57%	1256,91%
j3027_9Robu_e_82	61	0,037	-0,477	6,917	46	0,118	79	55	0,000	33	10,91%	∞
j3028_2Robu_e_22	58	0,026	-0,431	6,001	91	0,187	97	57	0,025	73	1,75%	4,37%
j3029_6Robu_e_15	104	0,062	-0,598	12,140	25	0,137	150	94	0,000	5	10,64%	∞
j3030_7Robu_e_24	74	0,045	-0,594	9,020	26	0,295	98	68	0,000	14	8,82%	∞
j3031_3Robu_e_66	62	0,045	-0,578	7,595	55	0,092	86	58	0,023	70	6,90%	98,36%

J30												
Solución obtenida						**Solución Ideal**		**Solución Vanilla**			**Ob vs Vani**	
Instance	Ma	Ro	En	T	TS	RoI	MaI	MaV	RoV	TSv	Sacr	mRo
j3032_9Robu_e_3	65	0,053	-0,696	7,274	112	0,148	97	65	0,009	96	0,00%	465,90%
j3033_7Robu_e_7	58	0,197	-0,535	7,821	65	0,260	110	58	0,184	62	0,00%	6,91%
j3035_9Robu_e_1	59	0,153	-0,230	7,870	47	0,460	95	59	0,153	47	0,00%	0,00%
j3036_3Robu_e_2	61	0,050	-0,574	6,564	109	0,374	94	61	0,040	88	0,00%	27,60%
j3037_10Robu_e_30	82	0,045	-0,469	8,696	60	0,577	136	81	0,000	41	1,23%	∞
j3039_9Robu_e_33	87	0,177	-0,721	6,829	76	0,262	93	64	0,071	63	35,94%	148,80%
j3042_4Robu_e_8	51	0,046	-0,426	7,653	38	0,131	83	49	0,023	17	4,08%	101,81%
j3043_9Robu_e_67	64	0,074	-0,577	7,010	44	0,177	91	57	0,000	30	12,28%	∞
j3044_4Robu_e_5	57	0,028	-0,445	7,349	34	0,201	108	57	0,028	34	0,00%	0,00%
j3045_4Robu_e_42	97	0,042	-0,362	10,646	24	0,237	154	85	0,013	9	14,12%	233,40%
j3046_5Robu_e_70	65	0,027	-0,446	8,676	23	0,297	95	57	0,000	7	14,04%	∞

Table 10: Resultados del GA implementando la nueva medida de robustez propuesta en el conjunto de casos de prueba $j30$.

J60												
Solución obtenida						**Solución Ideal**		**Solución Vanilla**			**Ob vs Vani**	
Instance	Ma	Ro	En	T	TS	RoI	MaI	MaV	RoV	TSv	Sacr	mRo
j6040_6Robu_e_2	69	0,038	-0,814	20,666	120	0,101	102	69	0,038	120	0,0%	0,0%
j6034_4Robu_e_49	83	0,114	-0,834	20,736	216	0,302	129	83	0,039	191	0,0%	188,8%
j602_2Robu_e_41	85	0,080	-0,773	26,042	217	0,185	132	82	0,037	125	3,7%	117,4%
j6014_2Robu_e_39	70	0,016	-0,276	32,077	28	0,100	108	65	0,005	25	7,7%	214,6%
j6048_3Robu_e_44	84	0,041	-0,656	21,764	156	0,066	108	84	0,034	151	0,0%	19,7%
j6013_3Robu_e_14	102	0,013	-0,649	52,273	37	0,050	149	93	0,013	19	9,7%	5,5%
j6012_4Robu_e_3	69	0,048	-0,527	21,745	217	0,335	119	69	0,048	217	0,0%	0,0%
j606_4Robu_e_22	81	0,061	-0,510	30,547	102	0,237	117	67	0,045	44	20,9%	37,6%
j6041_9Robu_e_41	150	0,030	-0,719	59,856	79	0,119	216	135	0,000	25	11,1%	∞
j604_7Robu_e_1	67	0,074	-0,632	22,826	278	0,154	109	67	0,074	278	0,0%	0,0%
j6027_9Robu_e_3	78	0,035	-0,798	24,156	124	0,078	98	76	0,033	104	2,6%	5,6%
j601_3Robu_e_27	79	0,031	-0,844	27,126	161	0,142	108	68	0,019	152	16,2%	64,6%
j6023_8Robu_e_15	72	0,042	-0,915	22,201	142	0,078	99	72	0,038	139	0,0%	11,0%
j6031_10Robu_e_11	61	0,035	-0,537	22,996	50	0,064	95	56	0,012	29	8,9%	205,2%
j6022_8Robu_e_7	76	0,030	-0,778	24,823	132	0,089	124	59	0,010	43	28,8%	191,4%
j6017_2Robu_e_37	76	0,041	-0,856	27,598	162	0,315	126	69	0,026	140	10,1%	56,4%
j6011_9Robu_e_26	69	0,037	-0,632	20,968	185	0,074	111	62	0,013	88	11,3%	183,2%
j6045_10Robu_e_4	123	0,025	-0,791	42,937	23	0,078	211	120	0,000	14	2,5%	∞
j6021_5Robu_e_40	101	0,089	-0,681	36,606	119	0,258	145	91	0,033	111	11,0%	169,7%
j607_5Robu_e_1	71	0,042	-0,959	23,857	131	0,073	90	71	0,041	145	0,0%	3,1%

Table 11: Resultados del GA implementando la nueva medida de robustez propuesta en el conjunto de casos de prueba $j60$.

J120												
Solución obtenida						**Solución Ideal**		**Solución Vanilla**			**Ob vs Vani**	
Instance	Ma	Ro	En	T	TS	RoI	MaI	MaV	RoV	TSv	Sacr	mRo
j1205_7Robu_e_15	100	0,028	-0,973	80,930	315	0,053	140	84	0,016	272	19,048%	70,506%
j12045_10Robu_e_2	107	0,027	-1,080	71,994	401	0,075	157	99	0,017	282	8,081%	58,561%
j12056_9Robu_e_16	339	0,016	-0,718	391,675	72	0,055	493	314	0,005	31	7,962%	206,909%
j12048_4Robu_e_18	147	0,024	-0,920	136,434	148	0,057	231	134	0,008	102	9,701%	216,737%
j1208_3Robu_e_16	108	0,022	-1,118	116,526	96	0,055	161	102	0,010	66	5,882%	122,839%
j12039_5Robu_e_25	112	0,014	-0,642	126,492	39	0,050	171	106	0,007	15	5,660%	103,158%
j12055_2Robu_e_13	89	0,015	-0,910	78,578	115	0,034	138	83	0,010	45	7,229%	55,119%
j12030_9Robu_e_16	93	0,014	-0,920	101,933	125	0,041	142	93	0,009	144	0,000%	54,479%
j12041_3Robu_e_16	159	0,015	-1,127	130,278	266	0,032	211	148	0,008	235	7,432%	80,590%
j12013_6Robu_e_13	116	0,023	-0,458	157,430	52	0,046	188	106	0,005	39	9,434%	334,450%

Table 12: Resultados del GA implementando la nueva medida de robustez propuesta en el conjunto de casos de prueba $j120$.

dentro de ese límite. Particularmente, para esa instancia representa una reducción superior al 50% del makespan. De manera similar, en la instancia $j305\_8Robu\_e\_14.sm$, el makespan disminuye de 122 a 70 sin comprometer significativamente la estabilidad de la programación en función de la robustez. Los valores promedio, máximos y mínimos de makespan y robustez para el conjunto $j30$ obtenidos por la propuesta y por la programación ideal se resumen en la Tabla 13.

Propuesta			
Métrica	**Mínimo**	**Máximo**	**Promedio**
Makespan [periodos]	37	113	65
Robustness [und]*	0,026	0,226	0,071
Ideales			
Métrica	**Mínimo**	**Máximo**	**Promedio**
Makespan [periodos]	68	155	99
Robustness [und]*	0,075	1,374	0,266

* Unidades de robustez conforme a la Sección 4.

Table 13: Comparación de valores máximos, mínimos y promedios de los resultados obtenidos por la propuesta vs resultados idealmente robustos para el conjunto $j30$.

La Fig. 16 muestra la comparación entre ambas estrategias en términos de makespan y robustez. Se observa que, aunque la robustez ideal siempre es mayor, también implica un incremento en el makespan. A pesar de que la solución idealmente robusta presenta una mayor estabilidad ante perturbaciones, esta ventaja implica un aumento de gran magnitud en el makespan, que en promedio pasa de 65 a 99 (mas de un 52.3%). Este incremento no solo representa un mayor tiempo de ejecución del proyecto, sino que también puede traducirse en costos adicionales y menor eficiencia en la utilización de recursos. En contraste, la metodología propuesta logra tiempos de finalización significativamente menores sin comprometer la estabilidad de manera crítica, alcanzando en promedio un 32% de la robustez ideal, tomando esta última como el 100%. A diferencia de la solución idealmente robusta, que prioriza la estabilidad a costa de un incremento en el makespan, nuestra estrategia maximiza la robustez posible dentro de los valores mínimos de makespan, evitando prolongar innecesariamente la duración del proyecto. Por otra parte, al analizar los valores de robustez ideal, se observa que su valor máximo (1.374) es un dato atípico, considerablemente superior al promedio (0.266). Esto sugiere que algunas soluciones presentan una robustez extrema en ciertos casos particulares, pero no reflejan el comportamiento general del sistema. En cambio, la metodología propuesta alcanza una robustez más consistente sin sacrificar la duración del proyecto.

Cabe destacar que algunas instancias logran valores de robustez significativamente superiores al promedio de la metodología propuesta. En particular, las instancias 2, 13, 31 y 35 alcanzan un 60.7%, 59.4%, 75.5% y 67.3% de la robustez ideal, respectivamente. Esto demuestra que, en ciertos casos, es posible obtener altos niveles de estabilidad sin incurrir en el alto costo en makespan que implica la programación idealmente robusta. Otro aspecto clave es la subutilización de recursos. Una programación excesivamente robusta puede dejar

inactivos ciertos recursos para preservar flexibilidad ante retrasos, como ocurre en la solución con robustez ideal. Sin embargo, esto conlleva una menor eficiencia y mayores costos operativos. Además, no todas las perturbaciones ocurren en la realidad, por lo que diseñar un plan excesivamente conservador puede ser innecesario y poco eficiente en la práctica. La metodología propuesta encuentra un balance entre estabilidad y eficiencia, alcanzando la máxima robustez posible sin incrementar significativamente el makespan, lo que permite absorber incertidumbres comunes sin incurrir en los elevados costos de una solución idealmente robusta.

En la Fig. 17 se presenta una comparación de la programación de actividades de la instancia $j307\_3Robu\_e\_43.sm$. Se observa que la estrategia propuesta (Fig. 17-a) permite un uso más equilibrado de los recursos, asegurando una asignación eficiente sin generar tiempos muertos o innecesarios. La programación es más compacta, con una ejecución eficiente de actividades; por ejemplo la 16 y la 14 comienzan en tiempos más tempranos en comparación con la solución idealmente robusta. Por otro lado, en la solución idealmente robusta (Fig. 17-b), se evidencia una mayor dispersión en la holgura de las actividades; por ejemplo la holgura de la actividad 16 se extiende por un período considerablemente mayor (hasta $t = 47$), lo que, si bien incrementa la holgura ante perturbaciones, también introduce tiempos muertos significativos en la secuencia y un uso ineficiente de los recursos. Específicamente en los los periodos del 38 al 43 no se secuencia otras actividades. Esta diferencia en la estructuración del cronograma impacta directamente el makespan, reduciéndolo de 75 a 50 unidades de tiempo en la solución propuesta. Esto demuestra que la metodología desarrollada logra alcanzar un nivel de robustez significativo sin incurrir en sobrecostos ocasionados por holguras excesivas, favoreciendo un balance entre eficiencia y flexibilidad operativa.

En cuanto a los conjuntos $j60$ y $j120$, los resultados se comportan de manera similar al conjunto $j30$, como se observa en las figuras 15 y 18. En ambos casos, los valores de la robustez ideal son mayores que los obtenidos, pero también se presenta un aumento considerable en el makespan. Para el conjunto $j60$, los valores de robustez ideal varían entre 0.049 y 0.334, mientras que los valores de robustez alcanzados se encuentran en un rango menor, con un máximo de 0.113. En términos de makespan, se observa que las instancias con robustez ideal tienden a presentar valores más altos, como en el caso de la instancia $j6041\_9Robu\_e\_41$, donde el makespan alcanza un valor de 150 unidades. En promedio, la solución propuesta presenta un makespan de 83.3 y una robustez de 0.046, mientras que para la solución con robustez ideal alcanza un makespan promedio de 124.8 y una robustez es de 0.144. En el caso del conjunto $j120$, el comportamiento es similar. La robustez ideal varía entre 0.032 y 0.074, mientras que la robustez obtenida se encuentra en un rango inferior, con un máximo de 0.028. En promedio, la robustez obtenida es de 0.019, mientras que la robustez ideal alcanza un valor promedio de 0.049. En términos de makespan, los valores más altos de robustez ideal se asocian con makespan elevados, como en la instancia $j12056\_9Robu\_e\_16$, cuyo makespan alcanza 339. En promedio, el makespan para la solución propuesta es de 137 con una robustez de 0.019, mientras que, para la solución con robustez ideal, el makespan promedio es de 203.2 y la robustez de 0.049.

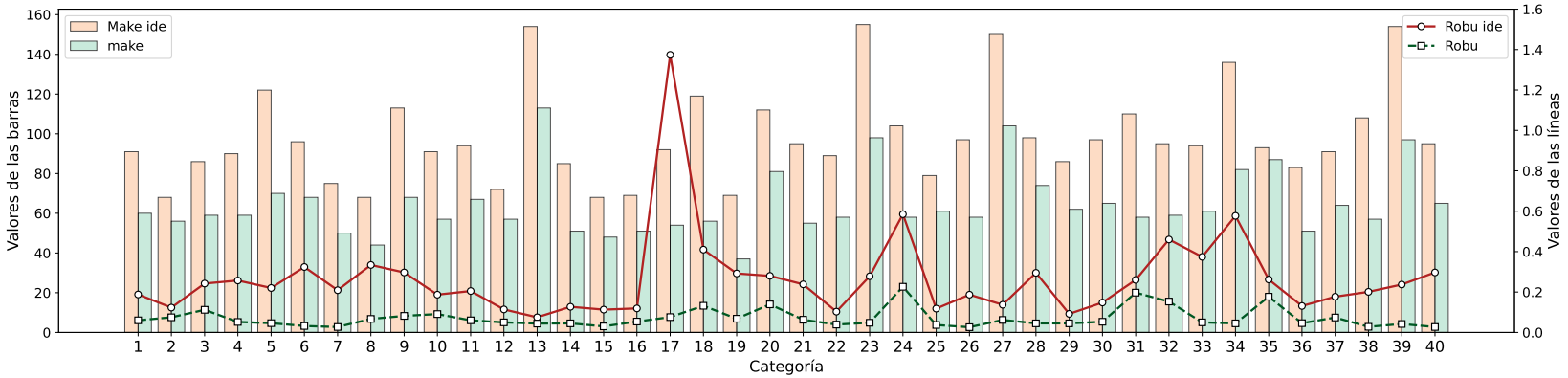

Figure 16: Comparación resultados obtenidos versus resultados de la programación base ideal para el conjunto de instancias $j30$.

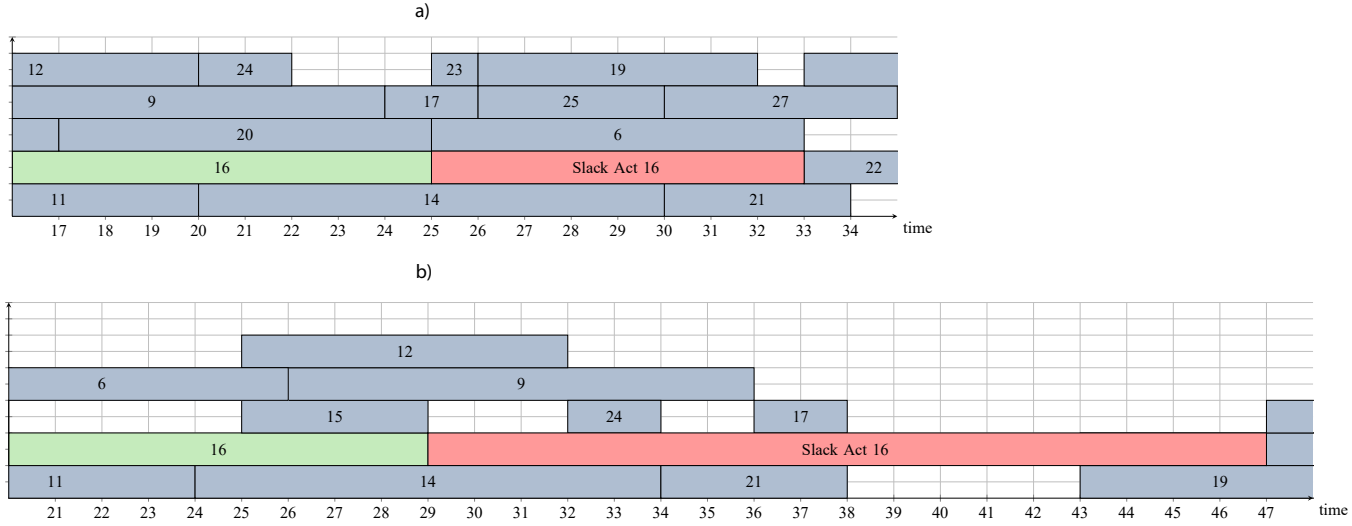

Figure 17: Comparación entre la programación de la solución propuesta y la solución idealmente robusta para la instancia $j307\_3Robu\_e\_43.sm$.

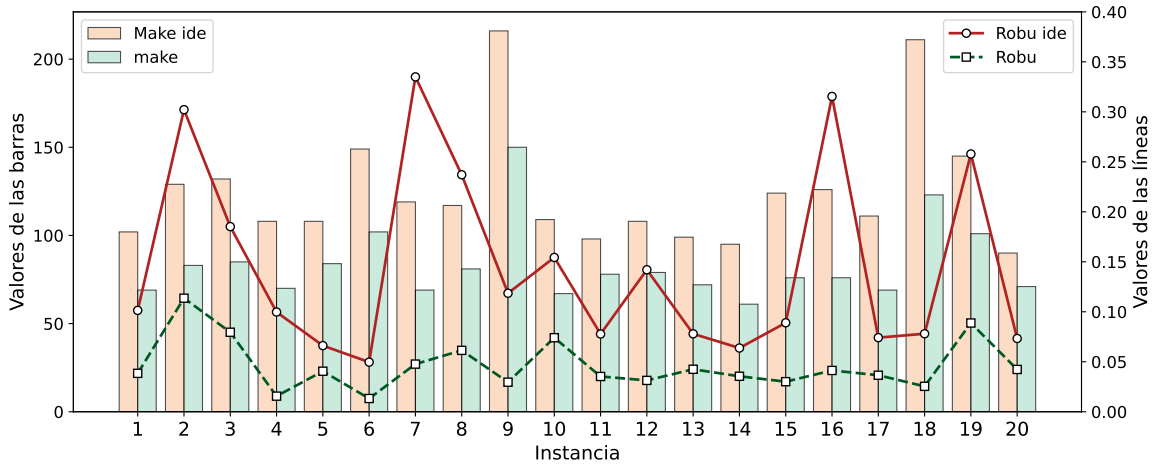

Figure 15: Comparación resultados obtenidos versus resultados de la programación base ideal para el conjunto de instancias $j60$.

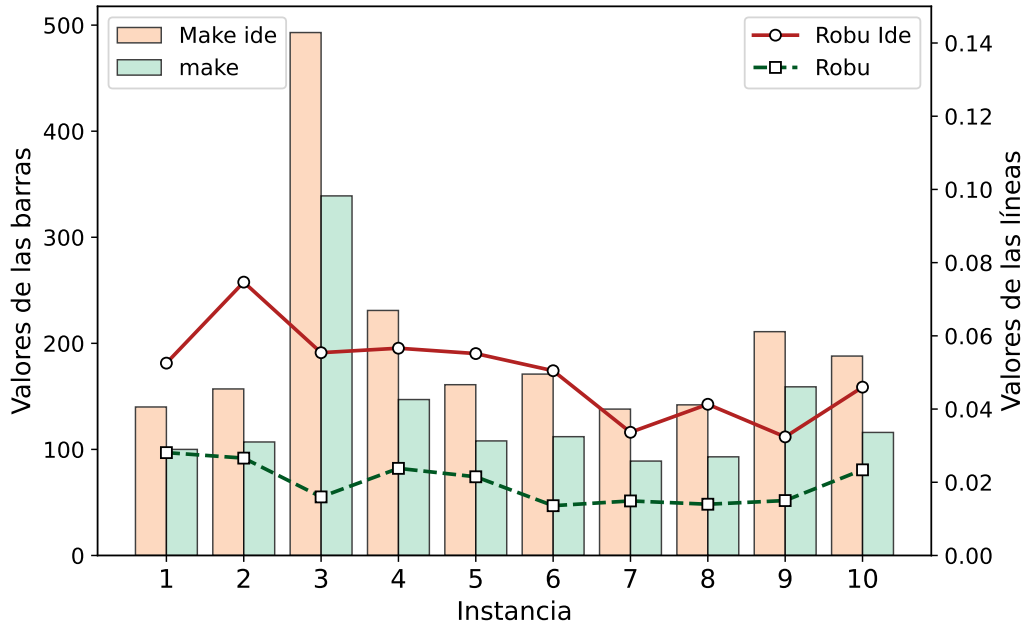

Figure 18: Comparación resultados obtenidos versus resultados de la programación base ideal para el conjunto de instancias $j120$.

Estos resultados refuerzan la tendencia observada en el conjunto $j30$, donde las soluciones con mayor makespan tienden a acercarse más a la robustez ideal, aunque sin alcanzarla completamente. Sin embargo, aunque el makespan y la robustez son directamente proporcionales, es decir, a mayor makespan mayor robustez, no son comparables en términos de unidades. Una pequeña variación positiva en el makespan puede generar un incremento significativo en

la robustez. Esto se debe a que aumento en el makespan se traduce en una reprogramación de las actividades, pudiendo generar varias unidades de holguras, que al ser simultáneas pueden no ser reflejadas en la variación del makespan. Estas holguras se pueden distribuir inteligentemente entre la programación, de forma que se genere esos espacios a las actividades con mayor riesgo. De esta forma, la robustez puede aumentar en mayor medida que el makespan; haciendo difícil establecer una relación precisa entre estas dos medidas, pues es dependiente de las características propias del problema y de la solución evaluada.

6.4.3 Análisis de resultados: metodología propuesta vs solución vanilla

En esta sección se comparan los resultados de la metodología propuesta con los de la denominada solución vanilla. Es decir aquella solución obtenida con el GA pero sobre los datos originales, sin cambios ni escenarios, es decir el problema en su versión determinista. El análisis tiene en cuenta el sacrificio en makespan necesario para mejorar la robustez (metodología propuesta), en contraste con no incluir la información del componente aleatorio (solución vanilla).

En la Fig. 19 se observa la relación entre sacrificio y mejora para el conjunto de instancias $j30$. Como se mencionó en la sección anterior, no es posible determinar con exactitud en qué medida una unidad de makespan contribuye al aumento de la robustez, pero para este conjunto de instancias en particular, el promedio de sacrificio de makespan es del 6.29%, mientras que la mejora en robustez alcanza un promedio del 148%. Cabe notar que, el cálculo del promedio de mejora en robustez excluye las instancias 5, 7, 9, 10, 14, 17, 18, 22, 25, 27, 28, 34, 37 y 40, que representan el 35% del total de instancias. Esto se debe a que en estos problemas la robustez en la solución vanilla es 0 unidades, lo que implica un porcentaje de mejora infinito frente a cualquier valor de robustez de la solución propuesta mayor a 0. En estos casos, la robustez promedio obtenida de la propuesta fue 0.059. Estas instancias tampoco se incluyeron en las gráficas de mejora - sacrificio (figuras 19, 20 y 21).

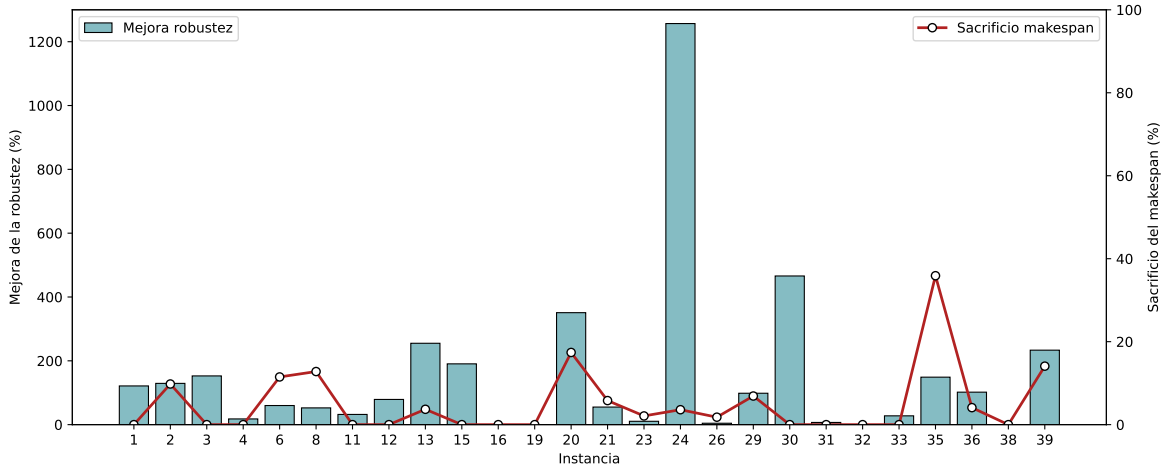

Figure 19: Comparación de resultados: metodología propuesta versus solución vanilla para el conjunto de instancias $j30$.

Por otro lado, se observa que el 25% de las instancias no presentó sacrificio en el makespan

(0%). Dentro de este grupo, una instancia registró un aumento infinito en la robustez. Excluyéndola, el promedio de mejora en la robustez para estos casos fue del 121.54%. Esto indica que la metodología propuesta fue capaz de obtener soluciones más robustas que la solución vanilla sin afectar la calidad del makespan en un 25% de los casos. Esto se logra mediante una reprogramación y asignación de recursos considerando la generación y distribución inteligente de las holguras. En el 10% de los casos, no se logró aumentar la robustez; sin embargo, tampoco hubo incremento en el makespan, lo que significa que se obtuvo la misma solución que en la versión vanilla. Adicionalmente, se destaca el caso particular de la instancia 24, en la que se logró un aumento en la robustez del 1256% con un sacrificio en el makespan de solo 3.57%.

La Fig. 20 muestra el aumento de robustez de la propuesta vs sacrificio de makespan de la solución vanilla para el conjunto $j60$. El 10% de las instancias presentó una mejora infinita en la robustez con un sacrificio de makespan de solo 6.81%, lo que implica que las soluciones vanilla obtuvieron una robustez de 0. Estas instancias (9 y 17) fueron excluidas del promedio global y de la figura. De forma general, en el 20% de las instancias se logró una mejoría en la robustez del 55.64% sin sacrificar makespan, mientras que en el 15% los resultados fueron iguales, es decir, no hubo ni aumento en la robustez ni sacrificio en el makespan. En promedio, el sacrificio de makespan para el conjunto $j60$ fue del 7.23%, con una mejora promedio en la robustez del 81.9%.

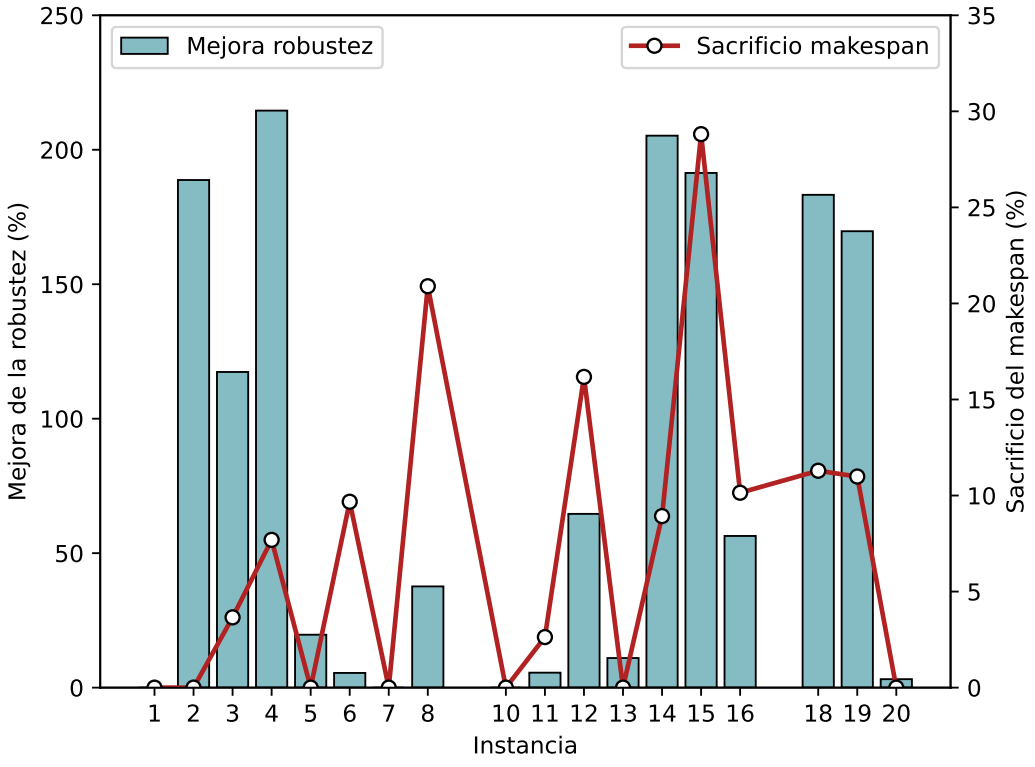

Figure 20: Comparación de resultados: metodología propuesta versus solución vanilla para el conjunto de instancias $j60$.

Por otra parte, la Fig. 21 muestra la comparación de la metodología propuesta vs la solución vanilla para el conjunto $j120$. Particularmente, la instancia 8 alcanzó un incremento del 54.48% en la robustez de la solución con respecto a la versión vanilla, sin ningún sacrificio en el makespan (0%). Implicando que, en el 10% de los casos, se logró mejorar la robustez sin afectar la calidad del makespan. El comportamiento de las demás instancias es consistente con los demás conjuntos ($j30$ y $j60$), mostrando mejora significativas en la robustez con apenas sacrificio en el makesman. En resumen, el promedio de sacrificio de makespan fue del 8.04%, mientras que la mejora en la robustez alcanzó un 130.33%.

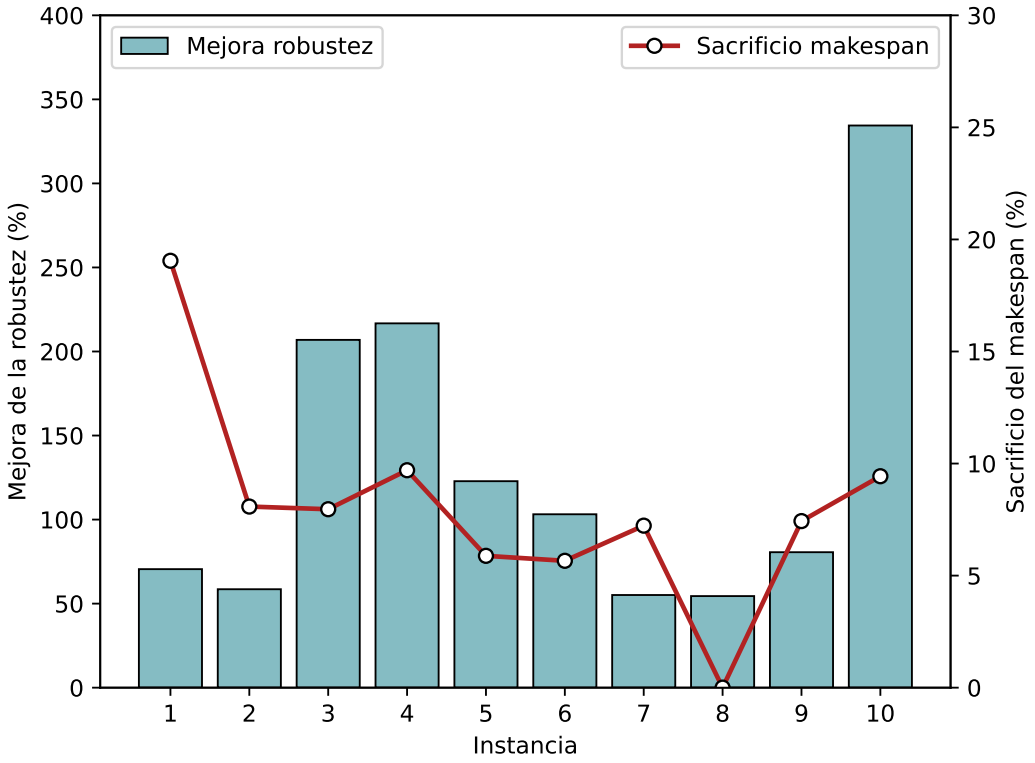

Figure 21: Comparación de resultados: metodología propuesta versus solución vanilla para el conjunto de instancias $j120$.

En general, para todos los conjuntos de instancias, los resultados muestran que la metodología propuesta permite mejorar significativamente la robustez en comparación con la solución vanilla, con un sacrificio moderado en el makespan. En particular, se identificaron casos en los que la robustez aumentó sin afectar el makespan, lo que resalta la eficiencia del enfoque. Además, la exclusión de instancias con robustez nula en la solución vanilla evitó distorsiones en el análisis, pero son casos que muestran un aumento de robustez respecto a una solución que no tiene valor en unidades de robustez. Estos hallazgos respaldan la efectividad de la metodología para obtener soluciones más robustas minimizando en lo posbile el makespan.

6.4.4 Análisis de resultados: relación entre holgura-Robustez

En esta sección se analiza la relación entre la holgura total en la programación y la robustez de la solución obtenida con la metodología propuesta. En la literatura, generalmente, se asume que un incremento en la holgura de las actividades conlleva una mayor capacidad de absorción de perturbaciones, lo que resulta en una programación más robusta. No obstante, este trabajo evidencia que dicha relación no es necesariamente directa. La distribución de holguras juega un papel crucial, ya que su impacto en la robustez depende de si estas se asignan a actividades sujetas a variabilidad; pues holguras en actividades cuya duración es determinista no contribuyen a mejorar la estabilidad de la programación frente a perturbaciones. También se debe evaluar si las holguras son excesivas para un nivel de servicio esperado, pues estos excesos no aportan significativamente a la robustez.

Para evaluar esta relación, se compararon los cambios en holgura y robustez en los conjuntos de instancias analizados. En el conjunto $j30$, solo el 40% de las instancias presentó un comportamiento coherente entre ambas métricas, es decir, cuando la holgura se incrementó, la robustez también lo hizo, y cuando la holgura disminuyó, la robustez se redujo, aunque no en la misma medida. En el 60% restante, esta correlación no se mantuvo, lo que sugiere que la holgura no es un predictor absoluto de robustez. Para los conjuntos $j60$ y $j120$, la consistencia en la relación entre holgura y robustez fue del 60%, lo que indica una mayor alineación entre ambas variables en comparación con el conjunto $j30$. Sin embargo, aún se observa un porcentaje significativo de instancias en las que la holgura no impacta directamente en la robustez, lo que refuerza la hipótesis de que su distribución estratégica de la holgura es más relevante que su cantidad total.

Las figuras 23, 22 y 24 muestran la evolución de la holgura y la robustez en cada conjunto de instancias, evidenciando que un incremento en la holgura no siempre se traduce en una mejora en la robustez. Un ejemplo claro de esta tendencia se observa en el conjunto $j30$ (Fig. 23), particularmente en las instancias 8 y 9, donde la suma total de holguras se redujo de 110 a 21, lo que representa una disminución del 80.91%, mientras que la robustez aumentó de 0.075 a 0.112, reflejando un incremento del 49.33%.

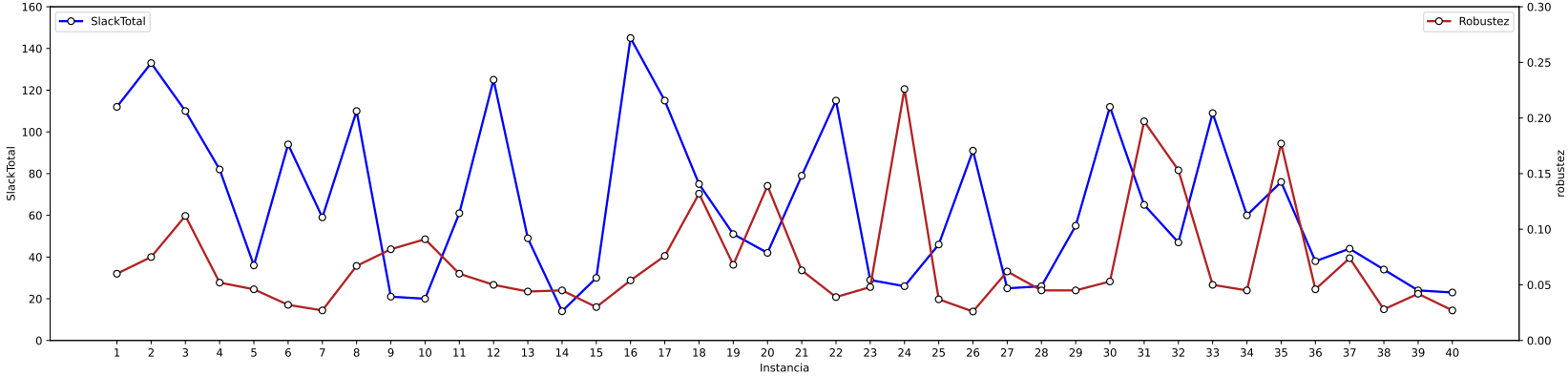

Figure 23: Comportamiento de holgura total y robustez para el conjunto de instancias $j30$.

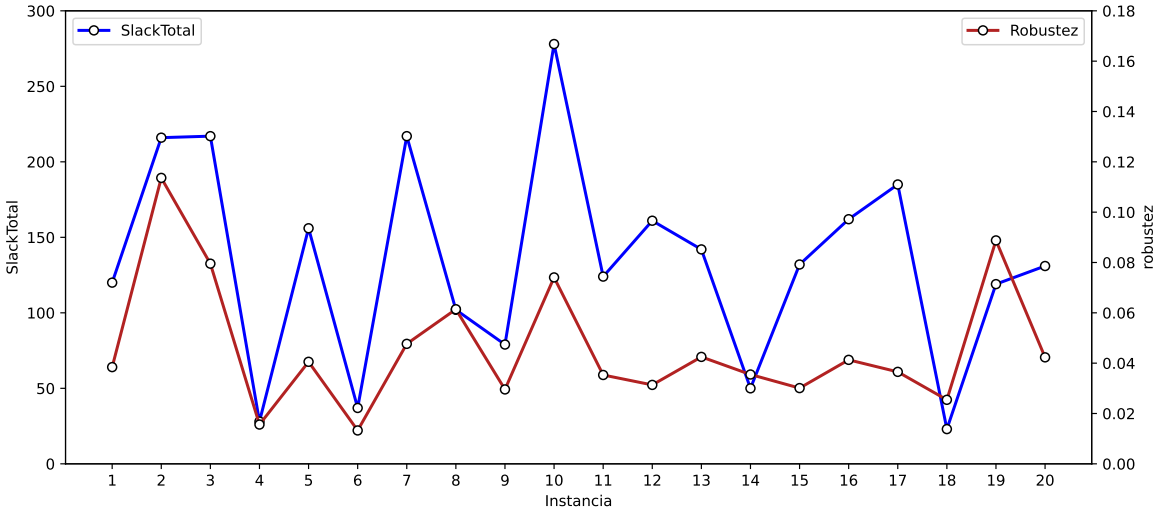

Figure 22: Comportamiento de holgura total y robustez para el conjunto de instancias $j60$.

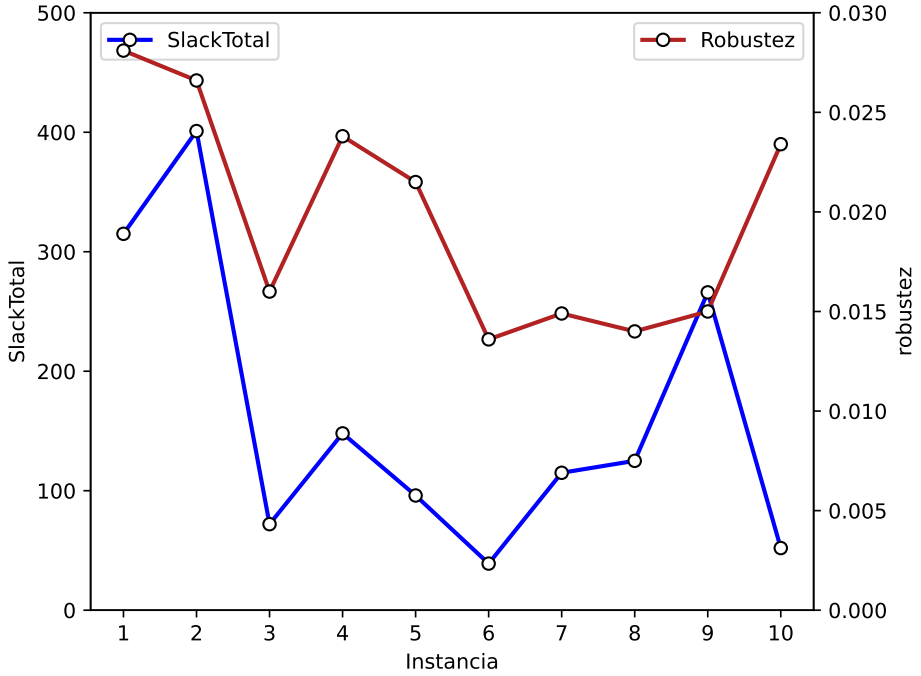

Figure 24: Comportamiento de holgura total y robustez para el conjunto de instancias $j120$.

Estos resultados demuestran la necesidad de enfoques de programación que optimicen la asignación de holguras considerando la susceptibilidad de las actividades a perturbaciones y la variación en magnitud a la que pueden estar expuestas, en lugar de solo incrementar la holgura de manera uniforme en la programación. Así, los resultados refuerzan la idoneidad y pertinencia de la metodología desarrollada en este trabajo.

7 Conclusions

En esta investigación se abordó el Resource Constrained Project Scheduling bajo incertidumbre (SRCPSP), donde las duraciones de las actividades tienen un comportamiento aleatorio según una función de distribución de probabilidad. El enfoque de solución es proactivo en las decisiones, es decir que se busca una programación base (una solución), antes de la ocurrencia de los eventos aleatorios (las disrupciones en las duraciones de las actividades). Ademas, en lugar de considerar un enfoque multi-objetivo tradicional, se plantea determinar un nuevo indicador de robustez de una solución. Este medida se interpreta como la capacidad de una programación base para absorber cambios inesperados en la duración de las actividades.

Así, la contribución de este documento es triple: primero se propone una nueva medida de robustez basada en la entropía que incorpora elementos topográficos de la red, elementos operativos dependientes de la secuencia (como las holguras), y el uso de recursos de las actividades. Segundo se diseña una metodología de solución general y extrapolable a problemas de programación de actividades; esta metodología consiste de dos etapas: (1) se generan múltiples escenarios del SRCPSP a partir de la modelación probabilística de las duraciones de actividades susceptibles a riesgos y se resuelve cada escenario en su versión determinista (RCPSP), mediante un Genetic Algorithm (GA) que minimiza el makespan. La finalidad es encontrar una colección de Activity Lists (AL) en el espacio de soluciones que serán las candidatas para ser la solución del sistema. (2) las soluciones obtenidas son decodificadas y evaluadas con la medida de robustez propuesta, considerando las duraciones originales del problema para garantizar una evaluación independiente de los escenarios simulados, seleccionando como solución al sistema, aquella AL con mayor robustez. Finalmente, la tercera contribución consiste en la adaptación de la librería de casos de prueba PSPLIB, para incluir el componente bajo incertidumbre de la duración, pero implementando un modelo realista de riesgos en la ejecución de las actividades; esto con el propósito de tener una base de datos de casos de prueba para futuras comparaciones entre metodologías de solución.

La metodología propuesta y la medida de robustez se validaron en la nueva librería propuesta. Los resultados, además de mostrar la eficiencia del GA propuesto, sugieren que las soluciones de mayor robustez (ganadoras entre los escenarios) no siempre son aquellas que tiene mayor suma total de holguras, como se sugiere en la literatura. Esto se debe a que la robustez general de la programación depende de ciertas actividades que son más relevantes que otras, esta importancia relativa está sujeta a la propia secuenciación de actividades en una solución, y a la estructura de sucesores del problema. Esta robustez también depende de que el número de periodos de holgura sea significativo, pues un valor muy bajo no sería suficiente para absolver cambios inesperados y uno muy alto podría implicar un desperdicio de recursos. Por otra parte, en algunas soluciones vanilla se logró aumentar la robustez sin afectar el makespan, lo que indica que una adecuada reorganización de la programación puede mejorar la robustez sin comprometer la calidad de la solución en términos de tiempo. En conclusión, los resultados sugieren que la combinación de la nueva métrica de robustez y la metodología de solución que incluye al GA propuesto no solo mejora la capacidad de gestión de proyectos en condiciones de incertidumbre, sino que también proporciona una herramienta valiosa para la toma de decisiones en la planificación y ejecución de proyectos bajo escenarios realistas en general.

Para futuras investigaciones, se sugiere explorar la aplicación de la métrica de robustez

desarrollada y la metodología de solución no solo en problemáticas similares como la versión multimodal del RCPSP o la variante que incluye un requerimiento de recursos variable en el tiempo, sino también en otros sistemas de programación de tareas tales como el Job Shop o el Flow shop. Además, se podría considerar la integración de otras fuentes de incertidumbre, como cambios en los recursos disponibles o en los requisitos del proyecto. Asimismo, se podría explorar la posibilidad determinar qué metaheurísticas tiene mejores resultados en la programación bajo incertidumbre, mediante técnicas avanzadas de hibridación con otros métodos. Adicionalmente, como una extensión de este trabajo, se podrían generar escenarios de riesgos y establecer un indicador de rendimiento para evaluar la eficiencia de la solución obtenida. Esto permitiría analizar el comportamiento de la programación ante diferentes escenarios adversos, proporcionando una validación rigurosa de diferentes métricas de robustez.

Acknowledgements

This work was supported by the MinCiencias Colombia under Grant BPIN 2022000100068.

Disclosure statement

No potential conflict of interest was reported by the author(s).

Data availability statement

The data that supports the findings of this study are available at https://github.com/dmorill/Robust_PSPLIB.

References

[1] Dmitry Arkhipov, Olga Battaïa, Julien Cegarra, and Alexander Lazarev. Operator assignment problem in aircraft assembly lines: a new planning approach taking into account economic and ergonomic constraints. *Procedia CIRP*, 76:63–66, 2018. ISSN 2212-8271. doi:10.1016/j.procir.2018.01.020.

[2] Xiaojin Zhang, Shuang Ma, and Songlin Chen. Healthcare service configuration based on project scheduling. *Advanced Engineering Informatics*, 43:101039, 2020. ISSN 1474-0346. doi:10.1016/j.aei.2020.101039.

[3] Mohammad Khamechian and Matthew E H Petering. A mathematical modeling approach to university course planning. *Computers & Industrial Engineering*, 168:107855, 2022. ISSN 0360-8352. doi:10.1016/j.cie.2021.107855.

[4] Dmitry Arkhipov, Olga Battaïa, Julien Cegarra, and Alexander Lazarev. Work planning in low-volume assembly lines under ergonomic constraints. *Procedia CIRP*, 72:786–789, 2018. ISSN 2212-8271. doi:10.1016/j.procir.2018.03.019.

[5] Daniel Morillo Torres, Luis Moreno, and Javier Díaz. Metodologías Analíticas y Heurísticas para la Solución del Problema de Programación de Tareas con Recursos Restringidos (RCPSP): una revisión. Parte 1. *Ingeniería y Ciencia*, 10:247–271, jan 2014.

[6] Chris Potts and Vitaly Strusevich. Fifty years of scheduling: A survey of milestones. *Journal of the Operational Research Society*, 60:S41–S68, feb 2009. doi:10.1057/jors.2009.2.

[7] Sanderson C Vanucci, Eduardo G Carrano, Rafael Bicalho, and Ricardo H C Takahashi. A modified NSGA-II for the Multiobjective Multi-mode Resource-Constrained Project Scheduling Problem. In *2012 IEEE Congress on Evolutionary Computation*, pages 1–7, 2012. doi:10.1109/CEC.2012.6256616.

[8] Salim Rostami, Stefan Creemers, and Roel Leus. New strategies for stochastic resource-constrained project scheduling. *Journal of Scheduling*, 21(3):349–365, 2018. ISSN 1099-1425. doi:10.1007/s10951-016-0505-x.

[9] Farhad Habibi, Farnaz Barzinpour, and Seyed Sadjadi. Resource-constrained project scheduling problem: review of past and recent developments[1] F. Habibi, F. Barzinpour, and S. Sadjadi, "Resource-constrained project scheduling problem: review of past and recent developments," J. Proj. Manag., vol. 3, pp. 55. *Journal of Project Management*, 3:55–88, jan 2018. doi:10.5267/j.jpm.2018.1.005.

[10] Daniel Morillo Torres, Federico Barber, and Miguel Salido. A new model and metaheuristic approach for the energy-based resource-constrained scheduling problem. *Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture*, 233:095440541771173, jan 2017. doi:10.1177/0954405417711734.

[11] Matthew Bold and Marc Goerigk. A compact reformulation of the two-stage robust resource-constrained project scheduling problem. *Computers & Operations Research*, 130:105232, 2021. ISSN 0305-0548. doi:10.1016/j.cor.2021.105232.

[12] Mojtaba Moradi, Ashkan Hafezalkotob, and Vahidreza Ghezavati. Robust resource-constrained project scheduling problem of the project's subcontractors in a cooperative environment under uncertainty: Social complex construction case study. *Computers & Industrial Engineering*, 133:19–28, 2019. ISSN 0360-8352. doi:10.1016/j.cie.2019.04.046.

[13] Christian Artigues. On the strength of time-indexed formulations for the resource-constrained project scheduling problem. *Operations Research Letters*, 45(2):154–159, 2017. ISSN 0167-6377. doi:10.1016/j.orl.2017.02.001.

[14] Yanting Wang, Zhengwen He, Louis-Phillipe Kerkhove, and Mario Vanhoucke. On the performance of priority rules for the stochastic resource constrained multi-project scheduling problem. *Computers & Industrial Engineering*, 114:223–234, 2017. ISSN 0360-8352. doi:10.1016/j.cie.2017.10.021.

[15] Sönke Hartmann and Dirk Briskorn. A survey of variants and extensions of the resource-constrained project scheduling problem. *European Journal of Operational Research*, 207 (1):1–14, 2010. ISSN 0377-2217. doi:10.1016/j.ejor.2009.11.005.

[16] A Alan B Pritsker, Lawrence J Waiters, and Philip M Wolfe. Multiproject Scheduling with Limited Resources: A Zero-One Programming Approach. *Management Science*, 16 (1):93–108, 1969. doi:10.1287/mnsc.16.1.93.

[17] J Blazewicz, J K Lenstra, and A.H.G.Rinnooy Kan. Scheduling subject to resource constraints: classification and complexity. *Discrete Applied Mathematics*, 5(1):11–24, 1983. ISSN 0166-218X. doi:10.1016/0166-218X(83)90012-4.

[18] Lin-Lin Xie, Yajiao Chen, Sisi Wu, Rui-Dong Chang, and Yilong Han. Knowledge extraction for solving resource-constrained project scheduling problem through decision tree. *Engineering, Construction and Architectural Management*, ahead-of-print(ahead-of-print), jan 2023. ISSN 0969-9988. doi:10.1108/ECAM-04-2022-0345.

[19] Carlo Vercellis. Constrained multi-project plannings problems: A Lagrangean decomposition approach. *European Journal of Operational Research*, 78(2):267–275, 1994. ISSN 0377-2217. doi:10.1016/0377-2217(94)90389-1.

[20] Vincent Y X Chen. A 0–1 goal programming model for scheduling multiple maintenance projects at a copper mine. *European Journal of Operational Research*, 76(1):176–191, 1994. ISSN 0377-2217. doi:10.1016/0377-2217(94)90015-9.

[21] Richard F Deckro, E P Winkofsky, John E. Hebert, and Roger Gagnon. A decomposition approach to multi-project scheduling. *European Journal of Operational Research*, 51(1): 110–118, 1991. ISSN 0377-2217. doi:10.1016/0377-2217(91)90150-T.

[22] Willy Herroelen. Project Scheduling—Theory and Practice. *Production and Operations Management*, 14:413 – 432, mar 2005. doi:10.1111/j.1937-5956.2005.tb00230.x.

[23] Fred. Glover and Gary A. Kochenberger. *Handbook of Metaheuristics*, volume 57. Springer US, Boston, MA, 1 edition, 2003. ISBN 978-1-4020-7263-5. doi:10.1007/b101874.

[24] Jose Torres-Jiménez and Juan Pavón. Applications of metaheuristics in real-life problems. *Progress in Artificial Intelligence*, 2(4):175–176, 2014. ISSN 2192-6360. doi:10.1007/s13748-014-0051-8.

[25] Mario Brcic, Damir Kalpic, and Krešimir Fertalj. Resource Constrained Project Scheduling under Uncertainty: A Survey. feb 2012.

[26] Maciej Laszczyk and Paweł B Myszkowski. Improved selection in evolutionary multi–objective optimization of multi–skill resource–constrained project scheduling problem. *Information Sciences*, 481:412–431, 2019. ISSN 0020-0255. doi:10.1016/j.ins.2019.01.002.

[27] M.A. Al-Fawzan and Mohamed Haouari. A bi-objective model for robust resource-constrained project scheduling. *International Journal of Production Economics*, 96(2): 175–187, 2005. ISSN 0925-5273. doi:10.1016/j.ijpe.2004.04.002.

[28] Hamidreza Maghsoudlou, Behrouz Afshar-Nadjafi, and Seyed Taghi Akhavan Niaki. A multi-objective invasive weeds optimization algorithm for solving multi-skill multi-mode resource constrained project scheduling problem. *Computers & Chemical Engineering*, 88:157–169, 2016. ISSN 0098-1354. doi:10.1016/j.compchemeng.2016.02.018.

[29] Christian Blum, Jakob Puchinger, Günther R Raidl, and Andrea Roli. Hybrid metaheuristics in combinatorial optimization: A survey. *Applied Soft Computing*, 11(6): 4135–4151, 2011. ISSN 1568-4946. doi:10.1016/j.asoc.2011.02.032.

[30] Robert Pellerin, Nathalie Perrier, and François Berthaut. A survey of hybrid metaheuristics for the resource-constrained project scheduling problem. *European Journal of Operational Research*, 280(2):395–416, 2020. ISSN 0377-2217. doi:10.1016/j.ejor.2019.01.063.

[31] Willy Herroelen and Roel Leus. Project scheduling under uncertainty: Survey and research potentials. *European Journal of Operational Research*, 165(2):289–306, 2005. ISSN 0377-2217. doi:10.1016/j.ejor.2004.04.002.

[32] Hani El Sakkout and Mark Wallace. Probe Backtrack Search for Minimal Perturbation in Dynamic Scheduling. *Constraints*, 5(4):359–388, 2000. ISSN 1572-9354. doi:10.1023/A:1009856210543.

[33] Kevin M. Calhoun, Richard F. Deckro, James T. Moore, James W. Chrissis, and John C. Van Hove. Planning and re-planning in project and production scheduling. *Omega*, 30 (3):155–170, jun 2002. ISSN 0305-0483. doi:10.1016/S0305-0483(02)00024-5.

[34] Nagajyothi Virivinti and Kishalay Mitra. Fuzzy Expected Value Analysis of an Industrial Grinding Process. *Powder Technology*, 268:9–18, 2014. ISSN 0032-5910. doi:10.1016/j.powtec.2014.08.001.

[35] M Susana Moreno and Aníbal M Blanco. A fuzzy programming approach for the multi-objective patient appointment scheduling problem under uncertainty in a large hospital. *Computers & Industrial Engineering*, 123:33–41, 2018. ISSN 0360-8352. doi:10.1016/j.cie.2018.06.013.

[36] Dragan Simić, Ilija Kovačević, Vasa Svirčević, and Svetlana Simić. 50 years of fuzzy set theory and models for supplier assessment and selection: A literature review. *Journal of Applied Logic*, 24:85–96, 2017. ISSN 1570-8683. doi:10.1016/j.jal.2016.11.016.

[37] Willy Herroelen and Roel Leus. The construction of stable project baseline schedules. *European Journal of Operational Research*, 156(3):550–565, 2004. ISSN 0377-2217. doi:10.1016/S0377-2217(03)00130-9.

[38] Babak H Tabrizi and Seyed Farid Ghaderi. A robust bi-objective model for concurrent planning of project scheduling and material procurement. *Computers & Industrial Engineering*, 98:11–29, 2016. ISSN 0360-8352. doi:10.1016/j.cie.2016.05.017.

[39] Yong Ma, Zhengwen He, Nengmin Wang, and Erik Demeulemeester. Tabu search for proactive project scheduling problem with flexible resources. *Computers & Operations Research*, page 106185, 2023. ISSN 0305-0548. doi:10.1016/j.cor.2023.106185.

[40] Stijn Van de Vonder, Erik Demeulemeester, Willy Herroelen, and Roel Leus. The use of buffers in project management: The trade-off between stability and makespan. *International Journal of Production Economics*, 97(2):227–240, 2005. ISSN 0925-5273. doi:10.1016/j.ijpe.2004.08.004.

[41] Zhiqiang Ma, Erik Demeulemeester, Zhengwen He, and Nengmin Wang. A computational experiment to explore better robustness measures for project scheduling under two types of uncertain environments. *Computers & Industrial Engineering*, 131:382–390, 2019. ISSN 0360-8352. doi:10.1016/j.cie.2019.04.014.

[42] Yakup Koç, Martijn Warnier, Robert E Kooij, and Frances M T Brazier. A robustness metric for cascading failures by targeted attacks in power networks. pages 48–53, 2013. doi:10.1109/ICNSC.2013.6548709.

[43] Stijn Van de Vonder, Erik Demeulemeester, and Willy Herroelen. Proactive heuristic procedures for robust project scheduling: An experimental analysis. *European Journal of Operational Research*, 189(3):723–733, 2008. ISSN 0377-2217. doi:10.1016/j.ejor.2006.10.061.

[44] Öncü Hazır, Mohamed Haouari, and Erdal Erel. Robust scheduling and robustness measures for the discrete time/cost trade-off problem. *European Journal of Operational Research*, 207(2):633–643, 2010. ISSN 0377-2217. doi:10.1016/j.ejor.2010.05.046.

[45] Laure-Emmanuelle Drezet and Jean-Charles Billaut. PREDICTIVE AND PROACTIVE APPROACHES FOR RCPSP WITH LABOUR CONSTRAINTS. *IFAC Proceedings Volumes*, 39(3):125–130, 2006. ISSN 1474-6670. doi:10.3182/20060517-3-FR-2903.00076.

[46] Mariam Gómez Sánchez, Eduardo Lalla-Ruiz, Alejandro Fernández Gil, Carlos Castro, and Stefan Voß. Resource-constrained multi-project scheduling problem: A survey. *European Journal of Operational Research*, 309:958–976, 2023. ISSN 0377-2217. doi:10.1016/j.ejor.2022.09.033.

[47] Hédi Chtourou and Mohamed Haouari. A two-stage-priority-rule-based algorithm for robust resource-constrained project scheduling. *Computers & Industrial Engineering*, 55 (1):183–194, 2008. ISSN 0360-8352. doi:10.1016/j.cie.2007.11.017.

[48] Wendi Tian, Yan Zhao, and Erik Demeulemeester. Generating a robust baseline schedule for the robust discrete time/resource trade-off problem under work content uncertainty. *Computers & Operations Research*, 143:105795, 2022. ISSN 0305-0548. doi:10.1016/j.cor.2022.105795.

[49] Olivier Lambrechts, Erik Demeulemeester, and Willy Herroelen. A tabu search procedure for developing robust predictive project schedules. *International Journal of Production Economics*, 111:493 – 508, 2008. doi:10.1016/j.ijpe.2007.02.003. Cited by: 110; All Open Access, Green Open Access.

[50] Kiseok Keith Lee, Yeonwoo Park, and Seppe Kuehn. Robustness of microbiome function. *Current Opinion in Systems Biology*, 36:100479, 2023. ISSN 2452-3100. doi:10.1016/j.coisb.2023.100479.

[51] André T Beck, Lucas da Rosa Ribeiro, Luis G L Costa, and Mark G Stewart. Comparison of risk-based robustness indices in progressive collapse analysis of building structures. *Structures*, 57:105295, 2023. ISSN 2352-0124. doi:10.1016/j.istruc.2023.105295.

[52] Mohamad Sadeq Karimi, Ramin Mohammadi, Mehrdad Raisee, Patrick Hendrick, and Ahmad Nourbakhsh. Robust optimization of a marine current turbine using a novel robustness criterion. *Energy Conversion and Management*, 295:117608, 2023. ISSN 0196-8904. doi:10.1016/j.enconman.2023.117608.

[53] Hongzhi Cheng, Ziliang Li, Penghao Duan, Xingen Lu, Shengfeng Zhao, and Yanfeng Zhang. Robust optimization and uncertainty quantification of a micro axial compressor for unmanned aerial vehicles. *Applied Energy*, 352:121972, 2023. ISSN 0306-2619. doi:10.1016/j.apenergy.2023.121972.

[54] Jian-Wei Wang and Li-Li Rong. Cascade-based attack vulnerability on the us power grid. *Safety Science*, 47:1332–1336, 2009. ISSN 0925-7535. doi:10.1016/j.ssci.2009.02.002.

[55] Ying-Cheng Lai, Adilson Motter, and Takashi Nishikawa. *Attacks and Cascades in Complex Networks*, volume 650, pages 299–310. 2004. ISBN 978-3-540-22354-2. doi:10.1007/978-3-540-44485-5_14.

[56] Alexander E. David, Blazhe Gjorgiev, and Giovanni Sansavini. Quantitative comparison of cascading failure models for risk-based decision making in power systems. *Reliability Engineering & System Safety*, 198:106877, 2020. ISSN 0951-8320. doi:10.1016/j.ress.2020.106877.

[57] German Maya-Rodríguez, Daniel Morillo-Torres, and John Willmer Escobar. A new method for the measurement of robustness in reverse logistics supply chains based on entropy and nodal importance. *Computers & Industrial Engineering*, 183:109533, 2023. ISSN 0360-8352. doi:10.1016/j.cie.2023.109533.

[58] J.H Holland. *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence*. University of Michigan Press, 1975. ISBN 9780472084609.

[59] Monia Rekik Anas Neumann Adnene Hajji and Robert Pellerin. Genetic algorithms for planning and scheduling engineer-to-order production: a systematic review. *International Journal of Production Research*, 0:1–30, 2023. doi:10.1080/00207543.2023.2237122.

[60] Joanna Ochelska-Mierzejewska, Aneta Poniszewska-Marańda, and Witold Marańda. Selected genetic algorithms for vehicle routing problem solving. *Electronics*, 10, 2021. ISSN 2079-9292. doi:10.3390/electronics10243147.

[61] Mohammed Farsi, Doreswamy Hosahalli, B R Manjunatha, Ibrahim Gad, El-Sayed Atlam, Althobaiti Ahmed, Ghada Elmarhomy, Mahmoud Elmarhoumy, and Osama A Ghoneim. Parallel genetic algorithms for optimizing the sarima model for better forecasting of the ncdc weather data. *Alexandria Engineering Journal*, 60:1299–1316, 2021. ISSN 1110-0168. doi:10.1016/j.aej.2020.10.052.

[62] Luigi Rarità, Ivanka Stamova, and Stefania Tomasiello. Numerical schemes and genetic algorithms for the optimal control of a continuous model of supply chains. *Applied Mathematics and Computation*, 388:125464, 2021. ISSN 0096-3003. doi:10.1016/j.amc.2020.125464.

[63] Ahmed Aidy, Mohammed Rady, Ibrahim Mohsen Mashhour, and Sameh Youssef Mahfouz. Structural design optimization of flat slab hospital buildings using genetic algorithms. *Buildings*, 12, 2022. ISSN 2075-5309. doi:10.3390/buildings12122195.

[64] Roger Resmini, Lincoln Silva, Adriel S Araujo, Petrucio Medeiros, Débora Muchaluat-Saade, and Aura Conci. Combining genetic algorithms and svm for breast cancer diagnosis using infrared thermography. *Sensors*, 21, 2021. ISSN 1424-8220. doi:10.3390/s21144802.

[65] Sönke Hartmann. A competitive genetic algorithm for resource-constrained project scheduling. *Naval Research Logistics (NRL)*, 45(7):733–750, 1998. doi:10.1002/(SICI)1520-6750(199810)45:7<733::AID-NAV5>3.0.CO;2-C.

[66] Kusol Pimapunsri, Darawan Weeranant, and Andreas Riel. Genetic algorithms for the resource-constrained project scheduling problem in aircraft heavy maintenance. 11 2022. doi:10.48550/arXiv.2208.07169.

[67] Jia Liu, Yisheng Liu, Ying Shi, and Jian Li. Solving resource-constrained project scheduling problem via genetic algorithm. *Journal of Computing in Civil Engineering*, 34: 4019055, 2020. doi:10.1061/(ASCE)CP.1943-5487.0000874.

[68] Vahid Shahhosseini Mohammad Reza Afshar and Mohammad Hassan Sebt. A genetic algorithm with a new local search method for solving the multimode resource-constrained project scheduling problem. *International Journal of Construction Management*, 22, 2022. doi:10.1080/15623599.2019.1623992.

[69] Daniel Morillo, Federico Barber, and Miguel A. Salido. Chromosome mutation vs. gene mutation in evolutive approaches for solving the resource-constrained project scheduling problem (rcpsp). *Recent Trends and Future Technology in Applied Intelligence. Lecture Notes in Computer Science*, pages 601–612, 2018. doi:10.1007/978-3-319-92058-0_58.

[70] J J M Mendes, J F Gonçalves, and M G C Resende. A random key based genetic algorithm for the resource constrained project scheduling problem. *Computers & Operations Research*, 36:92–109, 2009. ISSN 0305-0548. doi:10.1016/j.cor.2007.07.001.

[71] Lawrence Davis. Job shop scheduling with genetic algorithms. pages 136–140, 1985. ISBN 9781315799674.

[72] Roubila Lilia Kadri and Fayez F Boctor. An efficient genetic algorithm to solve the resource-constrained project scheduling problem with transfer times: The single mode case. *European Journal of Operational Research*, 265:454–462, 2018. ISSN 0377-2217. doi:10.1016/j.ejor.2017.07.027.

[73] Lihua He and Yajun Zhang. Bi-objective optimization of rcpsp under time-of-use electricity tariffs. *KSCE Journal of Civil Engineering*, 26:4971–4983, 2022. ISSN 1226-7988. doi:10.1007/s12205-022-0095-4.

[74] Hongbo Li, Hanyu Zhu, Linwen Zheng, and Fang Xie. Software project scheduling under activity duration uncertainty. *Annals of Operations Research*, 338(1):477–512, 2024. ISSN 1572-9338. doi:10.1007/s10479-023-05343-0.

[75] Wanjun Liu, Jingwen Zhang, Mario Vanhoucke, and Weikang Guo. Resource allocation models and heuristics for the multi-project scheduling with global resource transfers and local resource constraints. *Computers & Industrial Engineering*, 200:110843, 2025. ISSN 0360-8352. doi:10.1016/j.cie.2024.110843.

[76] Chao Lu, Biao Zhang, Liang Gao, Jin Yi, and Jianhui Mou. A knowledge-based multiobjective memetic algorithm for green job shop scheduling with variable machining speeds. *IEEE Systems Journal*, 16:844–855, 2022. doi:10.1109/JSYST.2021.3076481.

[77] Satyajit Padhy and Jerry Chou. Mirage: A consolidation aware migration avoidance genetic job scheduling algorithm for virtualized data centers. *Journal of Parallel and Distributed Computing*, 154:106–118, 2021. ISSN 0743-7315. doi:10.1016/j.jpdc.2021.03.004.

[78] Ahmad Hassanat, Khalid Almohammadi, Esra'a Alkafaween, Eman Abunawas, Awni Hammouri, and V B Surya Prasath. Choosing mutation and crossover ratios for genetic algorithms—a review with a new dynamic approach. *Information*, 10, 2019. ISSN 2078-2489. doi:10.3390/info10120390.

[79] F F BOCTOR. Resource-constrained project scheduling by simulated annealing. *International Journal of Production Research*, 34:2335–2351, 1996. ISSN 0020-7543. doi:10.1080/00207549608905028.

[80] Babak Abbasi, Shahram Shadrokh, and Jamal Arkat. Bi-objective resource-constrained project scheduling with robustness and makespan criteria. *Applied Mathematics and Computation*, 180:146–152, 2006. ISSN 0096-3003. doi:10.1016/j.amc.2005.11.160.

[81] Jian Xiong, Ying wu Chen, Ke wei Yang, Qing song Zhao, and Li ning Xing. A hybrid multiobjective genetic algorithm for robust resource-constrained project scheduling with stochastic durations. *Mathematical Problems in Engineering*, page 786923, 2012. ISSN 1024-123X. doi:10.1155/2012/786923.

[82] Rainer Kolisch and Arno Sprecher. PSPLIB - A project scheduling problem library: OR Software - ORSEP Operations Research Software Exchange Program. *European Journal of Operational Research*, 96(1):205–216, 1997. ISSN 0377-2217. doi:10.1016/S0377-2217(96)00170-1.

[83] Néstor Raúl Ortíz-Pimiento and Francisco Javier Diaz-Serna. An optimization model to solve the resource constrained project scheduling problem rcpsp in new product development projects • . *DYNA*, 87:179–188, 1 2020. ISSN 2346-2183. doi:10.15446/dyna.v87n212.81269.

[84] Rahmat Rabet, Seyed Mojtaba Sajadi, and Mahshid Tootoonchy. A hybrid metaheuristic and simulation approach towards green project scheduling. *Annals of Operations Research*, 2024. ISSN 1572-9338. doi:10.1007/s10479-024-06291-z.

[85] Amir Hossein Hosseinian and Vahid Baradaran. A two-phase approach for solving the multi-skill resource-constrained multi-project scheduling problem: a case study in construction industry. *Engineering, Construction and Architectural Management*, 30:321–363, 1 2023. ISSN 0969-9988. doi:10.1108/ECAM-07-2019-0384.

[86] Md. Asadujjaman, Humyun Fuad Rahman, Ripon K Chakrabortty, and Michael J Ryan. An immune genetic algorithm for solving npv-based resource constrained project scheduling problem. *IEEE Access*, 9:26177–26195, 2021. doi:10.1109/ACCESS.2021.3057366.

[87] Hatem M H Saad, Ripon K Chakrabortty, Saber Elsayed, and Michael J Ryan. Quantum-inspired genetic algorithm for resource-constrained project-scheduling. *IEEE Access*, 9: 38488–38502, 2021. doi:10.1109/ACCESS.2021.3062790.

[88] Wanlin Liu, Haotian Zhang, Yumeng Chen, Chunli Qu, and Jingwen Zhang. Simulation-based hybrid genetic algorithms for the stochastic multi-mode resource-constrained project scheduling problem with minimized financial risk. *Applied Soft Computing*, 161: 111716, 2024. ISSN 1568-4946. doi:https://doi.org/10.1016/j.asoc.2024.111716.

[89] Firoz Mahmud, Forhad Zaman, Ali Ahrari, Ruhul Sarker, and Daryl Essam. Genetic algorithm for singular resource constrained project scheduling problems. *IEEE Access*, 9:131767–131779, 2021. doi:10.1109/ACCESS.2021.3114702.

[90] Jingyu Luo, Mario Vanhoucke, and José Coelho. Automated design of priority rules for resource-constrained project scheduling problem using surrogate-assisted genetic programming. *Swarm and Evolutionary Computation*, 81:101339, 2023. ISSN 2210-6502. doi:10.1016/j.swevo.2023.101339.