Universidad de Concepción
Facultad de Ingeniería
Departamento de Ingeniería Industrial

Modelamiento estocástico y solución para el *scheduling* de proyectos de investigación en bases antárticas chilenas

(*Stochastic modelling and solution for scheduling research projects at Chilean Antarctic stations*)

Autor:
Mauricio Maximiliano Vega Hidalgo

Profesora Guía:
Dra. Lorena Pradenas Rojas

Profesor Coguía:
Dr. Víctor Parada Daza

Tesis presentada a la
DIRECCIÓN DE POSTGRADO
UNIVERSIDAD DE CONCEPCIÓN

para optar al grado de
DOCTOR EN INGENIERÍA INDUSTRIAL

Concepción, marzo de 2026

A mi amada esposa Carmen Gloria.
A mis padres Hilda y Heriberto.
A mi hermano Camilo.
Gracias por su amor y su apoyo en esta larga travesía.
Gracias Señor por tu fidelidad y por las personas que pones en mi camino.

Agradecimientos

Quiero expresar mi más sincero agradecimiento a todas las personas e instituciones que, de una u otra manera, hicieron posible el desarrollo y culminación de esta tesis.

En primer lugar, agradezco profundamente a mi profesora guía, Dra. Lorena Pradenas Rojas, por su permanente orientación, rigurosidad académica, confianza y generosidad a lo largo de este proceso. Su apoyo fue fundamental no solo para la realización de esta investigación, sino también para mi formación como investigador. Del mismo modo, agradezco a mi profesor coguía, Dr. Víctor Parada Daza, por sus valiosos aportes, su disposición constante y sus observaciones siempre oportunas, las que contribuyeron significativamente a fortalecer este trabajo.

Agradezco también a los integrantes de la comisión de evaluación, Dr. Juan Antonio Carrasco, Dr. Carlos Contreras, Dr. Rodrigo Linfati, y Dr. Héctor Cancela, por el tiempo dedicado a la revisión de esta tesis, por sus comentarios, sugerencias y observaciones, que sin duda enriquecieron la calidad académica del manuscrito.

Deseo expresar un reconocimiento especial al Dr. Héctor Cancela, por su acogida y apoyo durante mi pasantía de investigación en Montevideo, y al Dr. Lluis Plá, por su gentil recepción en Lleida. Ambas experiencias fueron altamente formativas y enriquecedoras, tanto en lo académico como en lo personal, y contribuyeron de manera significativa al desarrollo de esta investigación.

A mis colegas y compañeros, Carlos Contreras, Rosa Medina, Magdalena Jensen, Dagoberto Cifuentes, Patricio Salas, Patricio Sáez, Beatriz Millán, así como a otros colegas del Departamento de Ingeniería Industrial de la Universidad de Concepción, les agradezco sinceramente por su compañerismo, sus consejos, conversaciones e intercambio de ideas, que hicieron de este camino una experiencia más enriquecedora y llevadera.

Extiendo también mi agradecimiento a los pares evaluadores de los artículos derivados de esta tesis, cuyas observaciones y sugerencias permitieron mejorar sustantivamente la calidad de los trabajos desarrollados. De igual forma, agradezco a los pares que asistieron a mis presentaciones en las distintas conferencias en las que participé, por sus preguntas, comentarios y apreciaciones, que contribuyeron a ampliar la reflexión sobre los resultados de esta investigación.

Finalmente, agradezco a la Agencia Nacional de Investigación y Desarrollo (ANID), a través de la Subdirección de Capital Humano, por el financiamiento otorgado mediante la Beca de Doctorado Nacional 2021, Folio 21210451, apoyo que resultó fundamental para llevar adelante mis estudios doctorales y el desarrollo de esta tesis.

Índice general

Modelamiento estocástico y solución para el *scheduling* de proyectos de investigación en bases antárticas chilenas

Mauricio Maximiliano Vega Hidalgo

Concepción, marzo de 2026

Profesora guía: Dra. Lorena Pradenas Rojas
Profesor coguía: Dr. Víctor Parada
Programa: Doctorado en Ingeniería Industrial

Resumen

Esta tesis desarrolla un marco integrado de modelamiento y resolución para apoyar la toma de decisiones en la planificación de expediciones científicas en la Antártica chilena, abordando de forma conjunta la selección y la programación de proyectos bajo restricciones logísticas extremas, recursos renovables distribuidos por base, relaciones de precedencia y tiempos de traslado interbase. El trabajo se estructura en dos ejes: (i) el Problema de Selección y Programación de Proyecots de Investigación en Múltiples Bases Antárticas (RPSAP, por sus siglas en inglés) determinístico, formulado como un modelo de programación lineal entera mixta que maximiza el beneficio neto (ingresos menos costos de recursos), y (ii) su extensión estocástica de dos etapas, RPSAP estocástico (SRPSAP, por sus siglas en inglés), donde la primera etapa decide la selección y la segunda calendariza por escenario frente a incertidumbre en la duración de proyectos y con incorporación de ventanas temporales. Para el RPSAP se proponen y evalúan metaheurísticas *iterated local search* (ILS), *variable neigborhood search* (VNS) y *simulated annealing* (SA) en 480 instancias, observándose un patrón consistente: ILS tiende a liderar en calidad y SA en rapidez, con ventajas claras en escalas medianas y grandes respecto de enfoques exactos. Para el SRPSAP se diseña un método de descomposición *integer L-shaped* (IL-S) con estructura maestro–subproblemas, un esquema híbrido IL-S+SA que aproxima las soluciones de segunda etapa mediante SA para acelerar las iteraciones, y una alternativa SA para el problema estocástico (SA-SP) basada en *list-scheduling* como solucionador directo o mecanismo de inicialización. La validación empírica incluye la adaptación de 36 instancias para el caso estocástico, lo que permite comparar enfoques exactos, metaheurísticos y de descomposición bajo condiciones controladas. En síntesis, los modelos exactos resultan adecuados como referencia y para instancias pequeñas, mientras que la obtención de planes oportunos en escalas realistas requiere estrategias metaheurísticas e híbridas.

Stochastic modelling and solution for scheduling research projects at Chilean Antarctic stations

Mauricio Maximiliano Vega Hidalgo

Concepción, March 2026

Advisor: Dr. Lorena Pradenas Rojas
Co-advisor: Dr. Víctor Parada
Program: PhD. in Industrial Engineering

Abstract

This thesis develops an integrated modelling and resolution framework to support decision-making in the planning of scientific expeditions in Chilean Antarctica, jointly addressing project selection and scheduling under extreme logistical constraints, renewable resources distributed across stations, precedence relationships, and inter-stations transfer times. The work is structured around two axes: (i) the deterministic Research Project Selection and Scheduling in Multiple Antarctic Station Problem (RPSAP), formulated as a mixed integer programming model that maximises net benefit (revenue minus resource costs), and (ii) its two-stage stochastic extension Stochastic RPSAP (SRPSAP), where the first stage decides the selection and the second schedules by scenario in the face of uncertainty in project duration and with the incorporation of time windows. For the RPSAP, iterated local search (ILS), variable neighborhood search (VNS), and simulated annealing (SA) metaheuristics are proposed and evaluated in 480 instances, with a consistent pattern observed: ILS tends to lead in quality and SA in speed, with clear advantages in medium and large scales over exact approaches. For the SRPSAP, an integer L-shaped decomposition (IL-S) method with a master-subproblem structure is designed, a hybrid IL-S+SA scheme that approximates the second stage solutions using SA to accelerate iterations, and a SA for the stochastic problem (SA-SP) alternative based on list-scheduling as a direct solver or initialisation mechanism. Empirical validation includes adapting 36 instances to the stochastic case, enabling comparisons of exact, metaheuristic, and decomposition approaches under controlled conditions. In summary, exact models are suitable as a reference and for small instances, while obtaining timely plans at realistic scales requires metaheuristic and hybrid strategies.

Capítulo 1

Introducción

La ciencia en la región antártica es fundamental para comprender el clima global de la Tierra, el rol de los seres humanos en el cambio climático y los efectos del calentamiento global en el aumento del nivel del mar (Kennicutt et al, 2015; Seroussi, 2019). Además, en el Polo Sur se realizan investigaciones en diversas áreas de las ciencias naturales, como la astronomía, la biología y la geología (Kennicutt et al, 2014). En consecuencia, se espera que, año a año, aumente el número de investigadores interesados en realizar investigación científica en la Antártica. No obstante, debido a las condiciones climáticas extremas del continente, los recursos disponibles para la investigación son limitados. Adicionalmente, el Tratado Antártico (ATS, 1959) obliga a los países participantes a velar por la conservación del medio ambiente en esta zona. Por lo tanto, resulta fundamental gestionar adecuadamente las investigaciones en la Antártica, considerando los recursos disponibles, a fin de minimizar la intervención humana y maximizar el impacto científico.

Los países participantes del Tratado Antártico disponen de organismos encargados de promover y gestionar la investigación en el continente. En Chile, país que cuenta geográficamente con uno de los principales accesos al territorio antártico, el Instituto Antártico Chileno (INACH), creado por el gobierno chileno en 1963, es el organismo encargado de organizar las expediciones de investigación en la Antártica durante el verano. Desde 1964, el INACH convoca a concurso a los investigadores interesados en desarrollar sus estudios en esta región, quienes deben elaborar una propuesta de investigación, completar la formulación científica y, en caso de requerirlo, realizar actividades en territorio antártico, además de presentar los formularios de apoyo logístico y de cumplimiento ambiental. De este modo, los proyectos se adjudican considerando: (i) el mérito científico de los investigadores postulantes, con base en su productividad científica (publicaciones aceptadas); (ii) la calidad, relevancia y viabilidad de la propuesta; y (iii) la factibilidad de ejecución del proyecto, evaluada a partir de la política antártica nacional, la disponibilidad de recursos logísticos y el impacto potencial de las actividades sobre el medio ambiente antártico.

El INACH dispone de diversos recursos logísticos y financieros para la ejecución de los proyectos seleccionados. En la Antártica, el INACH opera tres bases científicas propias, ubicadas en distintos puntos del territorio antártico chileno (COMNAP, 2017). Cada base cuenta con infraestructura y facilidades, tales como camas, provisiones, laboratorios para el análisis de muestras y equipos de comunicación, además de equipamiento y vehículos para la exploración. Adicionalmente, existen laboratorios instalados en bases pertenecientes a organismos de las Fuerzas Armadas de Chile, con menor capacidad que la de las bases científicas del INACH, las cuales se encuentran a disposición de los investigadores. Asimismo, el INACH dispone de la lancha Karpuj y del rompehielos Almirante Viel, de reciente incorporación, equipados con camas, laboratorios y equipos para la toma de muestras, permitiendo realizar investigación durante varios días. Finalmente, el INACH gestiona los traslados marítimos y aéreos de personas, equipamiento y muestras entre el continente y la Antárti-

ca, así como al interior de esta, sujetos al itinerario propuesto por los organismos de las Fuerzas Armadas o por empresas privadas.

En este trabajo se aborda el problema estocástico de selección y programación de proyectos de investigación en múltiples bases antárticas (SRPSAP, por sus siglas en inglés). Este problema considera un conjunto de proyectos de investigación postulantes, los cuales deben ser seleccionados y programados en alguna de las bases antárticas disponibles dentro de un horizonte temporal acotado. Cada proyecto presenta una duración incierta, lo que da lugar a distintos escenarios. Además, cada proyecto requiere ciertos recursos para su ejecución, considerando tanto los disponibles en las distintas bases antárticas como aquellos que pueden desplazarse entre bases. Dadas las características propias de cada investigación, los proyectos seleccionados deben programarse dentro de determinadas ventanas de tiempo. Adicionalmente, los resultados de algunos proyectos pueden ser relevantes para la ejecución de otros; por ende, existen relaciones de precedencia entre proyectos. En consecuencia, se debe seleccionar un subconjunto de proyectos de investigación y, para cada escenario, programarlos respetando las ventanas temporales requeridas y asignarlos a bases que dispongan de los recursos necesarios para su ejecución.

El SRPSAP es afín al problema estocástico de selección y programación de carteras de proyectos (SPPSSP, por sus siglas en inglés), ampliamente estudiado en la literatura (Tofighian and Naderi, 2015; Liu et al, 2025). En estos problemas se construye una cartera a partir de un conjunto de proyectos candidatos que requieren recursos disponibles en cantidades finitas para su ejecución. Sin embargo, en el SRPSAP los recursos se distribuyen entre distintas bases antárticas, con la posibilidad de transportarlos entre ellas. En consecuencia, la selección de proyectos debe considerar, simultáneamente, su programación en una base que disponga de recursos suficientes para la ejecución de cada proyecto.

Si bien la literatura ha avanzado de manera significativa en la selección estocástica de carteras de proyectos, incorporando incertidumbre en costos, beneficios y probabilidades de éxito, y la programación estocástica con recursos, enfatizando la factibilidad temporal bajo duraciones inciertas, dichos cuerpos de investigación tienden a desarrollarse de forma relativamente disociada. En particular, los modelos de selección tienden representar los efectos temporales y de recursos mediante aproximaciones agregadas, mientras que los modelos de programación rara vez integran decisiones de selección de primera etapa, asignación multi-sitio y tiempos de transferencia entre bases con impacto directo en precedencias y disponibilidad efectiva de recursos. En consecuencia, persiste una brecha sustantiva: la ausencia de un marco integrado que unifique, en una formulación estocástica de dos etapas, la selección y la programación por escenario en un contexto multi-base con restricciones logísticas propias de campañas antárticas (p. ej., ventanas de tiempo, precedencias y traslados).

Para abordar esta brecha, la tesis desarrolla un marco integrado de modelamiento y métodos de solución orientado a apoyar la toma de decisiones en la planificación de expediciones científicas en la Antártica chilena. En primer lugar, se consolida la base determinística mediante la formulación del RPSAP y el diseño de estrategias metaheurísticas que permiten obtener soluciones de alta calidad en escalas donde los enfoques exactos se vuelven poco competitivos. En segundo lugar, se formula el SRPSAP como una extensión estocástica de dos etapas, en la que la primera etapa determina la selección de proyectos y la segunda etapa resuelve su programación por escenario ante incertidumbre en las duraciones, incorporando adicionalmente ventanas temporales. Sobre dicha formulación, se proponen y evalúan enfoques de resolución complementarios que explotan la estructura del problema, incluyendo un esquema de descomposición *integer L-shaped* (IL-S) y mecanismos híbridos con metaheurísticas para acelerar la aproximación de la solución de segunda etapa, además de un esquema metaheurístico directo para el caso estocástico; todo ello se valida empíricamente mediante conjuntos de instancias determinísticas y su adaptación al caso estocástico, habilitando comparaciones controladas entre métodos exactos, de descomposición y metaheurísticos.

1.1. Hipótesis de investigación

Una adecuada gestión de los recursos disponibles integrada a la selección de proyectos de investigación y su programación en las múltiples bases antárticas, permite una planificación confiable de las expediciones al territorio antártico en condiciones de duración incierta, aumentando el impacto científico y disminuyendo la intervención humana.

1.2. Objetivo general

Formular y evaluar métodos exactos y metaheurísticos para apoyar la toma de decisiones en la planificación de expediciones a la Antártica, abordando la selección y programación de proyectos de investigación en múltiples bases bajo restricciones de recursos, tiempos de transporte, relaciones de precedencia y horizonte de planificación, e incorporando incertidumbre en la duración de los proyectos, de manera de maximizar el beneficio esperado.

1.3. Objetivos específicos

1. **Formular** un modelo de programación lineal entera mixta determinístico para la selección y programación de proyectos de investigación en un horizonte de planificación acotado, incorporando restricciones de recursos, tiempos de transporte entre múltiples bases, relaciones de precedencia y duración de los proyectos, de modo de maximizar el beneficio económico-científico, modelado como la diferencia entre los ingresos asociados a los proyectos seleccionados y el costo total de utilización de recursos.

2. **Extender** el modelo determinístico a un modelo estocástico de dos etapas, en el cual la selección de proyectos se determine en la primera etapa y la programación de los proyectos seleccionados se resuelva en la segunda etapa para cada escenario inducido por la duración incierta de los proyectos.

3. **Diseñar e implementar** un conjunto de metaheurísticas orientadas a obtener soluciones de alta calidad en tiempos computacionales razonables, aplicables tanto a la formulación determinística como a la estocástica.

4. **Diseñar e implementar** un método de descomposición IL-S para reducir los tiempos de cómputo en la resolución del modelo estocástico de dos etapas.

5. **Construir y caracterizar** instancias de prueba basadas en la literatura y en datos reales, con el propósito de evaluar comparativamente el desempeño de los modelos determinístico y estocástico, las metaheurísticas propuestas y el método de descomposición IL-S, en términos de calidad de solución y esfuerzo computacional.

1.4. Estructura de la tesis

El resto de la tesis se estructura de la siguiente manera:

- En el **Capítulo 2** se analiza el RPSAP determinístico, que integra restricciones de recursos renovables distribuidos entre bases, relaciones de precedencia, tiempos de traslado entre bases, y la duración junto con los requerimientos de recursos asociados a cada proyecto. El objetivo es seleccionar y calendarizar los proyectos que maximicen el beneficio total, definido como la

diferencia entre el ingreso de los proyectos seleccionados y el costo atribuible al uso de recursos. La formulación se representa mediante un modelo de programación entera mixta. Debido a su alta complejidad computacional, se proponen tres metaheurísticas orientadas a obtener soluciones de alta calidad en tiempos de cómputo razonables. Para evaluar el desempeño de los métodos propuestos, se construyeron 480 instancias de prueba basadas en la literatura. Los resultados muestran que, en instancias de mayor escala, las metaheurísticas superan al modelo exacto en términos de desempeño global. En particular, ILS destaca por alcanzar soluciones cercanas al óptimo, mientras que SA logra soluciones competitivas en tiempos inferiores a 90 segundos. El contenido de este capítulo fue publicado en el artículo "*Optimizing research in Antarctica: a novel approach to project selection and scheduling across multiple stations*", en la revista "*Journal of the Operational Research Society*" (Vega-Hidalgo et al, 2026).

- En el **Capítulo 3** se desarrolla el SRPSAP como una extensión estocástica del RPSAP. En esta variante, la duración de los proyectos es incierta, lo que induce la generación de múltiples escenarios. Adicionalmente, se incorporan ventanas de tiempo para el inicio de los proyectos. El problema se modela mediante programación estocástica de dos etapas: en la primera etapa se decide la selección de proyectos y, en la segunda, se programa el conjunto seleccionado en cada escenario. Con el fin de disminuir los tiempos de cómputo y explotar la estructura del modelo, se propone la descomposición *integer L-shaped* (IL-S), que separa el problema en un maestro encargado de la selección y en subproblemas orientados a la programación bajo cada escenario. Dada la complejidad intrínseca de la etapa de programación y los hallazgos del **Capítulo 2**, se plantea resolver los subproblemas mediante SA (IL-S+SA). Asimismo, se propone un esquema basado en SA para resolver directamente el problema estocástico. Para evaluar los cuatro enfoques, se extrajeron y adaptaron 36 instancias de prueba desde Vega-Hidalgo et al (2026). Los resultados indican que IL-S e IL-S+SA superan al modelo matemático en términos de *gap* relativo. Por su parte, SA produce soluciones competitivas en tiempos de cómputo reducidos, superando, en algunas repeticiones, el *gap* obtenido por IL-S e IL-S+SA. El trabajo de este capítulo se plasmó en el artículo "*Integrated Stochastic Project Selection and Scheduling for Antarctic Research under Uncertain Project Durations*", enviado a la revista "*Computers & Industrial Engineering*".

- En el **Capítulo 4** se discuten de manera integrada los principales resultados de ambos artículos, enfatizando cómo el modelamiento y los desarrollos metaheurísticos propuestos para el RPSAP, junto con la evidencia empírica reportada en el **Capítulo 2**, constituyen la base metodológica para abordar el SRPSAP.

- Finalmente, el **Capítulo 5** presenta las conclusiones del estudio, sintetizando las contribuciones principales, sus implicancias teóricas y prácticas, las limitaciones identificadas y posibles líneas de investigación futura.

Capítulo 2

Optimizing research in Antarctica: a novel approach to project selection and scheduling across multiple stations

JOURNAL OF THE OPERATIONAL RESEARCH SOCIETY
2026, VOL. 77, NO. 1, 286–312
https://doi.org/10.1080/01605682.2025.2495766

RESEARCH ARTICLE

Optimizing research in Antarctica: a novel approach to project selection and scheduling across multiple stations

Mauricio Vega-Hidalgo[a], Lorena Pradenas[a,b] and Víctor Parada[c]

[a]Departamento de Ingeniería Industrial, Universidad de Concepción, Concepción, Chile; [b]Instituto Sistemas Complejos de Ingeniería, Santiago, Chile; [c]Departamento de Ingeniería Informática, Universidad de Santiago de Chile, Santiago, Chile

ABSTRACT

Antarctica's unique research environment necessitates innovative project selection and scheduling strategies to maximize scientific output while addressing logistical, environmental, and resource constraints. This study introduces a novel framework for the Research Project Selection and Scheduling in Multiple Antarctic Stations Problem (RPSAP), formulated as a mixed-integer programming model. The model incorporates resource-sharing constraints, station-specific capacities, transportation delays, and sustainability considerations. Given the problem's NP-hard complexity, three metaheuristic methods—Iterated Local Search (ILS), Variable Neighborhood Search (VNS), and Simulated Annealing (SA)—were developed to efficiently solve large-scale instances. Metaheuristics demonstrated robust performance through extensive computational experiments involving 480 test instances across 60 classes. The ILS consistently outperformed others in solution quality and scalability, while SA offered competitive results in execution time. Results reveal that the metaheuristics are superior to exact methods in handling large problem sizes, with optimality achieved only for small instances using the proposed exact model. This research bridges the gap between theoretical optimization models and practical applications, offering decision-making tools for research managers to enhance resource utilization, minimize ecological impacts, and prioritize high-impact projects.

ARTICLE HISTORY
Received 30 July 2024
Accepted 15 April 2025

KEYWORDS
Project scheduling; Project selection; Multisite context; Mixed-integer programming; Metaheuristic; Antarctic science

1. Introduction

Antarctica, often called the Earth's last frontier, offers an unparalleled environment for scientific research crucial to understanding global climate change and its impact on ecosystems (Kennicutt et al., 2015). The continent holds a treasure trove of data vital for comprehending ecological and atmospheric dynamics. Despite its immense potential, Antarctica presents unique logistical and environmental challenges that are more extreme than those in other research settings. The severe climate, remoteness of research stations, and limited resources necessitate meticulous planning, efficient resource allocation, and strategic scheduling to ensure project success. These constraints significantly amplify the complexity of decision-making, emphasizing the pressing need for innovative and adaptive methodologies to address these multifaceted challenges and optimize project selection and scheduling effectively.

Globally, 76 scientific stations, managed by 26 countries under the Antarctic Treaty, serve as hubs for interdisciplinary research encompassing fields such as glaciology, marine biology, climate science, and geophysics (COMNAP, 2017; SAT, 1959). These stations provide essential infrastructure for conducting cutting-edge research and foster international collaboration in some of the most inhospitable environments on Earth. Each project competing for inclusion in a research season's portfolio must carefully balance its potential scientific contributions against the imperative of environmental sustainability. This dual objective entails minimizing the human footprint, mitigating risks such as fuel spills and ecological disruptions, and maximizing the scientific outcomes to ensure impactful results (Kennicutt et al., 2014). Adding to this complexity are critical considerations such as the high costs of transportation, stringent precedence constraints for projects requiring sequential execution, and the efficient reuse of limited resources across multiple geographically dispersed stations. These challenges necessitate the development of advanced optimization strategies tailored to address Antarctic research operations' unique and multifaceted requirements.

This research addresses the Research Project Selection and Scheduling in Multiple Antarctic Stations Problem (RPSAP). The problem involves selecting and scheduling projects across multiple stations to maximize net scientific profit—the total project income minus associated costs—while adhering

CONTACT Mauricio Vega-Hidalgo mauriciovega@udec.cl Departamento de Ingeniería Industrial, Universidad de Concepción, Casilla 160-C, Correo 3, Concepción, 4070043, Chile

Supplemental data for this article can be accessed online at https://doi.org/10.1080/01605682.2025.2495766.

to operational constraints. Specifically, the study proposes and evaluates:

- A Comprehensive Mixed-Integer Programming (MIP) Model: The model captures the essential features of RPSAP, including project precedence, station-specific resource constraints, transportation times, and environmental considerations.
- Metaheuristic-Based Solutions: Given RPSAP's NP-hard nature (Tofighian & Naderi, 2015), the research designs and evaluates multiple metaheuristics, including Iterated Local Search (ILS), Variable Neighborhood Search (VNS), and Simulated Annealing (SA), complemented by a list-scheduling-based algorithm. These methods are compared with the proposed Mixed-Integer Programming model to assess their efficiency and effectiveness, particularly for large-scale instances where exact methods struggle.

While project portfolio selection and scheduling have been extensively studied in contexts such as construction and defense (e.g., Harrison et al., 2022; Tavakolan et al., 2024), their application to Antarctic research remains uncharted. Unlike traditional settings, the RPSAP incorporates unique constraints:

- Multi-site resource allocation with transportation delays.
- Environmental and logistical challenges intrinsic to Antarctica.

The novelty of this study lies in extending the project portfolio framework to this distinct domain, offering an optimization model and computational tools. The outcomes contribute to operational research by bridging theoretical advancements with practical implications for Antarctic science logistics.

The study employed a robust experimental design to evaluate the proposed methodologies using 480 test instances, classified into 60 classes representing realistic scenarios in Antarctic research. The MIP model was implemented using a commercial solver, while the metaheuristics—ILS, VNS, and SA—were coded in Python. Each test instance was analyzed for computational efficiency and solution quality.

The findings revealed that while the MIP model performed well for small-scale instances, its computational feasibility declined as the problem increased. In contrast, the metaheuristics consistently provided high-quality solutions across all test scenarios, with ILS demonstrating superior performance in balancing accuracy and runtime. These results highlight the practical utility of metaheuristic approaches for efficiently solving large-scale RPSAP cases.

1.1. Problem description

The RPSAP arises from the need to efficiently allocate scarce resources across multiple research stations while considering diverse operational constraints. The RPSAP integrates elements of project portfolio selection and resource-constrained scheduling in a multi-site context. Key characteristics of the problem include:

- Resource Allocation: Balancing fixed and mobile resources across stations to meet project requirements.
- Precedence Constraints: Ensuring the logical sequencing of projects based on dependencies.
- Transportation Delays: Accounting for the time required to move resources and research project outcomes between stations.
- Environmental Impact: Prioritizing projects that align with sustainability goals and minimize ecological disruption.

The objective function of the RPSAP aims to maximize the net scientific profit, which is calculated as the total income generated by the selected projects minus the total costs incurred. Costs include resource utilization, transportation, and environmental mitigation measures. This function aims to achieve the dual goals of enhancing scientific output while minimizing resource expenditures and environmental impact.

This combinatorial optimization problem necessitates robust methods that can effectively navigate a vast solution space and offer practical insights to decision-makers in extreme environments.

1.2. Structure of the paper

The paper is organized as follows: Section 2 reviews relevant literature, contrasting RPSAP with traditional selection and scheduling problems. Section 3 presents the proposed MIP model. Section 4 details the metaheuristics approach and its integration with the list-scheduling-based algorithm. Section 5 presents computational experiments, which include 480 test instances classified into 60 classes. Section 6 concludes with insights into the practical implications and future research directions.

This study seeks to chart a course for more sustainable and effective scientific endeavors in extreme environments by addressing the unique demands of Antarctic research operations.

2. Literature review

This section delves into the research related to the RPSAP. The RPSAP, with its similarities to project portfolio selection and scheduling problems with

constrained resources (PPSSP), is a topic of paramount importance. The independent studies of the two subproblems of the PPSSP, namely the project portfolio selection problem (PPSP) and the resource-constrained project scheduling problem (RCPSP), have also been a subject of considerable research, underscoring their critical role in the field (Tavakolan et al., 2024).

The PPSP is an optimization problem that shares specific characteristics with the classic knapsack problem (KP) (Chang & Lee, 2012; Mavrotas et al., 2008). However, it is essential to note that the PPSP introduces unique constraints that differentiate it from the KP. For instance, project interdependency refers to the relationships and interactions between projects within the portfolio. This interdependency means that one project's selection and execution can influence, or be influenced by, other projects in the portfolio. Li et al. (2016) consider precedence relations between projects and the reinvestment of selected project profits in another project within the portfolio, while Nascimento et al. (2025) examine resource sharing between projects. Synergies are a class of project interdependencies that establish that the presence of a project affects the presence of another project (Tao et al., 2023; Tavana et al., 2020). Thus, a project can be selected (or not) regardless of whether another project is selected. Litvinchev et al. (2014) define three types of synergies: benefit synergies, resource consumption synergies, and technical synergies. Shen and Li (2023) consider the synergies generated by the sharing and inheritance of productive factors to select a project portfolio. In the RPSAP, interdependencies arise from shared resources and precedence constraints: a project can only be selected if its predecessor has been selected, and the resources it requires must be available and not already allocated to other projects.

Some studies about PPSP include a previous phase to evaluate each project. This phase involves assessing and prioritizing potential projects based on various criteria. In the research by Mavrotas et al. (2008), an augmented score strategy is employed to enhance the agreement between the final selection and the initial ranking obtained from the multicriteria approach. Chang and Lee (2012) integrate data envelopment analysis (DEA), knapsack formulation, and fuzzy set theory to solve the PPSP in the engineering, procurement, and construction industries. Tavana et al. (2013) propose a novel DEA model incorporating ambiguity and vagueness into the objective function, input, and output data. Tavana et al. (2020) propose a dynamic two-stage mathematical programming model for project evaluation and selection under uncertainty, which combines a fuzzy technique for order preference by similarity to ideal solution (TOPSIS) and a bi-objective mixed-integer linear programming formulation. Wu et al. (2021) combine risk management techniques, schedule risk analysis, and fuzzy evaluation to quantify the expected value of the project portfolio. Mavrotas and Makryvelios (2021) employ multiple criteria analysis, mathematical programming, and Monte Carlo simulation to address uncertainty in R&D project portfolio selection. Casado et al. (2022) propose a multidimensional risk evaluation with an implicit enumeration algorithm to tackle the PPSP of a natural gas pipeline. Nascimento et al. (2025) propose a TOPSIS-based framework for selecting project portfolios in the public sector. Sohrabi et al. (2024) investigated a fuzzy model for multi-criteria project portfolio selection based on a modified version of Kerre's inequality. In contrast to these studies, we do not consider a previous phase to evaluate projects. In the RPSAP, we suppose that each project has an income for this execution, and the total profit of the portfolio is given by the difference between the total income of selected projects and the total cost generated by the resources used.

The RCPSP, a complex combinatorial optimization problem, involves limited resources, activities with known durations and resource requests, and precedence relations. The primary goal of the RCPSP is to find a schedule of minimal makespan by assigning start times to activities while respecting precedence relations and resource availabilities (e.g., Artigues, 2008; Hu et al., 2019; Koné et al., 2011; Sebt et al., 2013). However, the research also explores other objectives, such as minimizing the total cost of resource use (e.g., Artigues, 2017), maximizing the net present value of the project schedule (e.g., Thiruvady et al., 2019), or bi-objective RCPSP (e.g., Pham & Huynh, 2024; Zhang et al., 2024). The variety of explored objectives underscores the multifaceted nature of RCPSP research, offering a comprehensive understanding of the diverse factors that influence project scheduling.

The literature presents a rich array of extensions of RCPSP, each contributing a unique perspective to the problem. The RCPSP with resource transfer (RCPSP-RT) closely aligns with the RPSAP. The RCPSP-RT introduces special conditions when transferring renewable resources between activities that use the same resource unit. Rostami et al. (2017) and Zhang et al. (2024) consider a transfer cost when a resource unit transfers between activities. Considering the transfer cost, the objective is to minimize the project schedule's total cost. Kadri and Boctor (2018) and Liu et al. (2023) consider transfer time between activities that use the same resource unit. Transfer times alter the start times of each activity assigned to the same resource unit as its predecessor, affecting the project's

completion time. Zhao and Xu (2022) account for transfer time to minimize the average project delay. Zhao and Xu (2021) consider transfer cost and time to minimize the sum of total transfer cost, total idle cost, and total tardiness cost. In RPSAP, transfer times occur to transport resources and project results with precedence relationships between stations.

The RPSAP shares certain similarities with the RCPSP in a multi-site context (RCPSP-MS), proposed by Mika et al. (2009), and modeled using a binary mathematical programming model by Laurent et al. (2017). In RCPSP-MS, activities such as medical exams or surgeries within a hospital network (Gourgand et al., 2015) occur at various sites. In contrast to RCPSP-RT, transfer times in RCPSP-MS are considered between sites only when two activities occur in different sites and use the same resource unit or have a precedence relationship. Various studies have proposed new approaches to solve the RCPSP-MS. Stiti and Driss (2019) propose a particle swarm optimization metaheuristic to solve this problem. Gnagi and Trautmann (2019) propose a continuous-time mixed-integer programming model for the RCPSP-MS, and Bigler et al. (2024) propose a matheuristic method based on this continuous-time model. Patoghi and Mousavi (2021) consider fuzzy uncertainty in the parameters of RCPSP-MS. In the RPSAP, projects can be scheduled across different stations, similar to the sites in RCPSP-MS.

The PPSSP involves determining which projects to include in an organization's portfolio and achieving an optimal schedule. This problem is critical for organizations with limited resources, such as budget, personnel, and equipment, who must make strategic decisions about which projects to pursue.

The primary objective of the PPSSP is to maximize the portfolio's overall value and benefit. Several studies theoretically study the maximization of the total value of the objective function (e.g., Chen et al., 2023; Harrison et al., 2021, 2022). Other research focuses on maximizing the overall benefit of selection (e.g., Bai et al., 2025; Kumar et al., 2018, 2019). Related to these concepts, the net present value (NPV) is crucial in the PPSSP. It refers to the present value of a project's expected future cash flows or benefits minus the initial investment or costs (Alinaghian et al., 2016; Li et al., 2015). Different studies aim to select a portfolio of projects that collectively maximizes the total NPV, subject to resource constraints and other considerations (e.g., Mirkhorsandi Langaroudi et al., 2023; Shafahi & Haghani, 2018; Tao & Schonfeld, 2006; Tselios et al., 2024). In contrast to the studies presented above, there is research that approaches PPSSP as a minimization problem. In Hosseininasab and Shetab-Boushehri (2015), the goal is to select and schedule urban road construction projects, minimizing the total travel time of vehicles redirected due to construction. Jovanovic et al. (2018) minimize the cost of selecting and scheduling link and intersection improvements in urban networks. Bagloee et al. (2018) minimize the total delay over phases of road construction projects, and Miralinaghi et al. (2020) minimize the overall travel delay of multi-class road projects in urban areas. In our research, the RPSAP is a maximization problem where the objective function considers the difference between the total income of the selected projects and the total cost of the resources allocated.

The different variants of the PPSSP are formulated using the characteristics and constraints of the PPSP and RCPSP subproblems. Several studies consider interdependencies (e.g., Bagloee et al., 2018; Chien & Huynh, 2018; Kumar et al., 2019; Tao & Schonfeld, 2006), reinvestment (e.g., Shafahi & Haghani, 2018; Wang, 2018) and synergies constraints (e.g., Arratia-Martinez et al., 2021; Bai et al., 2025; Şahin-Zorluoğlu & Kabak, 2022; Xiong et al., 2017) from the PPSP. Another characteristic of PPSP is the budget limitations, which are addressed by different PPSSP research (e.g., Alinezhad et al., 2022; Harrison et al., 2021; 2022; Hosseininasab et al., 2018; Liu & Wang, 2011; Miralinaghi et al., 2020; Mirkhorsandi Langaroudi et al., 2023; Şahin-Zorluoğlu & Kabak, 2022; Sarnataro et al., 2021; Shafahi & Haghani, 2018; Tofighian & Naderi, 2015; Wang, 2018). Several studies address the resource constraints that are typical limitations in the RCPSP (e.g., Alinaghian et al., 2016; Alinezhad et al., 2022; Harrison et al., 2021; Kumar et al., 2019; Liu & Wang, 2011; Shafahi & Haghani, 2018; Tavakolan et al., 2024; Tofighian & Naderi, 2015). The resource constraints consider renewable resources, i.e., resources that can be used by an activity or project when another finishes this use (e.g., Chen & Askin, 2009; Liu & Wang, 2011), and nonrenewable resources (e.g., Xiong et al., 2017). Several studies on PPSSP address precedence as another important constraint from RCPSP, defining it as the condition in which an activity or project cannot start while its predecessor is still running. (e.g., Arratia-Martinez et al., 2021; Liu & Wang, 2011; Song et al., 2021). In RPSAP, we consider renewable resources and precedence constraints between projects. Interdependencies are considered between projects with precedence relationships, i.e., a project cannot be selected if its predecessor is not selected.

Particular constraints from RCPSP are considered in specific studies of PPSSP. Zolfaghari and Mousavi (2021) and Ranjbar et al. (2022) address the multi-mode PPSSP, where each activity can have multiple execution modes, representing different ways of

utilizing resources or performing the activity. Chen and Askin (2009) consider activity preemption, i.e., each activity can be interrupted to allow the execution of another activity. In contrast to these studies, RPSAP considers a multi-site context, similar to RCPSP-MS, where stations are analogous to sites. Therefore, RPSAP is the first study to consider project portfolio selection and scheduling in a multi-site context.

The real world inspires a wide variety of PPSSP applications. In construction management, the PPSSP enhances project scheduling by optimizing resource allocation, improving decision-making, integrating advanced scheduling techniques, balancing multiple objectives, and employing sophisticated algorithms to address complex scheduling challenges (e.g., Tavakolan et al., 2024). Specifically, several applications of PPSSP consider road construction and transportation projects where the main objective is to minimize the impact of road construction on traffic (e.g., Bagloee et al., 2018; Hosseininasab & Shetab-Boushehri, 2015; Mahmoudi et al., 2019; Miralinaghi et al., 2020; Sarnataro et al., 2021; Tao & Schonfeld, 2006). In the defense sector, the PPSSP enables defense organizations to make informed investment decisions for the future defense force (e.g., Harrison et al., 2021; 2022) or inform weapon selection and planning (e.g., Xiong et al., 2017). Tselios et al. (2024) apply the PPSSP to energy projects, divided into two categories: typical energy projects, such as the construction of solar power generation plants, and auxiliary energy projects. PPSSP is relevant for organizations that invest in R&D projects, where the primary objective is to maximize the NPV of the selected portfolio while minimizing the portfolio completion time (e.g., Arratia-Martinez et al., 2021; Chien & Huynh, 2018). However, existing literature does not report any applications of the PPSSP to the selection and scheduling of research projects in the Antarctic context. Consequently, the RPSAP is the first application of PPSSP in the Antarctic context.

Table 1 compares different studies related to RPSAP, showing the similarities and differences between RPSAP, PPSSP, and its subproblems. To the best of our knowledge, this paper makes a significant contribution to the literature by proposing the application of PPSSP in an Antarctic context and considering project portfolio selection and scheduling in a multi-site context.

Table 1. Relevant studies of PPSSP and RPSAP.

	PPSSP		Objective		Multi-site
References	PPSP	RCPSP	Single	Multi	Context
Mavrotas et al. (2008)	✓		✓		
Chang and Lee (2012)	✓		✓		
Tavana et al. (2013)	✓		✓		
Litvinchev et al. (2014)	✓		✓		
Li et al. (2016)	✓		✓		
Tavana et al. (2020)	✓			✓	
Mavrotas and Makryvelios (2021)	✓		✓		
Casado et al. (2022)	✓		✓		
Tao et al. (2023)	✓			✓	
Sohrabi et al. (2024)	✓		✓		
Gourgand et al. (2015)		✓		✓	✓
Laurent et al. (2017)		✓	✓		✓
Kadri and Boctor (2018)		✓	✓		
Gnagi and Trautmann (2019)		✓	✓		✓
Patoghi and Mousavi (2021)		✓		✓	✓
Bigler et al. (2024)		✓	✓		✓
Liu et al. (2023)		✓	✓		
Zhang et al. (2024)		✓		✓	
Liu and Wang (2011)	✓	✓	✓		
Tofighian and Naderi (2015)	✓	✓		✓	
Alinaghian et al. (2016)	✓	✓	✓		
Xiong et al. (2017)	✓	✓		✓	
Hosseininasab et al. (2018)	✓	✓		✓	
Kumar et al. (2019)	✓	✓	✓		
Miralinaghi et al. (2020)	✓	✓	✓		
Arratia-Martinez et al. (2021)	✓	✓		✓	
Harrison et al. (2022)	✓	✓	✓		
Bai et al. (2025)	✓	✓	✓		
Tavakolan et al. (2024)	✓	✓		✓	
RPSAP	✓	✓	✓		✓

3. Mathematical model

3.1. Sets, parameters, and decision variables

Let $G = (V', E)$ a graph where $V' = V \cup \{0, |V|+1\}$, $V = \{1, ..., |V|\}$ represents the set of research projects (RPs), 0 and $|V|+1$ represents dummy RPs, and E is the set of arcs representing the precedence relations between RPs. $\overline{E}$ is the transitive closure of E. Let $L = \{1, ..., |L|\}$ represent the set of stations, $R = \{1, ..., |R|\}$ represent the set of types of resources, and T represents the time horizon. Let $R_k = \{1, ..., |R_k|\}$ represent the set of units of $k \in R$ type resources. For each k type, there is a subset $F_k \subseteq R_k$ of fixed units of resources, i.e., units that cannot be transported between stations. Each fixed unit $u \in F_k$ is located on the station loc_{ku}. Each research project (RP) $i \in V$ needs p_i periods of execution and needs r_{ik} units of type k resource, but dummy RPs do not need anyone. We define a transportation time between station $l \in L$ and station l' as $\delta_{ll'}$. I_i gives the income of selecting the RP i. We define for each unit u of type k resource a variable cost of use it c^v_{ku}. This cost considers consumption (e.g., food, water) and penalizations for emissions (e.g., CO_2, wastewater). In addition, we define a fixed-cost c^f_{ku} for each unit u of type k resource. For example, depending on the station location, arrival, and departure to stations have a fixed cost.

For the mathematical model, we define five classes of decision variables. We define the variables x_i, for each $i \in V'$, by Equation (1), as follows:

$$x_i = \begin{cases} 1, & \text{if RP } i \text{ is selected} \\ 0, & \text{otherwise} \end{cases} \quad (1)$$

Decision variables y_{ij} are defined for each $i, j \in V$, such as $(i, j) \notin \overline{E}$ by Equation (2) as follows:

$$y_{ij} = \begin{cases} 1, & \text{if RP } i \text{ finishes before RP } j \text{ starts} \\ 0, & \text{otherwise} \end{cases} \tag{2}$$

We define the variables r_{ik}^{u} for each $i \in V$, $k \in R$ and $u \in R_k$ by Equation (3) as follows:

$$r_{ik}^{u} = \begin{cases} 1, & \text{if unit } u \text{ of type } k \text{ resource is assigned to RP } i \\ 0, & \text{otherwise} \end{cases} \tag{3}$$

Decision variables s_{il} are defined for each $i \in V$ and $l \in L$ by Equation (4):

$$s_{il} = \begin{cases} 1, & \text{if the RP } i \text{ is scheduled in station } l \\ 0, & \text{otherwise} \end{cases} \tag{4}$$

Finally, decision variables $B_i \in \mathbb{R}_{+}^{0}$ indicate the start time of RP $i \in V'$.

3.2. Continuous-time model

In the RPSAP, the objective is to maximize the total profit of the project selection, given by the difference between the total income and the total cost of using resources. Equation (5) represents this objective.

$$\sum_{i \in V} I_i x_i - \sum_{i \in V} \sum_{k \in R} \sum_{u \in R_k} \left(p_i c_{ku}^{v} + c_{ku}^{f} \right) r_{ik}^{u} \tag{5}$$

The constraints in Equation (6) guarantee that when two RPs are in a precedence relationship, the predecessor is selected if the successor is selected.

$$x_j \leq x_i, \quad \forall (i,j) \in E \tag{6}$$

Equation (7) establishes that if two RPs in a precedence relationship are selected and scheduled, the successor starts after the predecessor finishes. Additionally, if two RPs are scheduled at different stations, these constraints require time for transportation between stations. This time is necessary to transport the result from the predecessor to the successor. In addition, the parameter M in Equation (7) considers the case when the successor is not selected, where $M \geq \sum_{i \in V} p_i + |V| \delta_{max}$ is a big number, and $\delta_{max} = \max \left\{ \delta_{ll'} : l, l' \in L \right\}$.

$$B_i + p_i + \left(s_{il} + s_{jl'} - 1 \right) \delta_{ll'} \leq B_j + M(1 - x_j), \quad \forall (i,j) \in E; l, l' \in L \tag{7}$$

The constraints in Equation (8) establish that if two non-related RPs by precedence are executed simultaneously, the same required resource unit cannot be assigned to both RPs. Conversely, these constraints stipulate that two RPs cannot be performed simultaneously when the same resource unit is assigned.

$$r_{ik}^{u} + r_{jk}^{u} \leq y_{ij} + y_{ji} + 1, \quad \forall (i,j) \notin \overline{E} : i < j; \; k \in R : r_{ik} r_{jk} > 0; u \in R_k \tag{8}$$

The constraints in Equation (9) ensure that if two non-related RPs are selected and scheduled sequentially, the successor RP begins after the predecessor RP has been completed. Additionally, if both RPs occur at different stations, these constraints require time for transportation between the stations when the same resource unit is assigned to both RPs. In addition, parameter M in Equation (9) considers the case when one RP is not selected or both RPs co-occur.

$$B_i + p_i + \left(s_{il} + s_{jl'} - 1 \right) \delta_{ll'} \leq B_j + M(1 - x_j) + M(1 - y_{ij}), \forall (i,j) \notin \overline{E} : i \neq j; l, l' \in L \tag{9}$$

The constraints in Equation (10) ensure that the end of the selected RPs is before the horizon time.

$$B_i \leq (T - p_i) x_i, \quad \forall i \in V \cup \{0\} \tag{10}$$

Equation (11) indicates that the required resources for each type are assigned to the selected RP. Conversely, if the RP is not selected, these constraints guarantee that no resources are allocated.

$$\sum_{u \in R_k} r_{ik}^{u} = r_{ik} x_i, \quad \forall i \in V; k \in R : r_{ik} > 0 \tag{11}$$

The constraints in Equation (12) ensure that if a unit of a fixed resource in a station is assigned to a selected RP, the RP will be scheduled at that station.

$$r_{ik}^{u} \leq s_{i, loc_{ku}}, \quad \forall i \in V; k \in R; u \in F_k : r_{ik} > 0 \tag{12}$$

Finally, Equation (13) ensures that only selected RPs can be scheduled precisely on a single station.

$$\sum_{l \in L} s_{il} = x_i, \quad \forall i \in V \tag{13}$$

Thus, the continuous-time model (CTM) for the RPSAP is given by:

$$(CTM) \begin{cases} \max & (5) \\ \text{s.t.} & (6) - (13) \\ & x_i \in \{0,1\}, \quad \forall i \in V' \\ & y_{ij} \in \{0,1\}, \quad \forall i,j \in V : (i,j) \notin \overline{E} \\ & r_{ik}^{u} \in \{0,1\}, \quad \forall i \in V; k \in R; u \in R_k \\ & s_{il} \in \{0,1\}, \quad \forall i \in V; l \in L \\ & B_i \geq 0, \quad \forall i \in V \end{cases}$$

3.3. Illustrative example

We consider a set of 7 RPs. Furthermore, there are two stations, a horizon time of 11 periods, and four types of resources. RPs 0 and 8 are dummy RPs. There are two units of the Type 1 resource (one unit fixed at Station 1), one of the Type 2 resource (fixed at Station 2), one of the Type 3 resource, and one of the Type 4 resource. The transportation time between stations is two periods. The requirements for each RP, including durations, precedence relations, and costs, are detailed in Figure 1.

Figure 2 shows an optimal solution for the illustrative example obtained by CPLEX. Because RPs 1 and 3 require two units of Type 1 resource, these are scheduled in Station 1, where one unit of this type is fixed. RPs 2, 4, 5, and 6 are also selected and scheduled in Station 2. RPs 2, 4, and 6 require one unit of Type 2 resource, which is fixed at Station 2. Time is required to transport the project results between RP 1 and RPs 2 and 4 because they are scheduled at different stations. Similarly, transportation time is necessary between RPs 3 and 4. Because RPs 3 and 4 and RPs 3 and 5 are assigned the same resource unit and scheduled at different sites, time is required to transport resources between these RPs. Therefore, the solution shown by the Gantt chart in Figure 2 satisfies all problem constraints.

4. Proposed metaheuristics for the RPSAP

Due to the complexity of the RPSAP, metaheuristics provide an alternative method for finding near-optimal solutions. In this paper, we compare the performance of three metaheuristics: Iterated Local Search (ILS), Variable Neighborhood Search (VNS), and Simulated Annealing (SA). For the RPSAP, each metaheuristic method considers an ordered list of projects to represent a solution. Then, a list-scheduling-based algorithm builds a schedule of selected projects from the list and calculates the value of the objective function. We have a different schedule and value of the objective function for a different-ordered list. Thus, each metaheuristic is used to find different ordered lists using the value of the objective function. Figure 3 illustrates a general scheme of the proposed method, where the list-scheduling-based algorithm is a key procedure for evaluating the current solution and allows for finding near-optimal solutions by metaheuristics.

4.1. Representation of the solution and the list-scheduling-based algorithm

We represent a solution by a list $\boldsymbol{\sigma} = (\sigma_1, \ldots, \sigma_{|V|})$ as a sequence of RPs, where σ_i represents a non-dummy RP in the i-th position of $\boldsymbol{\sigma}$. The RPs are scheduled in the order on the list, i.e., if σ_i and σ_j are selected RPs and $i < j$, then the RP σ_i start before the RP σ_j.

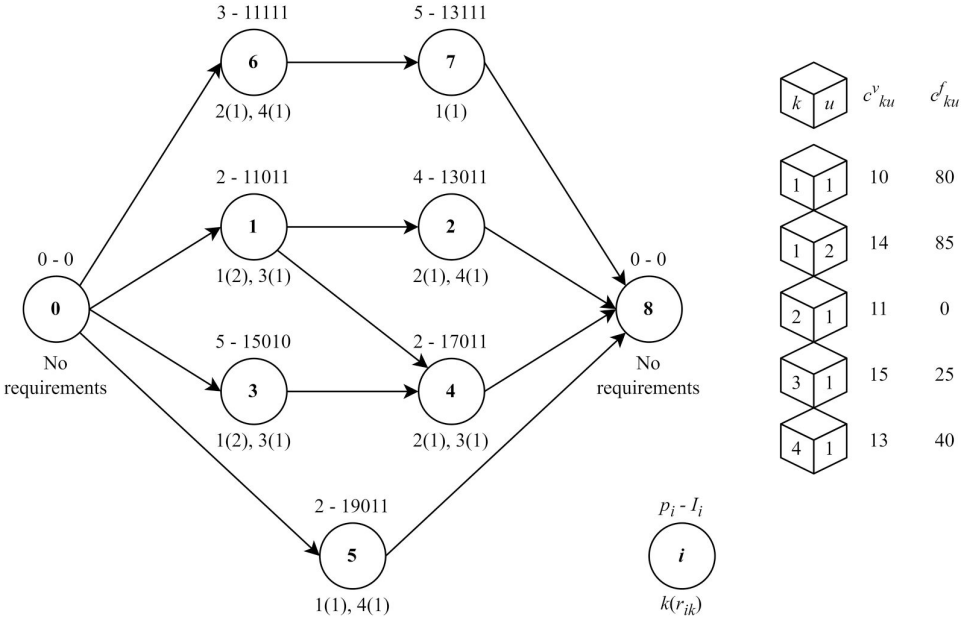

Figure 1. Data for illustrative example. Each node in the graph represents a research project, and the edges represent a precedence relation between projects. The numbers above the node represent the duration and income of the project, while the numbers below the node indicate the units (in parentheses) of each required resource type. The boxes identify the resource units available by the type (left number) and the unit (right number). The following columns indicate the variable and fixed costs of each unit.

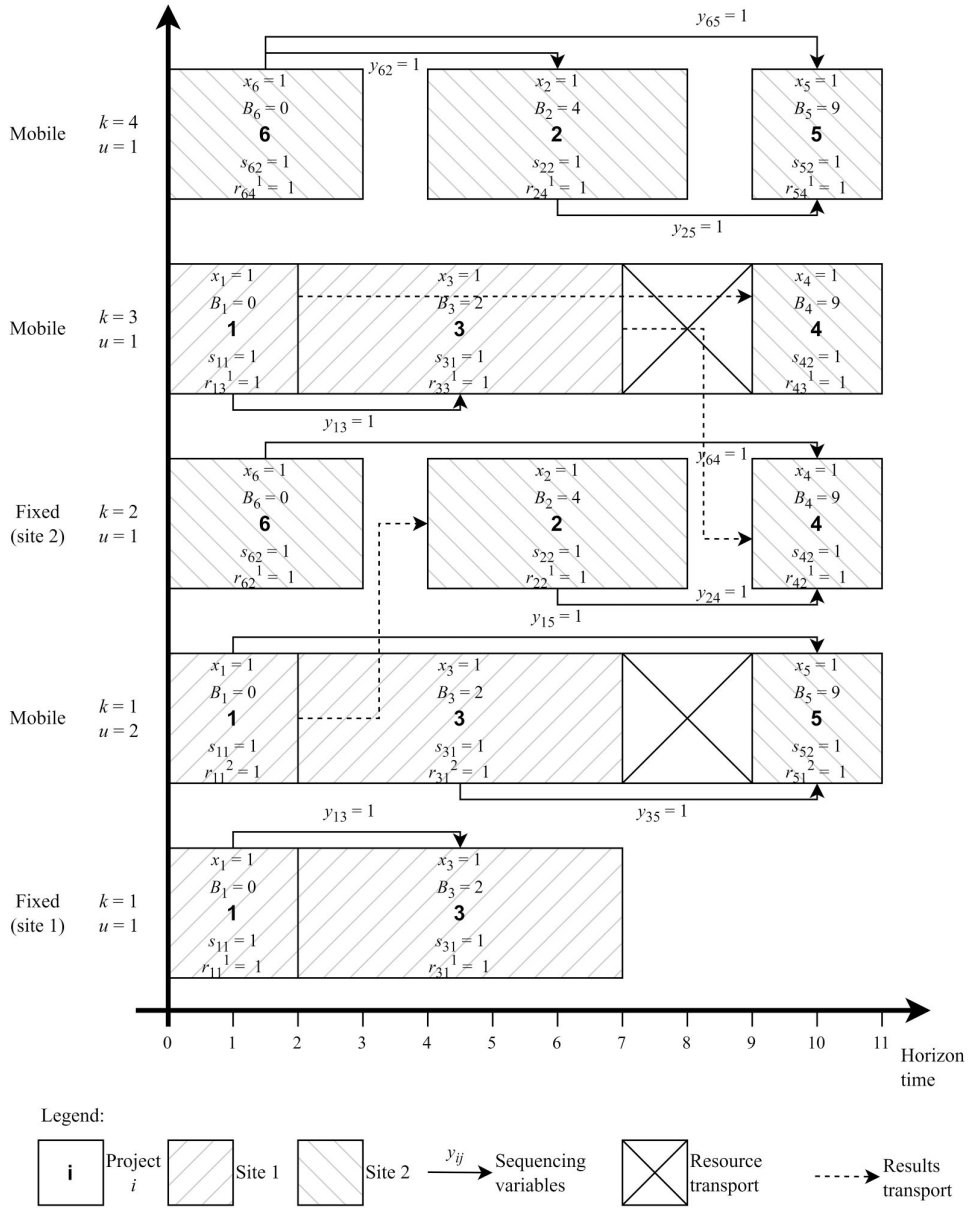

Figure 2. Optimal selection and scheduling of RPs in the illustrative example. Note that RP 7 was not selected.

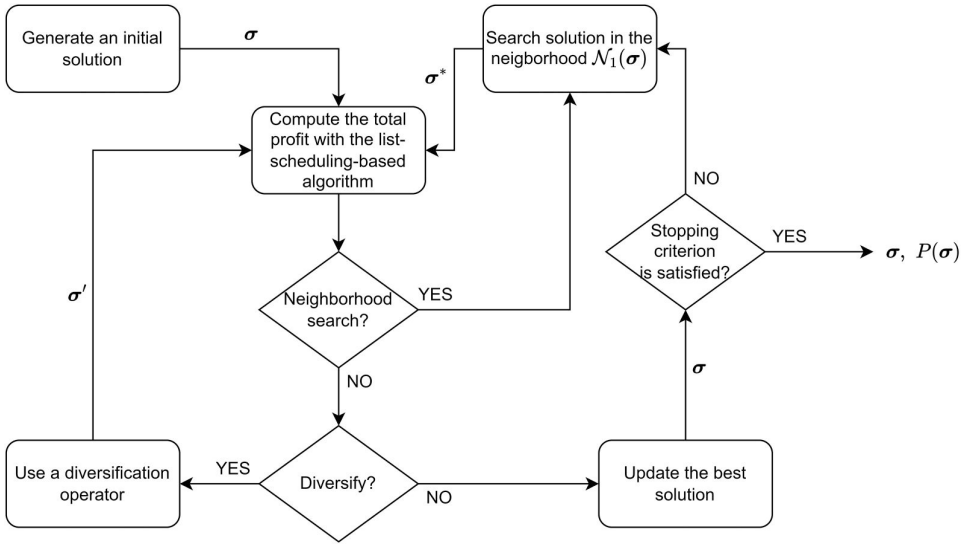

Figure 3. General scheme of the proposed method. The list-scheduling-based algorithm is the key procedure for finding near-optimal solutions.

Given a list of RPs $\boldsymbol{\sigma}$, the list-scheduled-based algorithm selects and schedules RPs, computing the earliest finish date f_i for each RP i, as follows.

Initially, for each RP i, we check to see if these predecessors are scheduled. This RP is only selected if all these predecessors are scheduled. Otherwise, we compute the availability date a^l_{ku}, for each unit u of type k required resource in each station l, according to the following cases:

- If $u \in F_k$, but $loc_{ku} \neq l$, then $a^l_{ku} = \infty$ (because the availability date is over the time horizon, we guarantee that the unit u of the k-type resource cannot be assigned to a selected RP j scheduled in the station l). If $loc_{ku} = l$, then $a^l_{ku} = 0$ if the unit u of type k resource is not yet assigned for any RP or $a^l_{ku} = f_j$ if the RP j is the last RP assigned to the unit u.
- If $u \notin F_k$, then $a^l_{ku} = 0$ if the unit u of type k resource is not yet assigned for any RP or $a^l_{ku} = f_j + \delta_{l_j l}$ if RP j is the last RP assigned to unit u, located at the station l_j.

At each station l, the algorithm assigns the earliest available resource to the RP i. In case of a tie, the algorithm assigns the cheapest available resource, considering the cost $p_i c^v_{ku} + c^f_{ku}$. Then, the algorithm determines the completion time f^l_i as follows: First, we compute the completion time of all preceding RPs of i at the station l, including the transportation time given by Equation (14).

$$A^l_i = \max\{d_j + \delta_{l_j l} : (j,i) \in E\} \tag{14}$$

If i has no precedence, $A^l_i = 0$. Then, Equation (15) calculates the time when all resources for executing RP i at station l are available.

$$B^l_i = \max\{a^l_{ku} : \text{the unit } u \text{ of type } k \text{ resource is assigned to RP } i\} \tag{15}$$

In Equation (16), we calculate the completion time for each station.

$$f^l_i = \max\{A^l_i, B^l_i\} + p_i \tag{16}$$

Finally, if $\min\{f^l_i : l \in L\} > T$, the RP i cannot be scheduled, i.e., the RP is not selected. Otherwise, we select the RP i and schedule it on the station l_i, with Equation (17).

$$l_i = \text{argmin}\{f^l_i : l \in L\} \tag{17}$$

Thus, we compute RP i on-station l_i completion time using Equation (18), assign the available resources to this station, and obtain the income of selecting this RP, the cost of using resources, and the resulting profit.

$$f_i = \min\left\{f^l_i : l \in L\right\} = f^{l_i}_i \tag{18}$$

Algorithm 1 shows the list-scheduling-based algorithm.

Algorithm 1. The list-scheduling-based algorithm

Input: $\boldsymbol{\sigma}$, parameters of the RPSAP
1: **for** $i \in \boldsymbol{\sigma}$ **such that** all predecessor RPs of i are scheduled **or** $(0,i) \in E$ **do**:
2: **for** $l \in L$ **do**:
3: Compute the availability date of each unit of resource a^l_{ku} at station l
4: Assign temporarily the selected resources to RP i at station l
5: Compute the completion time f^l_i at station l, with Equation (14)-(16)
6: **if** $min\left\{f^l_i : l \in L\right\} \leq T$ **then**:
7: Include i in the set of selected RPs.
8: Schedule the RP i at the station l_i and finish this in time f_i, according to Equation (17)-(18)
9: Compute the income, the cost, and the profit.
Output: f_i, set of selected RPs with resource assignation, total profit $P(\boldsymbol{\sigma})$

4.2. Project group swap neighborhood system

To explore the solution space, we consider a project group swap neighborhood system $\mathcal{N}_n(\boldsymbol{\sigma})$, where a neighbor of the solution $\boldsymbol{\sigma}$ is generated by the interchange of two blocks (or groups) of n consecutive RPs. Figure 4 shows an example of the project group swap neighborhood system. Note that, if $n = 1$, then the neighborhood system $\mathcal{N}_1(\boldsymbol{\sigma})$ considers all solutions generated by the interchange of two RPs in the list $\boldsymbol{\sigma}$.

4.3. Iterated local search

The first metaheuristic we present is the Iterated Local Search (ILS), proposed by Lourenço et al. (2003). ILS finds near-optimal solutions by searching in the solution space using a Local Search method for intensification and a Perturbation operator for diversification. Algorithm 2 shows the main procedure for ILS.

Algorithm 2. Iterated Local Search

Input: parameters of the RPSAP, parameters of ILS
1: Generate initial solution $\boldsymbol{\sigma}_0$
2: $\boldsymbol{\sigma}^* \leftarrow$ Local Search ($\boldsymbol{\sigma}_0, \mathcal{N}_1$, list-scheduling-based algorithm)
3: **while** the stopping criterion is not satisfied, **do**:
4: $\boldsymbol{\sigma}' \leftarrow$ Perturbation ($\boldsymbol{\sigma}^*$)
5: $\boldsymbol{\sigma}' \leftarrow$ Local Search ($\boldsymbol{\sigma}', \mathcal{N}_1$, list-scheduling-based algorithm)
6: $\boldsymbol{\sigma}^* \leftarrow$ Acceptance Criterion ($\boldsymbol{\sigma}'$, $\boldsymbol{\sigma}^*$, list-scheduling-based algorithm)
Output: $\boldsymbol{\sigma}^*$

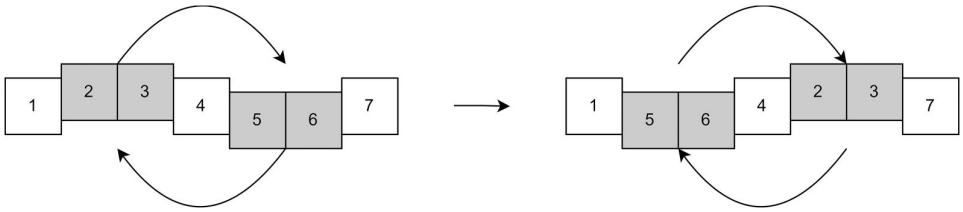

Figure 4. An example of the project group swap neighborhood system $\mathcal{N}_2(\sigma)$, where the solution $(1,5,6,4,2,3,7)$ is a neighbor of the solution $\sigma = (1,2,3,4,5,6,7)$.

The operators in the ILS algorithm are defined as follows. We generate a random list $\boldsymbol{\sigma}_0$ as an initial solution. The Local Search procedure considers a project group swap neighborhood system $\mathcal{N}_1$, given by a random interchange between two RPs. If no better solution is found after a specified number of iterations in the Local Search procedure, the final solution is considered the best local solution. For diversification, we define a Perturbation operator as a random interchange between a specific number of RPs in the list. At the end of each iteration, we accept a new solution if it has a better profit than the current solution. In the Local Search procedure and the Acceptance Criterion, we use the list-scheduled-based algorithm to evaluate the solution in the objective function. After the maximum iteration number Max_{it} or the execution time limit is reached, the ILS algorithm is stopped.

4.4. Variable neighborhood search

In this subsection, we present the Variable Neighborhood Search (VNS) proposed by Mladenović and Hansen (1997). Similarly to ILS, VNS employs a Local Search method for intensification but utilizes different project group swap neighborhood systems to explore various zones of the solution space. Algorithm 3 shows the main procedure for VNS.

Algorithm 3. Variable Neighborhood Search

Input: parameters of the RPSAP, parameters of VNS
Generate initial solution $\boldsymbol{\sigma}^*$
while the stopping criterion is not satisfied, **do**:
  $n \leftarrow 1$
  while $n \leq n_{max} - 1$ **do**:
    $\boldsymbol{\sigma}' \leftarrow$ Shake $(\boldsymbol{\sigma}^*, \mathcal{N}_{n+1})$
    $\boldsymbol{\sigma}' \leftarrow$ Local Search $(\boldsymbol{\sigma}', \mathcal{N}_1$, list-scheduling-based algorithm)
    $\boldsymbol{\sigma}^*, n \leftarrow$ Neighborhood Change $(\boldsymbol{\sigma}', \boldsymbol{\sigma}^*, n,$ list-scheduling-based algorithm)
Output: $\boldsymbol{\sigma}^*$

The operators are defined in the VNS as follows. As an initial solution, a random list $\boldsymbol{\sigma}^*$ is generated. The Shake operator randomly selects a neighbor $\boldsymbol{\sigma}\prime$ of $\boldsymbol{\sigma}^*$ in the project group swap neighborhood system $\mathcal{N}_{n+1}$, i.e., the list $\boldsymbol{\sigma}'$ is generated by the random interchange of two blocks of $n+1$ consecutive RPs of $\boldsymbol{\sigma}^*$. Note that the Shake operator considers the project group swap neighborhood systems from $\mathcal{N}_2$ to $\mathcal{N}_{n_{max}}$. The Local Search procedure utilizes a neighborhood system $\mathcal{N}_1$, defined by random swaps between two RPs within a project group. If no improvement occurs after a predefined number of iterations, the most recently identified solution is accepted as the best local solution. The Neighborhood Change operator accepts the new solution $\boldsymbol{\sigma}'$ if it has a better profit than the solution $\boldsymbol{\sigma}^*$ and returns to the initial neighborhood system $\mathcal{N}_2$. Otherwise, the operator goes to the next neighborhood system $\mathcal{N}_{n+1}$. VNS stops after the maximum iteration number Max_{it} or the execution time limit is reached.

4.5. Simulated annealing

Finally, we present the Simulated Annealing (SA) metaheuristic proposed by Kirkpatrick et al. (1983). The cooling process of certain materials inspires SA until they reach an equilibrium state. SA explores the solution space using a project swap neighborhood system $\mathcal{N}_1$ for intensification, but accepts worst solutions with a decreasing probability given by the actual temperature to escape from a local optimum. Algorithm 4 shows the main procedure for SA.

Algorithm 4. Simulated Annealing

Input: parameters of the RPSAP, parameters of SA
Generate initial solution $\boldsymbol{\sigma}'$
$\boldsymbol{\sigma}^* \leftarrow \boldsymbol{\sigma}'$
Generate initial temperature $Temp$ such that the probability of acceptance is near to $Prob_0$
while the stopping criterion is not satisfied, **do**:
  $\boldsymbol{\sigma}'' \leftarrow \boldsymbol{\sigma} \in \mathcal{N}_1(\boldsymbol{\sigma}')$ /* *selected randomly* */
  $\Delta \leftarrow P(\boldsymbol{\sigma}'') - P(\boldsymbol{\sigma}')$
  if $\Delta > 0$ **then:**
    $\boldsymbol{\sigma}' \leftarrow \boldsymbol{\sigma}''$
  else if $random(0,1) < \exp(\Delta/Temp)$ **then:**
    $\boldsymbol{\sigma}' \leftarrow \boldsymbol{\sigma}''$
  if $P(\boldsymbol{\sigma}') > P(\boldsymbol{\sigma}^*)$ **then:**
    $\boldsymbol{\sigma}^* \leftarrow \boldsymbol{\sigma}'$
  $Temp, \alpha \leftarrow$ Temperature Update $(Temp, \boldsymbol{\sigma}^*, \boldsymbol{\sigma}', \alpha)$
Output: $\boldsymbol{\sigma}^*$

We define the SA operators as follows. A random list $\boldsymbol{\sigma}'$ is generated as an initial solution and assigned as the best solution. The initial temperature is generated randomly by selecting a certain number of neighbors in $\mathcal{N}_1(\boldsymbol{\sigma}')$ and calculating the average of the absolute value of differences between the profits of solutions $\overline{\Delta} > 0$. Since $Prob_0 = \exp\left(-\overline{\Delta}/Temp\right)$, then the initial temperature is calculated by $Temp = -\overline{\Delta}/\ln\left(Prob_0\right)$. In each iteration of the algorithm a random neighbor $\boldsymbol{\sigma}'' \in \mathcal{N}_1(\boldsymbol{\sigma}')$ of the current solution $\boldsymbol{\sigma}'$ is selected. If the profit of the new solution is better than the current solution, then this solution is accepted as the new current solution. Otherwise, the new solution is accepted as the new solution with probability $\exp\left(\Delta/Temp\right)$. Next, if the profit of the current solution is better than the best solution, then the current solution is the new best solution. Finally, the Temperature Update operator updates the temperature $Temp$ and the cooling ratio α as follows: the temperature is updated by a geometric schedule $Temp \leftarrow \alpha \cdot Temp$, where the cooling ratio is $\alpha = \alpha_{fast}$, if the best solution is not updated after a certain number of iterations, or $\alpha = \alpha_{slow}$, otherwise. The algorithm stops after the minimal temperature $Temp_{min}$ or the execution time limit is reached.

4.6. Enhancements and innovations

We explicitly designed the proposed metaheuristics to address the unique challenges of the RPSAP by incorporating several innovative elements that enhance their performance. ILS employs a project group swap neighborhood system, enabling the practical exploration and exploitation of the solution space. Its perturbation mechanism has been fine-tuned to balance diversification and intensification. VNS leverages multiple neighborhood systems, allowing it to systematically explore different regions of the solution space and improve convergence. SA introduces an adaptive cooling schedule and refined acceptance criteria, enabling it to escape local optima efficiently while maintaining computational speed. Furthermore, all metaheuristics integrate a list-scheduling-based algorithm to construct feasible solutions that adhere to the problem's precedence, resource, and transportation constraints. As demonstrated in extensive computational experiments, these enhancements were developed to overcome traditional methods' limitations and ensure the proposed approaches' scalability and robustness.

5. Computational experiments

This section presents the results obtained from testing the mathematical model and metaheuristics. We generate test instances, and the generation strategy is provided in Section 5.1. Then, we present the results obtained from the model and metaheuristics, comparing and discussing their performance using test instances of varying sizes. These comparisons are presented in Sections 5.2 and 5.3, respectively. The mathematical model is solved with CPLEX 22.1.1, and each metaheuristic is implemented in Python 3.9.13. The execution is conducted on a 64-bit Windows 10 Pro operating system, an Intel Core i7-4790S processor (3.2 GHz), and 8 GB RAM. Because the parameters are integers, the optimal solution for the continuous-time model is integer. Thus, we test the performance of the continuous-time model (CTM) and the integer-time model (ITM), which is obtained by assuming integrality in the decision variable B_i (Bigler et al., 2024). For each instance, we run CPLEX until the relative gap between the upper bound (UB) and lower bound (LB) is under 10^{-4} or a time limit of 300 s is reached. Moreover, we run each metaheuristic 20 times by instance (Laurent et al., 2017). Each run of the ILS and VNS considers stopping criteria of 500 iterations or 300 s. The Local Search method considers stopping criteria of 50 iterations without a solution upgrade. For diversification, the perturbation operator in ILS randomly changes four RPs, and the VNS explores $n_{max} - 1 = 2$ neighborhood systems. The SA method runs until the temperature reaches 10^{-5}, or the execution time reaches 300 s. The initial temperature is calculated using three neighbors of the initial solution, so the initial probability of accepting the worst solution is approximately 95%. We use a fast-cooling ratio $\alpha_{fast} = .95$, where the best solution is not improved after 10 iterations and a slow-cooling ratio $\alpha_{slow} = .99$ otherwise.

5.1. Test instances

According to the literature, the instance generation strategy consists of a random and bounded variation of the parameters (Kolisch & Sprecher, 1997; Kumar et al., 2018; Laurent et al., 2017). Thus, we generate 480 test instances inspired by the real world to test the proposed mathematical model and metaheuristic performance. These instances belong to 60 different classes, with 8 test instances in each class. Thereby, the instances are generated as follows. Let $U_{\mathbb{Z}}(a, b)$ represents the uniform distribution of integer numbers on the interval (a, b). We generate instances with the following control parameters: $|V|$ RPs, $|L|$ stations, $|R|$ types of resources, and a time horizon of T periods. Additionally, we identify the class of instances with $|V|$ RPs, $|L|$ stations, $|R|$ types of resources, and T periods, with V$|V|$-L$|L|$-R$|R|$-TT. For example, a class with 15 RPs, two stations, 20

types of resources, and a time horizon of 30 periods is denoted as V15-L2-R20-T30. Each class is denoted as $\mathcal{C}_m$, with $m = 1, ..., 60$, according to Table 2. Thus, the instances are generated with $|V| \in \{10, 15, 20, 25, 30\}$ RPs, $|L| \in \{2, 3\}$ stations, $|R| \in \{10, 20\}$ types of resources (Laurent et al., 2017). Because the summer-research-season is about 120 days, we consider a time horizon of $T \in \{30, 60, 120\}$ periods (we consider 30 and 60 periods to stress the solution methods).

We randomly generate the other parameters as follows. For each station $l, l' \in L$, the transportation time $\delta_{ll'} \sim U_{\mathbb{Z}}(5, 8)$ (Laurent et al., 2017). For each resource of $k \in R$ type, the set of units available per type $|R_k| \sim U_{\mathbb{Z}}(20, 30)$ (Kumar et al., 2018); a unit r of type k resource $r \in F_k$, and $loc_{kr} \sim U_{\mathbb{Z}}(1, |L|)$, with a probability of 0.003. For each RP $i \in V$, the duration $p_i \sim U_{\mathbb{Z}}(7, 10)$ (Kumar et al., 2018) and type k resources needed for execution $r_{jk} \sim U_{\mathbb{Z}}\left(0, \frac{\gamma_k}{2}\right)$, where $\gamma_k = \min\{|\{r \in F_k : loc_{kr} = l\} \cup F_k^C| : l \in L\}$ represents the minimum availability of the type k resource at all stations, including fixed and non-fixed resources. For each RP i, the income $I_i \sim U_{\mathbb{Z}}(100\ 000, 200\ 000)$; for each unit u for type k resource, the variable cost $c^v_{ku} \sim U_{\mathbb{Z}}(5, 30)$, and the fixed cost $c^f_{ku} \sim U_{\mathbb{Z}}(50, 200)$, or $c^f_{ku} = 0$, with probability 0.5.

The precedence relations between RPs are randomly generated as follows. We randomly divide the set of RPs into N subsets such that $V = V_1 \cup \cdots \cup V_N$, and $V_m \cap V_n = \emptyset$, if $m \neq n$. The number of subsets N depends on the number of RPs $|V|$: if $|V| \in \{10, 15\}$, then $N \sim U_{\mathbb{Z}}(2, 4)$; if $|V| = 20$, then $N \sim U_{\mathbb{Z}}(2, 6)$; else if $|V| \in \{25, 20\}$, then $N \sim U_{\mathbb{Z}}(3, 6)$. The size of each subset is randomly selected. For each $i \in V_m$, we randomly choose $j \in V_{m+1}$ as a successor, with probability $prob$. Initially, $prob = 0.5$, updated by multiplying 0.5 each time a successor is assigned (Kolisch & Sprecher, 1997).

The test instances were meticulously crafted to capture the complexities and constraints inherent to the RPSAP, guaranteeing a realistic evaluation of the proposed methods. These instances encompass a range of configurations involving research projects, stations, and types of resources, featuring different levels of complexity in precedence relationships, transportation delays, and resource availability. Particular emphasis was placed on replicating real-world conditions by incorporating fixed and transportable resources, varying project durations, and environmental constraints. The instances vary in size, ranging from small (10 projects) to large (30 projects), and include a mix of station and resource configurations and varying horizon times to stress-test the scalability and robustness of the proposed algorithms. This thorough design ensures that the test instances yield valuable insights into the performance of the metaheuristics in tackling the challenges presented by large-scale, resource-constrained selection and scheduling problems.

The size of each instance is given by the number of decision variables and the number of constraints. Although these numbers depend on the control parameters, they also depend on other randomly generated parameters, including the number of units of each type of resource, the number of units of fixed resources, the types of resources required by RPs, and the precedence relations among them. Therefore, in the same class, we have instances of different sizes. Figure 5 illustrates the size of each instance, measured by the number of decision variables and constraints, for the test instances generated by CPLEX. In Figure 5, we note that the number of decision variables and constraints increases when the control parameters increase their values. Moreover, for similar classes, the number of decision variables and constraints increases significantly when the number of resources increases from 10 to 20 types.

Figure 6 shows the set of 480 instances. Each point represents an instance, and its location in the graph depends on the number of constraints and variables. Note that the instances are grouped into ten clusters. We obtain these clusters using the K-means algorithm. Table 3 describes each cluster,

Table 2. List of notations for each type of test instance.

$\mathcal{C}_m$	Class	$\mathcal{C}_m$	Class	$\mathcal{C}_m$	Class	$\mathcal{C}_m$	Class
$\mathcal{C}_1$	V10-L2-R10-T030	$\mathcal{C}_{16}$	V15-L2-R20-T030	$\mathcal{C}_{31}$	V20-L3-R10-T030	$\mathcal{C}_{46}$	V25-L3-R20-T030
$\mathcal{C}_2$	V10-L2-R10-T060	$\mathcal{C}_{17}$	V15-L2-R20-T060	$\mathcal{C}_{32}$	V20-L3-R10-T060	$\mathcal{C}_{47}$	V25-L3-R20-T060
$\mathcal{C}_3$	V10-L2-R10-T120	$\mathcal{C}_{18}$	V15-L2-R20-T120	$\mathcal{C}_{33}$	V20-L3-R10-T120	$\mathcal{C}_{48}$	V25-L3-R20-T120
$\mathcal{C}_4$	V10-L2-R20-T030	$\mathcal{C}_{19}$	V15-L3-R10-T030	$\mathcal{C}_{34}$	V20-L3-R20-T030	$\mathcal{C}_{49}$	V30-L2-R10-T030
$\mathcal{C}_5$	V10-L2-R20-T060	$\mathcal{C}_{20}$	V15-L3-R10-T060	$\mathcal{C}_{35}$	V20-L3-R20-T060	$\mathcal{C}_{50}$	V30-L2-R10-T060
$\mathcal{C}_6$	V10-L2-R20-T120	$\mathcal{C}_{21}$	V15-L3-R10-T120	$\mathcal{C}_{36}$	V20-L3-R20-T120	$\mathcal{C}_{51}$	V30-L2-R10-T120
$\mathcal{C}_7$	V10-L3-R10-T030	$\mathcal{C}_{22}$	V15-L3-R20-T030	$\mathcal{C}_{37}$	V25-L2-R10-T030	$\mathcal{C}_{52}$	V30-L2-R20-T030
$\mathcal{C}_8$	V10-L3-R10-T060	$\mathcal{C}_{23}$	V15-L3-R20-T060	$\mathcal{C}_{38}$	V25-L2-R10-T060	$\mathcal{C}_{53}$	V30-L2-R20-T060
$\mathcal{C}_9$	V10-L3-R10-T120	$\mathcal{C}_{24}$	V15-L3-R20-T120	$\mathcal{C}_{39}$	V25-L2-R10-T120	$\mathcal{C}_{54}$	V30-L2-R20-T120
$\mathcal{C}_{10}$	V10-L3-R20-T030	$\mathcal{C}_{25}$	V20-L2-R10-T030	$\mathcal{C}_{40}$	V25-L2-R20-T030	$\mathcal{C}_{55}$	V30-L3-R10-T030
$\mathcal{C}_{11}$	V10-L3-R20-T060	$\mathcal{C}_{26}$	V20-L2-R10-T060	$\mathcal{C}_{41}$	V25-L2-R20-T060	$\mathcal{C}_{56}$	V30-L3-R10-T060
$\mathcal{C}_{12}$	V10-L3-R20-T120	$\mathcal{C}_{27}$	V20-L2-R10-T120	$\mathcal{C}_{42}$	V25-L2-R20-T120	$\mathcal{C}_{57}$	V30-L3-R10-T120
$\mathcal{C}_{13}$	V15-L2-R10-T030	$\mathcal{C}_{28}$	V20-L2-R20-T030	$\mathcal{C}_{43}$	V25-L3-R10-T030	$\mathcal{C}_{58}$	V30-L3-R20-T030
$\mathcal{C}_{14}$	V15-L2-R10-T060	$\mathcal{C}_{29}$	V20-L2-R20-T060	$\mathcal{C}_{44}$	V25-L3-R10-T060	$\mathcal{C}_{59}$	V30-L3-R20-T060
$\mathcal{C}_{15}$	V15-L2-R10-T120	$\mathcal{C}_{30}$	V20-L2-R20-T120	$\mathcal{C}_{45}$	V25-L3-R10-T120	$\mathcal{C}_{60}$	V30-L3-R20-T120

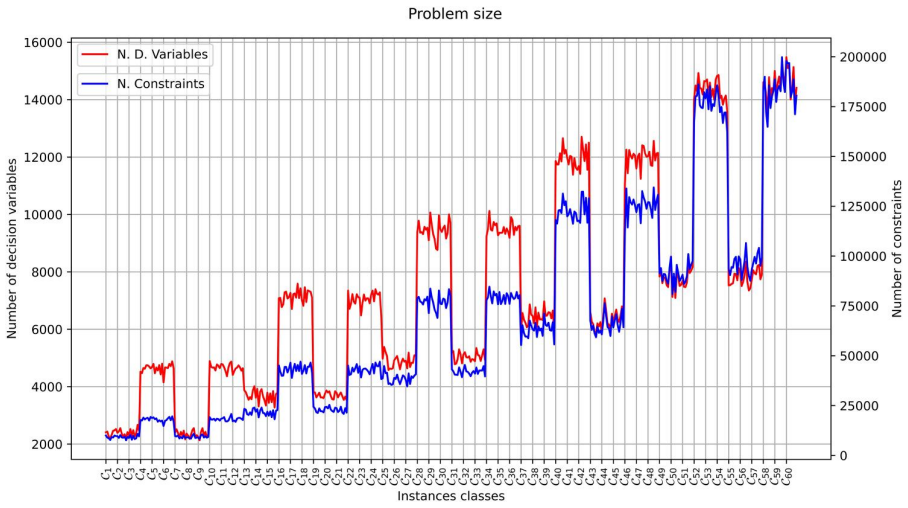

Figure 5. Number of decision variables and constraints for each instance. In the x-axis, all instances of the class $\mathcal{C}_m$ are between classes $\mathcal{C}_m$ and $\mathcal{C}_{m+1}$.

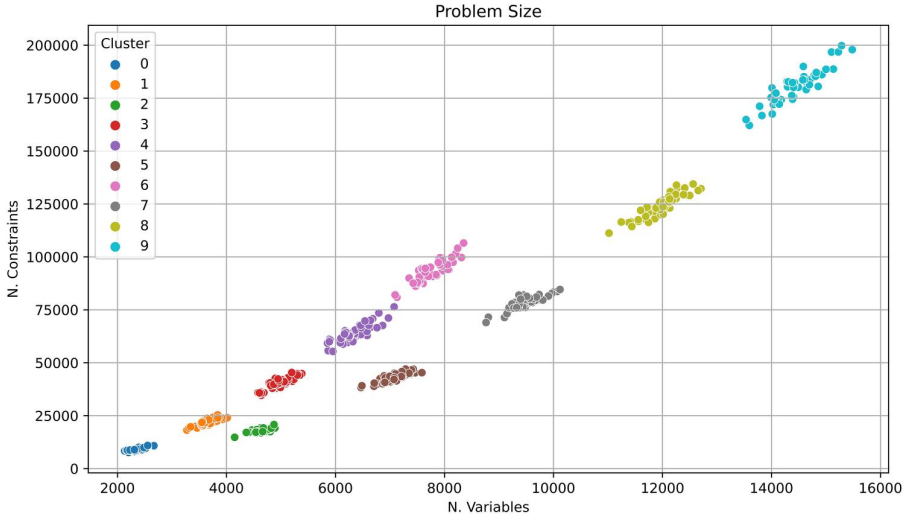

Figure 6. Problem size according to the number of constraints and decision variables. Each marker represents a test instance.

Table 3. Description of clusters.

		Number of decision variables			Number of constraints		
Cluster	Classes	Min	Average	Max	Min	Average	Max
0	V10-L2-R10-T30, V10-L2-R10-T60, V10-L2-R10-T120, V10-L3-R10-T30, V10-L3-R10-T60, V10-L3-R10-T120	2132	2368.9	2668	7599	9220.1	10897
1	V15-L2-R10-T30, V15-L2-R10-T60, V15-L2-R10-T120, V15-L3-R10-T30, V15-L3-R10-T60, V15-L3-R10-T120	3274	3687.3	4014	18155	22238.5	25334
2	V10-L2-R20-T30, V10-L2-R20-T60, V10-L2-R20-T120, V10-L3-R20-T30, V10-L3-R20-T60, V10-L3-R20-T120	4150	4653.8	4889	14761	18098.6	20782
3	V20-L2-R10-T30, V20-L2-R10-T60, V20-L2-R10-T120, V20-L3-R10-T30, V20-L3-R10-T60, V20-L3-R10-T120	4580	4969.3	5382	34603	40297.2	45356
4	V25-L2-R10-T30, V25-L2-R10-T60, V25-L2-R10-T120, V25-L3-R10-T30, V25-L3-R10-T60, V25-L3-R10-T120	5854	6365.1	7076	55333	64239.2	76383
5	V15-L2-R20-T30, V15-L2-R20-T60, V15-L2-R20-T120, V15-L3-R20-T30, V15-L3-R20-T60, V15-L3-R20-T120	6469	7100.8	7586	38292	43209.1	47001
6	V30-L2-R10-T30, V30-L2-R10-T60, V30-L2-R10-T120, V30-L3-R10-T30, V30-L3-R10-T60, V30-L3-R10-T120	7094	7782.0	8351	80892	93393.4	106538
7	V20-L2-R20-T30, V20-L2-R20-T60, V20-L2-R20-T120, V20-L3-R20-T30, V20-L3-R20-T60, V20-L3-R20-T120	8763	9484.4	10120	68979	78340.0	84603
8	V25-L2-R20-T30, V25-L2-R20-T60, V25-L2-R20-T120, V25-L3-R20-T30, V25-L3-R20-T60, V25-L3-R20-T120	11019	11967.9	12707	111223	124053.3	134401
9	V30-L2-R20-T30, V30-L2-R20-T60, V30-L2-R20-T120, V30-L3-R20-T30, V30-L3-R20-T60, V30-L3-R20-T120	13534	14443.6	15482	162106	180156.4	199781

indicating the classes of instances and the minimum, average, and maximum numbers of variables and constraints. We note that each instance is defined by changing the number of RPs or the number of resource types. Furthermore, we note that the clusters with ten resource types (clusters 0, 1, 3, 4, and 6) and the clusters with 20 resource types (clusters 2, 5, 7, 8, and 9) follow two distinct patterns, respectively. Therefore, the growth in resource types is more significant than the number of RPs for the problem size.

5.2. Results for the mathematical model

Table 4 summarizes the results of the experiments executed for the CTM and ITM mathematical models. Additionally, Figure 7 compares the relative gap in each instance between models. In general, the ITM performs better than the CTM. In fact, in 58.125% of the test instances, the ITM has a better relative gap than the CTM. Additionally, the average relative gap for the CTM is 1.310, and the relative gap for the ITM is 1.277. The ITM is better than the CTM in 6.223 in terms of the maximum relative gap. Both models achieve the optimal solution in only 6.875% of test instances. The execution time for the ITM is better than that of the CTM. Indeed, the average execution time for the ITM is 279.90 s, and for the CTM, it is 279.92 s. The standard deviation and the maximum execution time are better in the ITM. Only the minimum execution time is better in the CTM.

Table 5 summarizes the results of each model, categorizing the test instances by cluster. The first column indicates the cluster of instances. Model groups the following six columns. Each column displays the average relative gap, the percentage of optimal solutions obtained by the model, and the average execution time of the instance group. The last column shows the percentage of instances in the group where the ITM has a better or equal relative gap than the CTM.

The ITM has a better average relative gap than the CTM in 8 clusters. Indeed, Table 5 shows that only in clusters 2 and 8 does the CTM have a better average relative gap than the ITM. Nevertheless, the percentage of instances where the ITM has an equal or better relative gap than the CTM is over 50% in all clusters. The most significant difference in the average relative gap between the ITM and the CTM is found in cluster 9. The average relative gap differed by 0.209.

Furthermore, only 3 clusters are instances where an optimal solution is reached. In clusters 0 and 2, 33.333% of executions reach an optimal solution before 300 s (the clusters with the smallest-sized

Table 4. Summary of the execution of each model.

	Relative gap					Execution time (s)			
Model	Min.	Ave.	SD	Max.	% OS	Min.	Ave.	SD	Max.
CTM	0	1.310	1.777	15.198	6.875	**0.13**	279.92	76.05	306.42
ITM	0	**1.277**	**1.734**	**8.975**	6.875	0.14	**279.90**	**76.04**	**306.34**
Percentage of instances where ITM has a better Relative Gap than CTM									58.125

Abbreviations are defined as follows: CTM: Continuous-time model; ITM: Integer-time model; Min: Minimum; Ave.: Average; SD: Standard Deviation; Max.: Maximum; % OS: Percentage of instances where the optimal solution is reached. In each column, the best value for the relative gap and execution time between CTM and ITM is shown in bold.

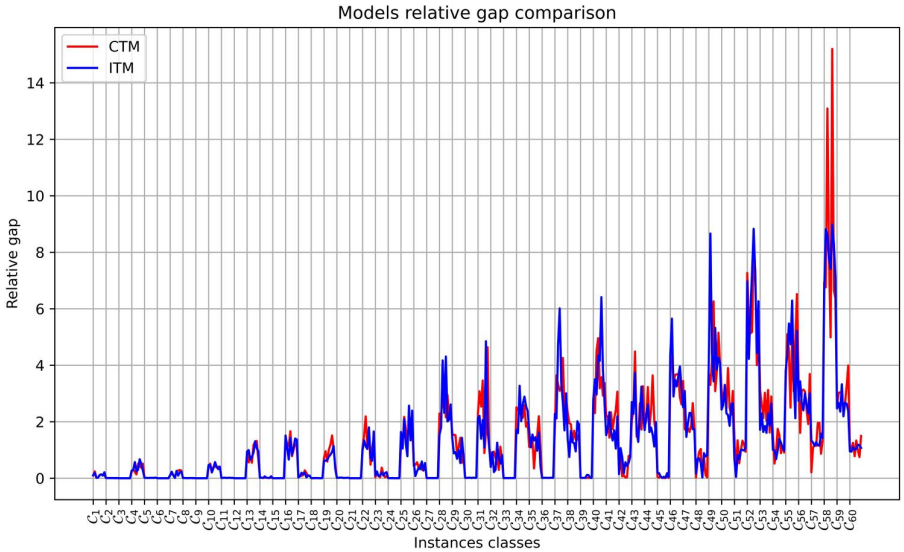

Figure 7. Relative Gap of the models, for each instance. In the x-axis, all instances of the class $\mathcal{C}_m$ are between classes $\mathcal{C}_m$ and $\mathcal{C}_{m+1}$.

Table 5. Summary of the execution of each model, classified by cluster.

	Continuous time model (CTM)			Integer time model (ITM)			
Cluster	ARG	%OS	AET (s)	ARG	%OS	AET (s)	ITMvCTM (%)
0	0.051	33.333	200.35	**0.046**	33.333	200.38	66.667
1	0.312	0	300.80	**0.289**	0	300.69	56.250
2	**0.128**	33.333	200.60	0.136	33.333	200.60	68.750
3	0.884	0	300.60	**0.790**	0	300.95	56.250
4	1.591	0	300.34	**1.503**	0	300.31	50.000
5	0.461	2.083	294.51	**0.432**	2.083	294.10	62.500
6	2.743	0	300.38	**2.760**	0	300.44	52.083
7	1.210	0	300.79	**1.207**	0	300.69	54.167
8	**2.007**	0	300.36	2.097	0	300.39	52.083
9	3.713	0	300.44	**3.504**	0	300.45	62.500

Abbreviations are defined as ARG: Average relative gap; % OS: Percentage of instances where the optimal solution is reached; AET: Average execution time; ITMvCTM: Percentage of instances where ITM has a better Relative Gap than CTM. In each row, the lowest average relative gap between CTM and ITM is shown in bold.

instances, having 10 and 20 resource types, respectively); in cluster 5, only 2.083% (the cluster with the second smallest-sized instances, having 20 resource types). For both models, the average execution time is 300 s for the clusters with no optimal solutions, 295 s for cluster 5, and 200 s for clusters 0 and 2.

Similar to Tables 5, 6 summarizes the results of each model, categorizing the test instances by control parameters. The first five rows classify the instances by the number of RPs. The next ones classify the instances by the number of stations and resource types. Finally, the last three rows classify the instances according to the time horizon.

In the 10 and 25 RP instances, the CTM has a better average relative gap than the ITM. The average relative gap for the instances with 10 RPs is 0.089 for CTM and 0.091 for ITM. The average relative gap for the instances with 25 RPs is 1.799 for CTM and 1.800 for ITM. Nevertheless, these differences do not imply a better performance of the CTM in these sets of instances. In all sets, the percentage of instances where the ITM has a better relative gap than the CTM is over 51%. Besides, both models have instances with an optimal solution only in sets with 10 or 15 RPs. An optimal solution is achieved in only 33.333% of instances with 10 RPs and 1.042% with 15 RPs. Depending on the percentage of instances with an optimal solution, the average execution times vary in each set. In both models, the average execution time for the 10-RP instances is approximately 200 s, for the 15-RP instances, approximately 297 s, and for the other sets, approximately 300 s.

The results show that the ITM performs better when the number of stations is used to categorize the instances. For both sets, the percentage of instances where the ITM has a relative gap greater than or equal to the CTM is over 57%. Additionally, an average relative gap of 1.220 was obtained with the ITM and 1.235 with the CTM for cases involving two stations. In the 3-station instances, the average relative gap was 1.333 for the ITM and 1.385 for the CTM. In neither of the two models was an optimal solution found in more than 7.1% of the instances. Specifically, the optimal solution was found in only 6.667% of instances with two stations and 7.083% with three stations. The average execution time is nearly 280 s for 2-station instances and 279 s for 3-station instances.

Table 6. Summary of the execution of each model, classified by control parameters.

	Continuous time model			Integer time model			
CP	ARG	%OS	AET (s)	ARG	%OS	AET (s)	ITMvCTM (%)
V10	**0.089**	33.333	200.47	0.091	33.333	200.49	67.708
V15	0.387	1.042	297.66	**0.361**	1.042	297.39	59.375
V20	1.047	0	300.69	**0.998**	0	300.82	55.208
V25	**1.799**	0	300.35	1.800	0	300.35	51.042
V30	3.228	0	300.41	**3.132**	0	300.44	57.292
L2	1.235	6.667	280.58	**1.220**	6.667	280.54	59.167
L3	1.385	7.083	279.25	**1.333**	7.083	279.25	57.083
R10	1.116	6.667	280.49	**1.078**	6.667	280.56	56.250
R20	1.504	7.083	279.34	**1.475**	7.083	279.24	60.000
T30	**2.491**	0	300.33	2.531	0	300.31	50.000
T60	1.172	0	300.70	**1.025**	0	300.73	60.000
T120	**0.266**	20.625	238.72	0.275	20.625	238.66	64.375

Abbreviations are defined as follows: CP: Control Parameter; ARG: Average Relative Gap; % OS: Percentage of instances where the optimal solution is achieved; AET: Average Execution Time; ITMvCTM: Percentage of instances where ITM has a better Relative Gap than CTM. In each row, the lowest average relative gap between CTM and ITM is shown in bold.

Similarly, when instances are classified by resource type, the ITM performs better than the CTM. The ITM has an average relative gap greater than or equal to the CTM over 56% of the instances. Additionally, in the set of instances with ten resource types, the ITM achieves an average relative gap of 1.078, while the CTM achieves an average relative gap of 1.116. Furthermore, the ITM and CTM obtain an average relative gap of 1.475 and 1.504, respectively, in the instances with 20 resource types. In both models, the percentage of optimal solutions found is less than 7.1%. Specifically, the commercial software found optimal solutions in only 6.667% of 10 resource-type instances and 7.083% of 20 resource-type instances. The average execution time is nearly 280 s for instances with ten resource types and 279 s for instances with 20 resource types.

Although the problem size does not depend on the time horizon, Table 6 shows that performance

improves as this control parameter increases. Both models exhibit a decrease in the average relative gap as the time horizon is extended. In particular, the average relative gaps for the sets of instances with time horizons of 30, 60, and 120 periods are 2.491, 1.172, and 0.266 for the CTM and 2.531, 1.025, and 0.275 for the ITM, respectively. Moreover, in both models, instances with optimal solutions (20.625%) are only found in the set with a time horizon of 120 periods. The average execution time is nearly 300 s for instances with a horizon time of 30 and 60 periods and 238 s for instances with 120 periods. The ITM has a lower average relative gap than the CTM only in the set of instances with a 60-period time horizon. However, the comparative performance by instances shows that ITM has a relative gap less than or equal to CTM in over 50% of the instances in each set.

5.3. Results for the metaheuristics

We used the same 480 test instances to compare the metaheuristic's performance with that of the mathematical models. We measured performance using the relative gap calculated according to Equation (19).

$$GAP_{ins}^{met} = \frac{UB_{ins} - FO_{ins}^{met}}{FO_{ins}^{met}}, \quad \forall met \in \{CTM, ITM\} \cup \{ILS_{rep}, VNS_{rep}, SA_{rep} : rep = 1, \ldots, 20\}, \quad ins = 1, \ldots, 480 \tag{19}$$

In Equation (19), GAP_{ins}^{met} is the relative gap of the instance *ins* solved by the method *met*. Methods refer to mathematical models (CTM or ITM) or a repetition *rep* of each metaheuristic (ILS_{rep}, VNS_{rep}, SA_{rep}). FO_{ins}^{met} is the objective function value, and $UB_{ins} = \min\{UB_{ins}^{CTM}, UB_{ins}^{ITM}\}$ is the best upper bound given by the commercial software.

Table 7 shows the overall results for each metaheuristic repetition. The first column indicates the repetition of the metaheuristic. The following three columns show the average relative gap, number of iterations, and execution time for the ILS metaheuristic. Next, six columns show the same information for the VNS and SA metaheuristics. The last two columns present the method that solves most instances with the smallest relative gap and the percentage of instances it solves. For comparison with the mathematical models, the repetition with the best relative gap is considered in each instance. The last two rows show the best and worst relative gap results. Figure 8 shows the percentage of instances in the best and worst repetition where each method has the lowest relative gap compared to the other methods. Table 8 presents the minimum, average, standard deviation, and maximum relative gaps, execution times, and numbers of iterations for each metaheuristic.

Metaheuristics perform better than mathematical models. In fact, according to Table 4, the average relative gaps for CTM and ITM are 1.310 and 1.277, respectively. However, from Table 7, the average relative gap of each metaheuristic in each repetition does not exceed 0.405. In the best repetition, Table 8 shows that the maximum relative gap for ILS, VNS, and SA is 1.977, 1.984, and 2.038, respectively. For the SA, the maximum relative gap is 7.465 times lower than CTM's maximum relative gap and 4.408 times lower than ITM's. Thus, the superior

Table 7. Summary of the execution of metaheuristics.

	ILS			VNS			SA				
Rep.	ARG	ANI	AET(s)	ARG	ANI	AET(s)	ARG	ANI	AET(s)	MSG	%MSG
1	0.351	78.383	303.917	**0.350**	33.981	308.395	0.404	614.669	32.904	ILS	35.208
2	**0.351**	83.079	303.685	0.353	34.790	309.142	0.401	617.690	33.168	ILS	33.750
3	**0.351**	82.935	303.891	0.354	34.554	309.058	0.407	614.975	32.927	ILS	36.042
4	**0.350**	81.433	303.654	0.351	34.398	308.181	0.403	613.890	32.992	ILS	34.583
5	**0.351**	81.081	304.140	**0.351**	34.594	307.878	0.403	616.944	32.976	ILS	34.583
6	0.351	80.890	304.055	**0.350**	34.429	308.291	0.405	614.421	32.889	ILS	34.167
7	**0.350**	82.760	303.897	0.354	34.669	308.288	0.401	615.573	32.958	ILS	36.042
8	0.354	81.879	304.062	**0.353**	34.552	308.498	0.402	612.735	32.710	ILS	36.042
9	**0.350**	83.769	303.833	**0.350**	43.817	306.349	0.406	615.383	32.711	ILS	33.125
10	0.351	83.392	303.756	**0.350**	43.842	306.487	0.405	616.594	33.016	ILS	32.292
11	**0.351**	82.988	303.734	0.360	9.825	598.698	0.397	620.158	33.076	ILS	31.458
12	**0.349**	83.660	304.051	0.352	35.458	308.413	0.404	615.375	32.849	ILS	34.792
13	**0.349**	82.913	303.791	0.352	35.396	308.395	0.402	616.285	32.960	ILS	36.042
14	**0.349**	81.183	304.226	0.354	34.983	308.043	0.404	619.404	33.062	ILS	33.958
15	**0.352**	80.894	303.717	0.354	35.304	308.549	0.401	613.465	32.594	ILS	36.042
16	**0.352**	80.750	303.649	0.353	35.517	307.970	0.402	617.450	32.667	ILS	34.792
17	**0.352**	83.315	304.078	0.353	35.248	308.281	0.404	615.344	32.787	ILS	33.333
18	**0.350**	80.292	303.790	0.351	43.604	306.429	0.405	615.317	32.717	ILS	33.958
19	0.352	79.671	303.996	**0.349**	43.440	306.676	0.403	616.663	32.853	VNS	30.833
20	0.352	79.775	303.633	**0.348**	43.129	306.772	0.405	614.444	33.122	VNS	31.250
BGR	**0.325**	79.902	303.696	0.326	33.531	442.321	0.347	633.738	34.262	ILS	37.917
WGR	**0.380**	81.410	303.958	0.382	32.556	313.845	0.471	592.331	31.021	ILS	36.042

Abbreviations are defined as follows: Rep: Repetition of execution of metaheuristics; ILS: Iterated local Search; VNS.: Variable neighborhood search; SA: Simulated annealing; ARG.: Average relative gap; ANI: Average number of iterations; AET(s): Average execution time (in seconds); MSG: Method that solves most instances with the smallest relative gap; %MSG: Percentage of instances solved by MSG; BGR: Repetition with the best Relative Gap in each instance; WGR: Repetition with the worst Relative Gap in each instance. In each row, the lowest average relative gap between ILS, VNS and SA is shown in bold.

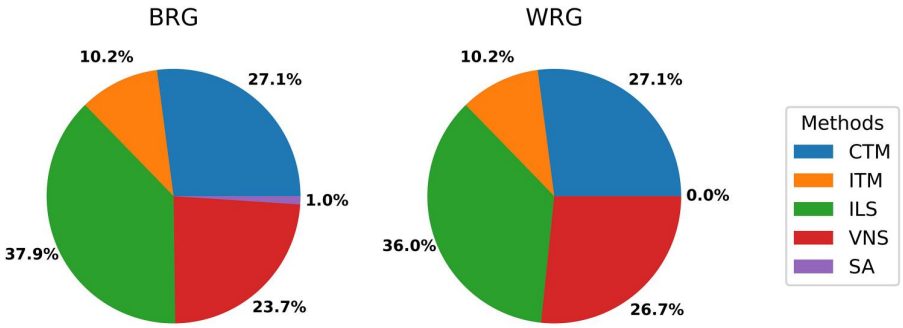

Figure 8. Percentage of instances for best repetition (BRG) and worst repetition (WRG) where each method has the lowest relative gap.

Table 8. Summary of the execution of each metaheuristic in the best repetition.

	Relative gap				Execution time (s)				Number of iterations			
Method	Min.	Ave.	SD	Max.	Min.	Ave.	SD	Max.	Min.	Ave.	SD	Max.
ILS	0.020	**0.325**	**0.432**	**1.977**	300.001	303.696	4.549	336.253	10	79.902	61.403	309
VNS	0.020	0.326	0.433	1.984	300.002	442.321	497.098	3982.884	1	33.531	30.078	145
SA	0.020	0.347	0.456	2.038	**6.524**	**34.262**	**20.188**	**108.645**	457	633.738	51.448	802

Abbreviations are defined as follows: ILS: Iterated Local Search; VNS: Variable neighborhood search; SA: Simulated annealing; Min: Minimum; Ave.: Average; SD: Standard Deviation; Max.: Maximum. In each column, the best value for the relative gap and executuon time between ILS, VNS and SA is shown in bold.

performance of the metaheuristics over the models is evident in the percentage of instances where each method achieves the lowest relative gap compared to the other methods. In the first 18 repetitions, ILS achieves the lowest relative gap in most instances, ranging from 31.458% to 36.042%. For the best and worst repetition of metaheuristics, Figure 8 shows that the mathematical models (CTM and ITM) achieve the lowest relative gap in 37.3% of instances, while the metaheuristics (ILS, VNS, and SA) achieve this in 62.7% of instances. Figure 9 compares the relative gap between the models and the best repetitions of each metaheuristic. In the most challenging instances, the performance of metaheuristics is superior to that of both models. Nevertheless, the metaheuristics were unable to find an optimal solution in any of the test instances. Indeed, Table 8 shows that the minimum relative gap in the three metaheuristics is 0.020 for the best repetition.

Figure 9 shows that the metaheuristics perform similarly. Table 7 shows that the highest average relative gap obtained by SA in repetition 3 is only 1.170 times higher than that obtained by VNS in repetition 20. However, SA performs the worst in terms of the average relative gap. Indeed, the lowest average relative gap of SA is 0.397, but the highest average relative gap in the other methods does not exceed 0.360. Furthermore, in 13 repetitions, ILS has the most instances with the best relative gap, followed by VNS in 6 repetitions. In repetition 9, there is a tie between ILS and VNS. In the best repetition, the relative gaps are similar. Table 8 shows that the average relative gaps for ILS, VNS, and SA are 0.325, 0.326, and 0.347, respectively, and the maximum relative gaps are 1.977, 1.984, and 2.038, respectively. Moreover, ILS has the highest number of instances with the best relative gap (37.917%). Therefore, ILS performs better in finding near-optimal solutions.

SA has a better execution time than other methods. For all repetitions, the average execution time in SA ranges from 32.594 to 33.168 s. However, the average execution times in ILS are between 303.633 and 304.226 s and between 306.349 and 598.698 s for VNS. For the best repetition, Table 8 indicates that the execution times range from 6.524 to 108.645 s for SA, 300.001 to 336.253 s for ILS, and 300.002–3982.884 s for VNS, exceeding the detention criteria by more than 13 times. The structure of the algorithms accounts for this difference in execution time. SA is a variant of a local search algorithm, where the diversification strategy involves accepting, with a certain probability, solutions worse than the best previously obtained solution.

In contrast, ILS and VNS are algorithms that repeat local search in their primary cycle. The diversification strategy gives a difference in execution time between these algorithms. In ILS, diversification consists of randomly perturbing a solution. VNS randomly selects a solution from a neighborhood system, changes it if the solution does not improve, and restarts it in the same iteration otherwise. These strategies also explain the differences in the number of iterations between the three methods. Overall, SA has the highest average number of iterations, ranging from 612,735 to 620,158, in contrast to the low average numbers of iterations for VNS (between 9,825 and 43,842) and ILS (between 78,383

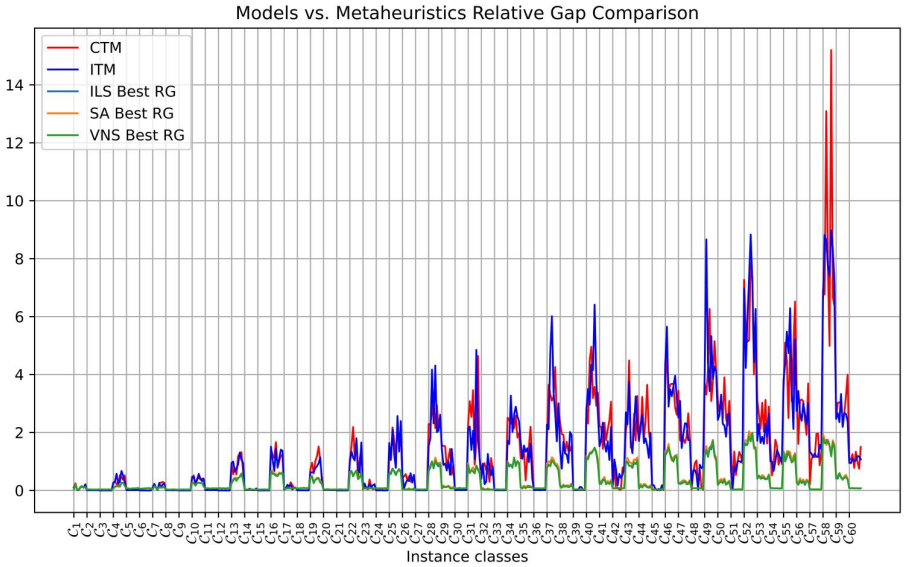

Figure 9. Relative Gap comparison between models and metaheuristics. In the x-axis, all instances of the class $\mathcal{C}_m$ are between classes $\mathcal{C}_m$ and. $\mathcal{C}_{m+1}$

Table 9. Summary of execution of metaheuristics in repetitions with best relative gaps, classified by cluster.

	CTM		ITM		ILS			VNS			SA				
Cluster	ARG	AET(s)	ARG	AET(s)	ARG	ANI	AET(s)	ARG	ANI	AET(s)	ARG	ANI	AET(s)	MSG	%MSG
0	0.051	200.349	**0.046**	200.382	0.052	225.792	300.906	0.052	102.917	301.593	0.052	575.708	10.386	CTM	66.667
1	0.312	300.803	0.289	300.688	**0.146**	113.708	301.732	0.147	47.417	304.885	0.155	623.563	17.035	CTM	54.167
2	0.128	200.599	0.136	200.599	**0.112**	112.417	301.584	**0.112**	51.667	303.064	0.113	571.729	20.975	CTM	54.167
3	0.884	300.603	0.790	300.953	**0.244**	77.333	302.567	0.247	30.896	309.175	0.261	647.271	23.349	ILS	45.833
4	1.591	300.338	1.503	300.314	0.354	60.229	303.833	**0.353**	24.375	374.580	0.402	660.646	29.028	ILS	41.667
5	0.461	294.509	0.432	294.097	0.237	57.542	303.253	**0.236**	23.500	307.779	0.243	615.396	33.682	ILS	45.833
6	2.743	300.377	2.760	300.439	**0.512**	52.500	303.256	0.517	19.333	546.032	0.557	680.958	35.843	VNS	58.333
7	1.210	300.786	1.207	300.695	**0.358**	39.625	305.205	0.359	14.938	350.253	0.377	641.875	45.632	ILS	33.333
8	2.007	300.356	2.097	300.386	**0.520**	32.646	307.070	0.522	11.583	647.312	0.556	659.417	57.906	ILS	58.333
9	3.713	300.443	3.504	300.446	0.714	27.229	307.554	**0.713**	8.688	978.535	0.750	660.813	68.781	VNS	54.167

Abbreviations are defined as follows: CTM: Continuous-time model; ITM: Integer-time model; ILS: Iterated local Search; VNS: Variable neighborhood search; SA: Simulated annealing; ARG: Average relative gap; AET(s): Average execution time (in seconds); ANI: Average number of iterations; MSG: Method that solves most instances with the smallest relative gap; %MSG: Percentage of instances solved by MSG. In each row, the lowest average relative gap between ILS, VNS and SA is shown in bold.

and 83,769). In the best repetition, the number of SA iterations ranges from 457 to 802. In contrast, the number of iterations of VNS and ILS ranges from 1 to 145 and 10 to 309, respectively.

Table 9 compares the performance of the metaheuristics in terms of the best relative gap with that of the mathematical models, categorizing the instances by cluster. The first column shows the cluster. The following columns show the average relative gap and execution time of CTM and ITM, respectively. The following columns show the average relative gap, the average number of iterations, and the average execution time of ILS, VNS, and SA. The last two columns indicate the method that achieved the best relative gap in the highest number of instances and the percentage of those instances. Figure 10 shows the percentage of instances in each cluster where each method has the lowest relative gap compared to the other methods.

Mathematical models only outperform metaheuristics in the smallest-sized instances. Table 9 shows that the average relative gap of CTM and ITM in the best repetition outperforms the metaheuristics in cluster 0. Specifically, the average relative gap in cluster 0 is 0.051 for CTM, 0.046 for ITM, and 0.052 for ILS, VNS, and SA in the best repetition. The models' superiority in this cluster is also observed in the significant percentage of instances where the lowest relative gap is obtained by CTM (66.667%). Figure 10 shows that in cluster 0, only 12.5% of instances obtain the lowest relative gap with ILS, and none with VNS and SA. Although in clusters 1 and 2, the method with the most instances with the lowest relative gap is CTM (54,167% in both clusters), each metaheuristic has a lower average relative gap than each model. In the other clusters, the average relative gap of ILS, VNS, and SA is smaller than the average relative gap obtained by the models. In clusters 3, 4, 5, 7, and 8, the method with the most significant number of instances with the lowest relative gap is ILS, while VNS is the most effective for clusters 6 and 9. Figure 10 shows that

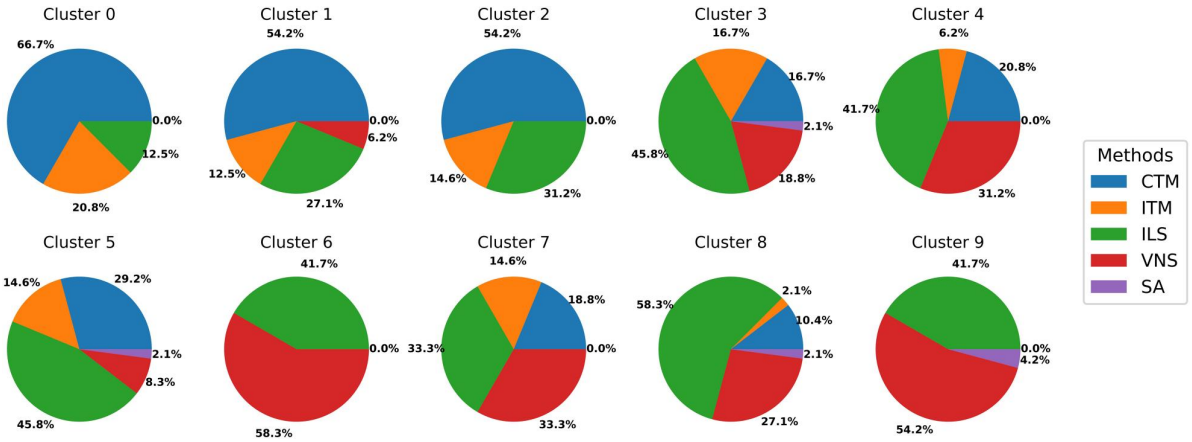

Figure 10. Percentage of instances grouped by cluster, where each method has the lowest gap.

in clusters 6 and 9, the models achieved the lowest relative gap in no instance. In other words, a metaheuristic solved all tested instances in these clusters with a relative gap less than or equal to that obtained by the models. Moreover, the difference between the average relative gap of the models and each metaheuristic is more than 2.000 in the clusters with the largest-sized instances (clusters 6 and 9).

Across all clusters, SA is the metaheuristic that performs the worst regarding the average relative gap but is the best regarding average execution time. In cluster 0, the average relative gap is 0.052 for all three metaheuristics. In the remaining clusters, the average relative gap of SA is more significant, ranging from 0.001 (cluster 2) to 0.048 (cluster 4) units, compared to the rest. Moreover, the average relative gaps of ILS and VNS in the remaining clusters are similar, with differences ranging from 0 (clusters 0 and 2) to 0.005 (cluster 6). The average execution time of SA on the set of most difficult instances (Cluster 9) is 68.781 s, which is significantly lower than the average execution time of ILS (300.906 s) and VNS (301.593 s) on the set of small instances (Cluster 0). Moreover, as the size of the instances increases, the growth rate of the average execution time of VNS is significantly higher than that of ILS. For the most challenging set of instances, the execution time of ILS is 307.554 s, while that of VNS is 978.535 s. As the size of the instances increases, the average relative gaps and execution times rise in each method. In contrast, the average number of iterations decreases.

Similar to Tables 9, 10 compares the results of the metaheuristics with those of the models, grouping the instances by their control parameters. Each row group classifies the instances by the number of RPs, stations, resource types, and time horizon. Figure 11 shows the percentage of instances in each group where each method has the lowest relative gap compared to the other methods.

The efficiency of the metaheuristic is notorious in instances with many RPs. Figure 11 shows that the metaheuristics outperform the models in all instances with 30 RPs. Table 10 shows that in all instance sets grouped by the number of RPs, the highest average relative gap of the metaheuristics is smaller than that of the models. For the set of instances with 10 RPs, the difference between the average relative gap of the CTM (0.089) and SA (0.083) is only 0.006. In the 30-RP instances, the difference between the ITM (3.132) and SA (0.654) is 2.478. Although metaheuristics exhibit better average performance in sets with 10 and 15 PRs, the CTM has the highest percentage of instances with the smallest relative gap (60.417% and 41.667%, respectively). Figure 11 shows that the metaheuristics achieve the smallest relative gap, at 21.9% and 44.8%, respectively.

In all instance sets ranked by RPs, ILS has the lowest average relative gap, and SA has the lowest average execution time among the metaheuristics. Only in the sets with 10, 15, and 25 RPs does VNS have the same average relative gap as ILS. The average relative gap between ILS and VNS is only 0.002 for the other sets. Additionally, the average relative gap difference with SA ranges from 0.001 to 0.042 for ILS and VNS. In Figure 11, the percentage of instances where a metaheuristic has the lowest relative gap increases as the number of RPs increases. In the set of instances with 10 RPs, the ILS only solves 21.9% of the instances with the lowest relative gap. However, in the following sets, not only does the percentage of instances solved by ILS with the lowest relative gap increase but also the percentage of instances solved by VNS and SA, which remains at 1.0%, until the set with 30 PRs, where some metaheuristic solves all instances with a lower relative gap. As the number of PRs increases, the performance of the metaheuristics decreases not only in terms of the average relative gap but also in terms of increasing the average execution time. Indeed,

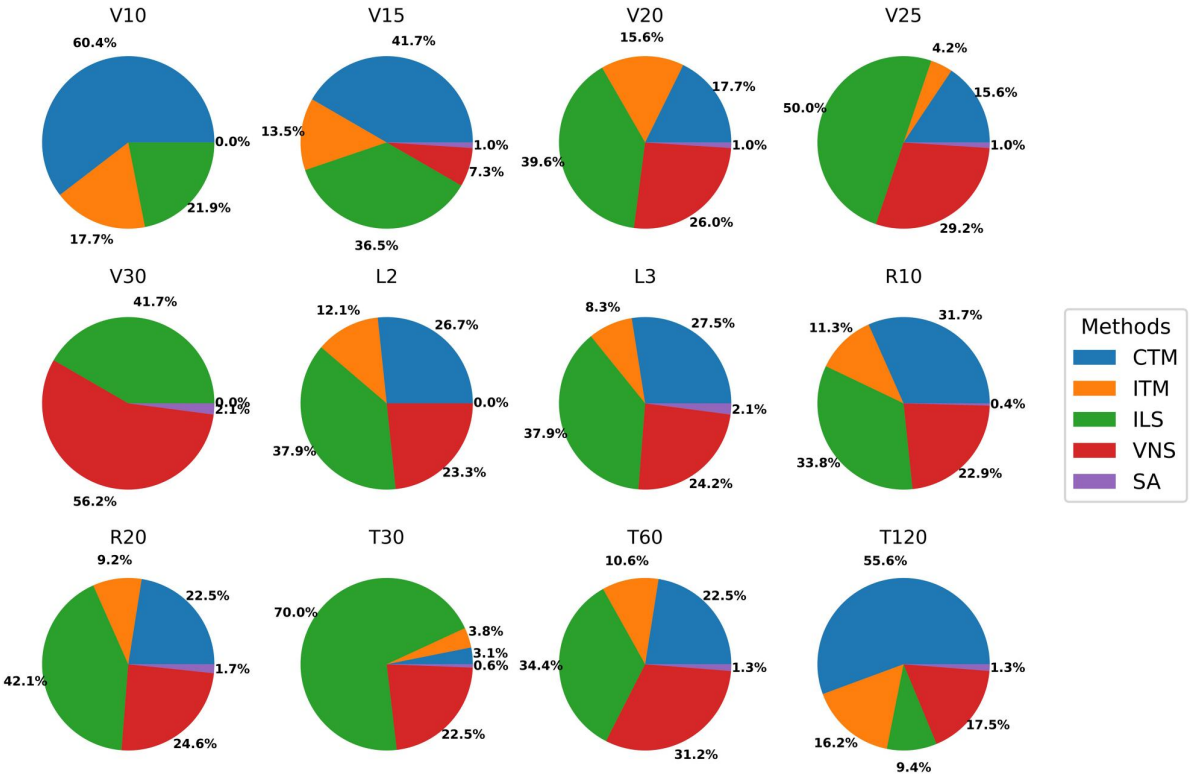

Figure 11. Percentage of instances grouped by the control parameter where each method has the best gap.

Table 10. Summary of the execution of the metaheuristic in repetitions with the best relative gaps, classified by the control parameter.

	CTM		ITM		ILS			VNS			SA				
CP	ARG	AET	ARG	AET	ARG	ANI	AET	ARG	ANI	AET	ARG	ANI	AET	MSG	%MSG
V10	0.089	200.474	0.091	200.490	**0.082**	169.104	301.245	**0.082**	77.292	302.328	0.083	573.719	15.681	CTM	60.417
V15	0.387	297.656	0.361	297.392	**0.192**	85.625	302.492	**0.192**	35.458	306.332	0.199	619.479	25.358	CTM	41.667
V20	1.047	300.695	0.998	300.824	**0.301**	58.479	303.886	0.303	22.917	329.714	0.319	644.573	34.490	ILS	39.583
V25	1.799	300.347	1.800	300.350	**0.437**	46.438	305.451	**0.437**	17.979	510.946	0.479	660.031	43.467	ILS	50.000
V30	3.228	300.410	3.132	300.442	**0.613**	39.865	305.405	0.615	14.010	762.284	0.654	670.885	52.312	VNS	56.250
L2	1.235	280.580	1.220	280.545	**0.325**	95.525	302.827	0.328	41.096	391.000	0.349	635.417	27.484	ILS	37.917
L3	1.385	279.253	1.333	279.255	**0.324**	64.279	304.565	**0.324**	25.967	493.641	0.345	632.058	41.039	ILS	37.917
R10	1.116	280.494	1.078	280.555	**0.262**	105.913	302.459	0.263	44.988	367.253	0.286	637.629	23.128	ILS	33.750
R20	1.504	279.338	1.475	279.244	**0.388**	53.892	304.933	**0.388**	22.075	517.389	0.408	629.846	45.395	ILS	42.083
T30	2.491	300.332	2.531	300.309	**0.776**	92.500	302.848	0.778	40.688	305.561	0.820	612.731	29.433	ILS	70.000
T60	1.172	300.698	1.025	300.735	**0.146**	74.350	304.305	0.147	30.325	406.996	0.167	643.794	35.014	ILS	34.375
T120	0.266	238.718	0.275	238.655	**0.053**	72.856	303.935	**0.053**	29.581	614.405	**0.053**	644.688	38.338	CTM	55.625

Abbreviations are defined as follows: CP: Control parameter; CTM: Continuous-time model; ITM: Integer-time model; ILS: Iterated local Search; VNS: Variable neighborhood search; SA: Simulated annealing; ARG: Average relative gap; AET(s): Average execution time (in seconds); ANI: Average number of iterations; MSG: Method that solves most instances with the smallest relative gap; %MSG: Percentage of instances solved by MSG. In each row, the lowest average relative gap between ILS, VNS and SA is shown in bold.

SA's average execution time increases from 15.681 s to 52.312 s. ILS has a slight increase in rate, from 301.245 s to 305.451 s, but with a higher average execution time than SA, and this is the only case where the maximum execution time occurs in the group of instances with 25 PRs. VNS's average execution time increases from 302.238 s to 762.284 s. Thus, as the number of PRs increases, the average number of ILS iterations decreases from 169.104 to 39.865, and the average number of VNS iterations decreases from 77.292 to 14.010. Only the average number of SA iterations increases from 573.719 to 670.885. This increase is due to two reasons: (i) by increasing the number of projects, an increasing number of possible lists (solutions) are available; (ii) by exploring an increasing number of solutions, there is a higher probability that SA will find a better solution in each iteration, which maintains a slow cooling rate and generates a more significant number of iterations.

The metaheuristics performance is superior to the models for the 2-station and 3-station instance sets. Indeed, Table 10 shows that for the set of instances with two stations, the average relative gap of each metaheuristic is no more than 0.349. However, the average relative gap of the models is not less than 1.220. Similarly, for the set of instances with three stations, the average relative gap of the models is at

least 1.333, while the average relative gap of the metaheuristics is no more than 0.345. Additionally, the average relative gap variation differs between models and metaheuristics as the number of stations varies. As the number of stations increases, the average relative gap rises from 0.113 to 0.150 for the models, decreasing by approximately 0.001 to 0.004 for the metaheuristics. These differences in the behavior of the average relative gap are because, as the number of stations increases, the number of variables and constraints in the model also increases, making it more challenging to solve. Nevertheless, it also increases the number of options for the list-based scheduling algorithm to select and schedule more PRs, thereby increasing the total revenue. The superiority of the metaheuristics can also be seen in the percentage of instances where each method achieves the smallest relative gap. In both sets, ILS solves 37.917% of the instances with the smallest relative gap. Moreover, Figure 11 shows that in the sets with 2 and 3 stations, the metaheuristics solve 61.2% and 64.2% of the instances with the smallest relative gap, respectively.

In both sets of instances classified by the number of stations, the metaheuristics perform similarly regarding the average relative gap but differ in execution time and the number of iterations. In Table 10, the average relative gap ranges from 0.325 (ILS) to 0.349 (SA) for the set with two stations. The average relative gap for the set with three stations ranges from 0.324 (ILS and VNS) to 0.345 (SA). Figure 11 shows that SA does not resolve any instances with two stations having the lowest relative gap. In this set, ILS and VNS resolve 37.9% and 23.3% of the instances, respectively, with the lowest relative gap. However, SA achieves the lowest average relative gap in 2.1% of the instances with three stations, ILS maintains 37.9%, and VNS increases to 24.2% of the instances. Regarding the average execution time, SA's performance is superior to that of the other metaheuristics. In the sets with 2 and 3 stations, the average execution times vary between 27.404 s (SA) and 302.827 s (VNS), and between 41.039 s (SA) and 493.641 s (VNS), respectively. In addition, the average execution time of SA is lower than that of the mathematical models, which is no less than 279.253 s. When the number of stations varies, the average ILS execution time shows a minor variation of 1.738 s. This difference is substantial in VNS, with an average of 102.641 s, but considerably smaller in SA, at 13.555 s. The average number of iterations also decreases as the number of stations increases from 2 to 3. Table 10 shows that the average number of iterations decreases from 95.525 to 64.279 in ILS, 41.096 to 25.967 in VNS, and 635.417 to 632.058 in SA. These values indicate that SA requires more iterations to solve the problem, whereas VNS solves it with fewer iterations.

When the number of resource types groups instances, we find that the metaheuristics outperform the mathematical models. Table 10 indicates that for instances with 10 resource types, the difference between the best average relative gap of the models (1.078) and the metaheuristic with the highest value (SA at 0.286) is 0.792. Similarly, for instances with 20 resource types, the difference between the best relative gap of the models (1.475) and the metaheuristic with the highest value (SA with 0.408) is 1.067. This superiority is also evident in the high percentage of instances in each group where the metaheuristics had a relative gap less than or equal to that of the models. Indeed, ILS solves 33.750% of the instances with 10 resource types, achieving the smallest relative gap and 42.083% of the instances with 20 resource types. Additionally, Figure 11 illustrates that the metaheuristics successfully solve 57.1% of the instances with 10 resource types, achieving the smallest relative gap. This value increases to 68.4% when considering instances with 20 resource types.

In each set of instances classified by resource type, the metaheuristics perform similarly regarding the average relative gap. Table 10 shows that the average relative gap of the set of instances with 10 and 20 resource types varies between 0.262 (ILS) and 0.286 (SA) and between 0.388 (ILS and VNS) and 0.408 (SA), respectively. Furthermore, for all metaheuristics presented in this study, the average relative gap increases as the number of resource types increases. These variations are 0.126, 0.125, and 0.122 for ILS, VNS, and SA, respectively. Figure 11 shows that with 10 resource types, SA solves 0.4% of instances with the smallest relative gap. In this set, ILS solves 33.8% of the instances with the smallest relative gap, and VNS solves 22.9%. These percentages increase in the set of instances with 20 resource types. ILS solves 42% of the instances with the smallest relative gap, VNS solves 24.6%, and SA solves 1.7%. The average execution time of SA is lower than that of the other metaheuristics. For instances with 10 resource types, the time varies between 23.128 (SA) and 367.253 (VNS) seconds, and for instances with 20 resource types, between 45.395 (SA) and 517.389 (VNS) seconds. The average number of iterations decreases as resource types increase from 10 to 20. Table 10 shows that the average number of iterations decreases from 105.913 to 53.892 in ILS, 44.988 to 22.075 in VNS, and 637.629 to 629.846 in SA. These values indicate that SA requires more iterations to solve the problem, whereas VNS solves it with fewer iterations.

Regarding the average relative gap, the metaheuristics perform better when the instances are grouped

by time horizon. Table 10 shows that, for the sets with 30 and 120-period horizons, the average relative gap between the worst-performing metaheuristic (SA) and the best-performing model (CTM) is 0.671 and 0.213, respectively. For the 60-period time horizon set, the difference between the worst-performing metaheuristic (SA) and the best-performing model (ITM) is 0.858. Although the performance of the metaheuristics is better on average, in the set with a time horizon of 120 periods, the CTM solves 55.625% of the instances with the smallest relative gap. Figure 11 shows that the metaheuristic with the smallest relative gap solves only 28.2% of the instances. In the other sets (with 30 and 60 periods), ILS solves 70.000% and 34.375% of the instances with a smaller relative gap, respectively.

Large time horizons make selecting and scheduling RPs more manageable. In particular, the list-scheduling-based algorithm finds it more accessible to perform scheduling from a list when the time horizon is long. Table 10 shows that the average relative gap obtained by the metaheuristics decreases as the time horizon increases. For ILS, as the time horizon increases from 30 to 60 and then to 120 periods, the average relative gap decreases from 0.776 to 0.146 and then to 0.053. Similarly, for VNS, the average relative gap decreases from 0.778 to 0.147 to 0.053; for SA, it decreases from 0.820 to 0.167 and then to 0.053, and the difference between the metaheuristic with the best average relative gap (ILS) and the one with the worst (SA) decreases as the time horizon increases. Thus, this difference is 0.044 for instances with 30 periods and 0.021 for instances with 60 periods. For instances with 120 periods, each metaheuristic's average relative gap is the same. Figure 11 shows that SA solves only 0.6% of the 30-period instances with the smallest relative gap and 1.3% of the 60 and 120-period instances. In addition, the performance of ILS is superior in the 30-period and 60-period sets, solving 70.0% and 34.4% of instances, respectively, while VNS solves 22.5% and 31.2% of instances, respectively. However, in the 120-period set, VNS outperforms ILS in the percentage of instances solved with the smallest relative gap, 17.5% versus 9.4%, respectively. For VNS and SA, the average execution time increases as the time horizon increases. Indeed, the average execution times of VNS are 305.561, 406.996, and 614.405 s, and those of SA are 29.433, 35.014, and 38.338 s, respectively. However, the average ILS run times oscillate: 302.848, 304.305, and 303.935 s, respectively. According to these values, the execution time of SA is lower than that of the other metaheuristics in all sets, and the average number of iterations, ILS and VNS decrease this number as the time horizon increases. The average number of iterations achieved by ILS is 92.500, 74.350, and 72.856, and by VNS is 40.688, 30.325, and 29.581, respectively. However, SA's average number of iterations is 612.731, 643.794, and 644.688, respectively. The average number of iterations SA achieves increases as the time horizon increases. Similar to increasing the number of PRs, extending the time horizon increases the likelihood of improving a solution. Therefore, the search for solutions in SA occurs at a slow cooling rate.

Figure 12 illustrates the evolution of the objective function value for instance V30-L3-R20-T30-6 as each metaheuristic is executed. This instance shows the most significant improvement in the relative gap between models and metaheuristics. Each graph compares the main cycle iterations (excluding sub-cycles, such as the local search procedure or neighborhood changes) with the objective function value of the best solution obtained in each iteration. The blue and green lines represent the objective function value obtained by CTM and ITM, respectively. Only ILS considers improving the initial solution in iteration 0 in the graph by local search. The title of each graph indicates the method used, the value of the objective function obtained, and the execution time for each method.

The diversification strategy determines the convergence of each metaheuristic. Although the initial solution is randomly generated, its value in the objective function is superior to that of the mathematical models. This value is due to the efficiency of the list-scheduling-based algorithm, which selects and schedules the most significant number of PRs, increasing the total revenue. Premature convergence occurs in ILS and VNS. However, ILS shows an increase in the value of the objective function every few iterations, suggesting that VNS shows almost continuous growth throughout its execution. On the contrary, after the first iteration, VNS keeps the value of the objective function constant until it stops because, in ILS, the improvement can occur within the primary cycle of the method. In turn, in VNS, the improvement can occur in each sub-cycle of the neighborhood change, which restarts each time an improvement occurs. Since SA is a modification of local search, this method does not have sub-cycles. In addition, the high probability of accepting solutions that are worse than the current solution allows escape from local optima. However, as this probability decreases, the value of the objective function tends to stabilize, as depicted in the third graph in Figure 12.

The results highlight the significant advantages of the proposed metaheuristics in solving the RPSAP, particularly for larger and more complex problem instances. ILS consistently achieved the best relative gaps across most test instances, showcasing its ability to balance exploration and exploitation through its

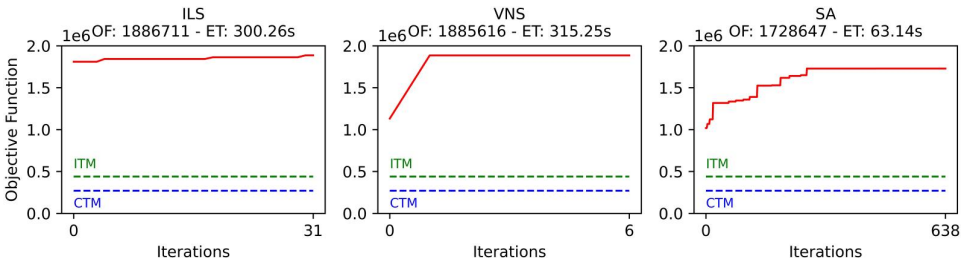

Figure 12. Evolution of the objective function value for the V30-L3-R20-T30-6 instance with each metaheuristic, ILS, VNS, and SA iteration, respectively. The "OF" and "ET" values represent the objective function value and the execution time after each metaheuristic run.

customized perturbation mechanism and project group swap neighborhood system. VNS also demonstrated competitive performance by systematically exploring multiple neighborhood systems, making it particularly effective for mid-sized problem instances. SA excelled in execution time, providing a rapid yet effective search process due to its adaptive cooling schedule and efficient acceptance criteria. Overall, the metaheuristics outperformed exact approaches, particularly regarding scalability and computational efficiency, with ILS yielding the most robust results regarding solution quality. These findings validate the effectiveness and suitability of the proposed methods for addressing large-scale, resource-constrained scheduling problems.

6. Conclusions

The study presented in this paper addresses the Antarctic Research Project Selection and Scheduling Problem (RPSAP), a novel and complex combinatorial optimization challenge. Our contributions include a mixed-integer programming (MIP) model and robust metaheuristic approaches, including Iterated Local Search (ILS), Variable Neighborhood Search (VNS), and Simulated Annealing (SA), designed to tackle the inherent complexities of resource allocation, project precedence, transportation delays, and environmental considerations in Antarctic operations. Through extensive computational experiments on 480 test instances across 60 distinct classes, we provided empirical evidence supporting the efficacy and scalability of our proposed methods.

This research introduces the first formal application of project portfolio selection and scheduling in the Antarctic context. Key contributions include:

- Theoretical Advancements: Developing a continuous-time MIP model incorporates critical aspects of RPSAP, including resource sharing, station-specific constraints, and environmental impacts. Additionally, the design of metaheuristics (ILS, VNS, and SA) is customized to the problem, providing scalable alternatives to exact solvers for large problem instances.
- Practical Implications: A decision-support framework that Antarctic research managers can adapt to prioritize and schedule projects, balancing environmental sustainability and scientific outcomes. This framework offers insights into the trade-offs between exact and heuristic solutions, highlighting scenarios where metaheuristics are more suitable for real-world applications.
- Empirical Validation: The generation and analysis of realistic test instances reveal that ILS consistently outperforms other methods regarding relative gap and solution quality for larger problem sizes. This finding establishes that metaheuristics, particularly ILS, can effectively solve instances with up to 30 projects, where commercial solvers often fail to achieve optimality within reasonable timeframes.

The study bridges the gap between theoretical operations research and practical logistics in Antarctic science. The proposed MIP model extends the literature on resource-constrained project selection and scheduling by incorporating a multi-site framework, transportation delays, and environmental considerations. For research managers, this work provides actionable tools to enhance resource utilization, reduce ecological footprints, and maximize the scientific profit of research portfolios.

Despite the significant contributions, the study has limitations:

- Scalability of Exact Models: Due to computational constraints, the MIP model's applicability is limited to small and medium-sized instances.
- Generalizability: Although the designed test instances mimic real-world scenarios, they may not fully capture the complexities encountered in Antarctic operations.
- Simplified Cost Structures: Certain operational costs, such as maintenance and long-term environmental impacts, were not explicitly accounted for in the model.
- Limitations of the list-scheduling-based algorithm: The algorithm relies heavily on the RP list

from metaheuristics and maintains fixed site and resource orders, which limits its ability to find optimal solutions, even in small instances with a minimum relative gap of 0.020.

To address these limitations and further advance the field, future research could explore:

- Model Enhancements: Incorporating stochastic elements to account for uncertainties in project durations, resource availability, and environmental conditions while extending the model to include non-linear cost structures and dynamic resource allocation policies and, additionally, developing a multi-objective model to balance goals like maximizing scientific profit, minimizing environmental impact, and optimizing resource use, offering a more integrated decision-making framework.
- Metaheuristic Development: Designing hybrid metaheuristics that combine the strengths of ILS with other approaches, such as genetic algorithms or machine learning-based methods, and exploring parallel computing frameworks to accelerate solution times for large-scale instances.
- Practical Applications: Adapting the framework for other extreme environments, such as space missions or deep-sea research, where similar logistical constraints exist, or collaborating with Antarctic research institutions to validate the model's effectiveness in operational settings and refine it based on stakeholder feedback.

This study marks a significant step toward optimizing Antarctic research operations, offering theoretical advancements and practical tools for decision-makers. By addressing the unique challenges of this context, we contribute to the broader goal of enabling sustainable and impactful scientific endeavors in extreme environments.

Acknowledgments

The authors utilize the following AI tools to enhance their language: Grammarly (v1.2.130.1584), DeepL (v25.1.4.15077), and ChatGPT (GPT-4o).

Disclosure statement

The authors declare that they have no conflict of interest.

Funding

FIN ANID supported this work under Grant AFB230002; and ANID—Subdirección de Capital Humano/Doctorado Nacional/2021 under Grant 21210451.

ORCID

Mauricio Vega-Hidalgo http://orcid.org/0000-0001-9595-9051
Lorena Pradenas http://orcid.org/0000-0001-5199-7187
Víctor Parada http://orcid.org/0000-0002-8649-5694

Data availability statement

The data supporting this study's findings are openly available in Zenodo at https://doi.org/10.5281/zenodo.14721986.

References

Alinaghian, M., Sajadi, S. M., & Ganji, M. (2016). A new model for optimising simultaneously projects selection and resource-constrained project scheduling problem. *International Journal of Productivity and Quality Management*, *19*(4), 511. https://doi.org/10.1504/IJPQM.2016.10000356

Alinezhad, R., Ansari, R., Mahdikhani, M., & Banihashemi, S. A. (2022). Multi-phase projects selection and scheduling problem: A multi-objective optimization approach. *Iranian Journal of Science and Technology, Transactions of Civil Engineering*, *46*(3), 2575–2591. https://doi.org/10.1007/s40996-021-00721-9

Arratia-Martinez, N. M., Hernandez-Gonzalez, N. M., & Lopez-Irarragorri, F. (2021). Project portfolio selection and scheduling with resource allocation, synergies, and project divisibility. *Mathematical Problems in Engineering*, *2021*, 1–14. https://doi.org/10.1155/2021/4163287

Artigues, C. (2008). The resource-constrained project scheduling problem. In C. Artigues, S. Demassey, & E. Néron (Eds.), *Resource-Constrained Project Scheduling: Models, Algorithms, Extensions and Applications* (pp. 19–35) John Wiley & Sons Inc. https://doi.org/10.1002/9780470611227.ch1

Artigues, C. (2017). On the strength of time-indexed formulations for the resource-constrained project scheduling problem. *Operations Research Letters*, *45*(2), 154–159. https://doi.org/10.1016/j.orl.2017.02.001

Bagloee, S. A., Sarvi, M., Patriksson, M., & Asadi, M. (2018). Optimization for roads' construction: Selection, prioritization, and scheduling. *Computer-Aided Civil and Infrastructure Engineering*, *33*(10), 833–848. https://doi.org/10.1111/mice.12370

Bai, L., Yang, M., Pan, T., & Sun, Y. (2025). Project portfolio selection and scheduling incorporating dynamic synergy. *Kybernetes*, *54*(2), 996–1026. https://doi.org/10.1108/K-04-2023-0694

Bigler, T., Gnägi, M., & Trautmann, N. (2024). MIP-based solution approaches for multi-site resource-constrained project scheduling. *Annals of Operations Research*, *337*(2), 627–647. https://doi.org/10.1007/s10479-022-05109-0

Casado, R. S. G. R., Alencar, M. H., & de Almeida, A. T. (2022). Combining a multidimensional risk evaluation with an implicit enumeration algorithm to tackle the portfolio selection problem of a natural gas pipeline. *Reliability Engineering & System Safety*, *221*, 108332. https://doi.org/10.1016/j.ress.2022.108332

Chang, P. T., & Lee, J. H. (2012). A fuzzy DEA and knapsack formulation integrated model for project selection.

Computers & Operations Research, *39*(1), 112–125. https://doi.org/10.1016/j.cor.2010.10.021

Chen, H., Dou, Y., Chen, Z., Jia, Q., Zhang, M., & Qiu, B. (2023). A deep reinforcement learning algorithm to project portfolio selection problem [Paper presentation]. In *2023 5th International Communication Engineering and Cloud Computing Conference* (pp. 21–25), IEEE. https://doi.org/10.1109/CECCC59577.2023.10560493

Chen, J., & Askin, R. G. (2009). Project selection, scheduling and resource allocation with time dependent returns. *European Journal of Operational Research*, *193*(1), 23–34. https://doi.org/10.1016/j.ejor.2007.10.040

Chien, C. F., & Huynh, N. T. (2018). An integrated approach for IC design R&D portfolio decision and project scheduling and a case study. *IEEE Transactions on Semiconductor Manufacturing*, *31*(1), 76–86. https://doi.org/10.1109/TSM.2018.2792783

COMNAP (2017). *Antarctic station catalogue*. Council of Managers of National Antarctic Programs. https://www.comnap.aq/antarctic-facilities-information

Gnagi, M., Trautmann, N. (2019). A continuous-time mixed-binary linear programming formulation for the multi-site resource-constrained project scheduling problem. In *IEEE International Conference on Industrial Engineering and Engineering Management* (pp. 611–614). IEEE. https://doi.org/10.1109/IEEM44572.2019.8978811

Gourgand, M., Grangeon, N., & Klement, N. (2015). Activities planning and resources assignment on distinct places: A mathematical model. *RAIRO - Operations Research*, *49*(1), 79–98. https://doi.org/10.1051/ro/2014028

Harrison, K. R., Elsayed, S., Garanovich, I. L., Weir, T., Galister, M., Boswell, S., Taylor, R., & Sarker, R. (2021). A hybrid multi-population approach to the project portfolio selection and scheduling problem for future force design. *IEEE Access*, *9*, 83410–83430. https://doi.org/10.1109/ACCESS.2021.3086070

Harrison, K. R., Elsayed, S. M., Weir, T., Garanovich, I. L., Boswell, S. G., & Sarker, R. A. (2022). Solving a novel multi-divisional project portfolio selection and scheduling problem. *Engineering Applications of Artificial Intelligence*, *112*, 104771. https://doi.org/10.1016/j.engappai.2022.104771

Hosseininasab, S. M., & Shetab-Boushehri, S. N. (2015). Integration of selecting and scheduling urban road construction projects as a time-dependent discrete network design problem. *European Journal of Operational Research*, *246*(3), 762–771. https://doi.org/10.1016/j.ejor.2015.05.039

Hosseininasab, S. M., Shetab-Boushehri, S. N., Hejazi, S. R., & Karimi, H. (2018). A multi-objective integrated model for selecting, scheduling, and budgeting road construction projects. *European Journal of Operational Research*, *271*(1), 262–277. https://doi.org/10.1016/j.ejor.2018.04.051

Hu, S., Zhang, Z., Wang, S., Kao, Y., & Ito, T. (2019). A project scheduling problem with spatial resource constraints and a corresponding guided local search algorithm. *Journal of the Operational Research Society*, *70*(8), 1349–1361. https://doi.org/10.1080/01605682.2018.1489340

Jovanovic, U., Shayanfar, E., & Schonfeld, P. M. (2018). Selecting and scheduling link and intersection improvements in urban networks. *Transportation Research Record: Journal of the Transportation Research Board*, *2672*(51), 1–11. https://doi.org/10.1177/0361198118758681

Kadri, R. L., & Boctor, F. F. (2018). An efficient genetic algorithm to solve the resource-constrained project scheduling problem with transfer times: The single mode case. *European Journal of Operational Research*, *265*(2), 454–462. https://doi.org/10.1016/j.ejor.2017.07.027

Kennicutt, M., Chown, S., Cassano, J., Liggett, D., Massom, R., Peck, L., Rintoul, S., Storey, J., Vaughan, D., Wilson, T., Sutherland, W., Allison, I., Ayton, J., Badhe, R., Baeseman, J., Barrett, P., Bell, R., Bertler, N., Sun, B., & Yang, H. (2014). Six priorities for antarctic science. *Nature*, *512*(7512), 23–25. https://doi.org/10.1038/512023a

Kennicutt, M. C., II, Chown, S. L., Cassano, J. J., Liggett, D., Peck, L. S., Massom, R., Rintoul, S. R., Storey, J., Vaughan, D. G., Wilson, T. J., Allison, I., Ayton, J., Badhe, R., Baeseman, J., Barrett, P. J., Bell, R. E., Bertler, N., Bo, S., Brandt, A., Bromwich, D., Cary, S. C., … Sutherland, W. J. (2015). A roadmap for Antarctic and Southern Ocean science for the next two decades and beyond. *Antarctic Science*, *27*(1), 3–18. https://doi.org/10.1017/S0954102014000674

Kirkpatrick, S., Gelatt, C. D., & Vecchi, M. P. (1983). Optimization by simulated annealing. *Science*, *220*(4598), 671–680. https://doi.org/10.1126/science.220.4598.671

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

Koné, O., Artigues, C., Lopez, P., & Mongeau, M. (2011). Event-based MILP models for resource-constrained project scheduling problems. *Computers & Operations Research*, *38*(1), 3–13. https://doi.org/10.1016/j.cor.2012.10.018

Kumar, M., Mittal, M. L., Soni, G., & Joshi, D. (2018). A hybrid TLBO-TS algorithm for integrated selection and scheduling of projects. *Computers & Industrial Engineering*, *119*, 121–130. https://doi.org/10.1016/j.cie.2018.03.029

Kumar, M., Mittal, M. L., Soni, G., & Joshi, D. (2019). A Tabu search algorithm for simultaneous selection and scheduling of projects. In N. Yadav, A. Yadav, J. C. Bansal, K. Deep, & J. H. Kim (Eds.), *Harmony Search and Nature Inspired Optimization Algorithms Theory and Applications. Advances in Intelligent Systems and Computing* (Vol. 741, pp. 1111–1121) Springer Verlag. https://doi.org/10.1007/978-981-13-0761-4_104

Laurent, A., Deroussi, L., Grangeon, N., & Norre, S. (2017). A new extension of the RCPSP in a multi-site context: Mathematical model and metaheuristics. *Computers & Industrial Engineering*, *112*, 634–644. https://doi.org/10.1016/j.cie.2017.07.028

Li, X., Fang, S. C., Guo, X., Deng, Z., & Qi, J. (2016). An extended model for project portfolio selection with project divisibility and interdependency. *Journal of Systems Science and Systems Engineering*, *25*(1), 119–138. https://doi.org/10.1007/s11518-015-5281-1

Li, X., Fang, S. C., Tian, Y., & Guo, X. (2015). Expanded model of the project portfolio selection problem with divisibility, time profile factors and cardinality constraints. *Journal of the Operational Research Society*, *66*(7), 1132–1139. https://doi.org/10.1057/jors.2014.75

Litvinchev, I., Lopez-Irarragorri, F., Arratia-Martínez, N. M., & Marmolejo, J. A. (2014). Selecting large portfolios of social projects in public organizations. *Mathematical Problems in Engineering*, *2014*(1), 654293. https://doi.org/10.1155/2014/654293

Liu, S. S., & Wang, C. J. (2011). Optimizing project selection and scheduling problems with time-dependent resource constraints. *Automation in Construction*, *20*(8), 1110–1119. https://doi.org/10.1016/j.autcon.2011.04.012

Liu, Y., Zhou, J., Lim, A., & Hu, Q. (2023). A tree search heuristic for the resource constrained project scheduling problem with transfer times. *European Journal of Operational Research*, *304*(3), 939–951. https://doi.org/10.1016/j.ejor.2022.05.014

Lourenço, H. R., Martin, O. C., & Stützle, T. (2003). Iterated local search. In: Glover, F., Kochenberger, G.A. (eds) *Handbook of metaheuristics, International Series in Operations Research & Management Science* (vol. 57). Springer. https://doi.org/10.1007/0-306-48056-5_11

Mahmoudi, R., Shetab-Boushehri, S. N., Hejazi, S. R., Emrouznejad, A., & Rajabi, P. (2019). A hybrid egalitarian bargaining game-DEA and sustainable network design approach for evaluating, selecting and scheduling urban road construction projects. *Transportation Research Part E: Logistics and Transportation Review*, *130*, 161–183. https://doi.org/10.1016/j.tre.2019.08.008

Mavrotas, G., Diakoulaki, D., & Kourentzis, A. (2008). Selection among ranked projects under segmentation, policy and logical constraints. *European Journal of Operational Research*, *187*(1), 177–192. https://doi.org/10.1016/j.ejor.2007.03.010

Mavrotas, G., & Makryvelios, E. (2021). Combining multiple criteria analysis, mathematical programming and Monte Carlo simulation to tackle uncertainty in research and development project portfolio selection: A case study from Greece. *European Journal of Operational Research*, *291*(2), 794–806. https://doi.org/10.1016/j.ejor.2020.09.051

Mika, M., Weglarz, J., & Waligóra, G. (2009). Project scheduling problem with multiple modes, multisite renewable resources and transportation network. *IFAC Proceedings Volumes*, *42*(13), 224–227. https://doi.org/10.3182/20090819-3-PL-3002.00039

Miralinaghi, M., Seilabi, S. E., Chen, S., Hsu, Y.-T., & Labi, S. (2020). Optimizing the selection and scheduling of multi-class projects using a Stackelberg framework. *European Journal of Operational Research*, *286*(2), 508–522. https://doi.org/10.1016/j.ejor.2020.03.051

Mirkhorsandi Langaroudi, S. M., Khosravi, H., Davoodi, A., & Movahedifar, S. M. (2023). Cash holding management for self-financing phase-able and non-phase-able project portfolio selection and scheduling problems. *The Engineering Economist*, *68*(2), 82–98. https://doi.org/10.1080/0013791X.2023.2172242

Mladenović, N., & Hansen, P. (1997). Variable neighborhood search. *Computers & Operations Research*, *24*(11), 1097–1100. https://doi.org/10.1016/S0305-0548(97)00031-2

Nascimento, C. R. S. d M. S., Almeida-Filho, A. T. d., & Perez Palha, R. (2025). A TOPSIS-based framework for construction projects' portfolio selection in the public sector. *Engineering, Construction and Architectural Management*, *32*(4), 2553–2570. https://doi.org/10.1108/ECAM-05-2023-0534

Patoghi, A., & Mousavi, S. M. (2021). A new approach for material ordering and multi-mode resource constraint project scheduling problem in a multi-site context under interval-valued fuzzy uncertainty. *Technological Forecasting and Social Change*, *173*, 121137. https://doi.org/10.1016/j.techfore.2021.121137

Pham, V. H. S., & Huynh, L. T. (2024). Optimizing multi-objective construction management for resource-constrained project scheduling problems: A bio-inspired approach. *Asian Journal of Civil Engineering*, *25*(6), 4819–4837. https://doi.org/10.1007/s42107-024-01082-0

Ranjbar, M., Nasiri, M. M., & Torabi, S. A. (2022). Multi-mode project portfolio selection and scheduling in a build-operate-transfer environment. *Expert Systems with Applications*, *189*, 116134. https://doi.org/10.1016/j.eswa.2021.116134

Rostami, M., Bagherpour, M., Mazdeh, M. M., & Makui, A. (2017). Resource pool location for periodic services in decentralized multi-project scheduling problems. *Journal of Computing in Civil Engineering*, *31*(5), 1–16. https://doi.org/10.1061/(asce)cp.1943-5487.0000671

Şahin-Zorluoğlu, Ö., & Kabak, Ö. (2022). An interactive multi-objective programming approach for project portfolio selection and scheduling. *Computers & Industrial Engineering*, *169*, 108191. https://doi.org/10.1016/j.cie.2022.108191

Sarnataro, M., Barbati, M., & Greco, S. (2021). A portfolio approach for the selection and the timing of urban planning projects. *Socio-Economic Planning Sciences*, *75*, 100908. https://doi.org/10.1016/j.seps.2020.100908

SAT (1959). *The Antarctic Treaty*. The Secretariat of the Antarctic Treaty. https://www.ats.aq/e/antarctictreaty.html

Sebt, M. H., Alipouri, Y., & Alipouri, Y. (2013). Solving resource-constrained project scheduling problem with evolutionary programming. *Journal of the Operational Research Society*, *64*(9), 1327–1335. https://doi.org/10.1057/jors.2012.69

Shafahi, A., & Haghani, A. (2018). Project selection and scheduling for phase-able projects with interdependencies among phases. *Automation in Construction*, *93*, 47–62. https://doi.org/10.1016/j.autcon.2018.05.008

Shen, Z., & Li, X. (2023). An extended model of dynamic project portfolio selection problem considering synergies between projects. *Computers & Industrial Engineering*, *179*, 109175. https://doi.org/10.1016/j.cie.2023.109175

Stiti, C., & Driss, O. B. (2019). A new approach for the multi-site resource-constrained project scheduling problem. *Procedia Computer Science*, *164*, 478–484. https://doi.org/10.1016/j.procs.2019.12.209

Sohrabi, A. A., Ghanbari, R., Ghorbani-Moghadam, K., & Sadeghi, S. (2024). A new fuzzy model for multi-criteria project portfolio selection based on modified Kerre's inequality. *OPSEARCH*, *61*(1), 33–50. https://doi.org/10.1007/s12597-023-00685-6

Song, S., Wei, T., Yang, F., & Xia, Q. (2021). Stochastic multi-attribute acceptability analysis-based heuristic algorithms for multi-attribute project portfolio selection and scheduling problem. *Journal of the Operational Research Society*, *72*(6), 1373–1389. https://doi.org/10.1080/01605682.2020.1718018

Tao, X., & Schonfeld, P. (2006). Selection and scheduling of interdependent transportation projects with island models. *Transportation Research Record: Journal of the Transportation Research Board*, *1981*(1), 133–141. https://doi.org/10.3141/1981-21

Tao, Y., Luo, X., Wu, Y., Zhang, L., Liu, Y., & Xu, C. (2023). Portfolio selection of power generation projects

considering the synergy of project and uncertainty of decision information. *Computers & Industrial Engineering*, *175*, 108896. https://doi.org/10.1016/j.cie.2022.108896

Tavakolan, M., Chokan, F., & Dadashi Haji, M. (2024). Simultaneous project portfolio selection and scheduling from contractor perspective. *International Journal of Construction Management*, *24*(3), 298–313. https://doi.org/10.1080/15623599.2023.2222995

Tavana, M., Khalili-Damghani, K., & Sadi-Nezhad, S. (2013). A fuzzy group data envelopment analysis model for high-technology project selection: A case study at NASA. *Computers & Industrial Engineering*, *66*(1), 10–23. https://doi.org/10.1016/j.cie.2013.06.002

Tavana, M., Khosrojerdi, G., Mina, H., & Rahman, A. (2020). A new dynamic two-stage mathematical programming model under uncertainty for project evaluation and selection. *Computers & Industrial Engineering*, *149*, 106795. https://doi.org/10.1016/j.cie.2020.106795

Thiruvady, D., Blum, C., & Ernst, A. T. (2019). Maximising the net present value of project schedules using CMSA and parallel ACO. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* (Vol. 11299, pp. 16–30). Springer International Publishing. https://doi.org/10.1007/978-3-030-05983-5_2

Tofighian, A. A., & Naderi, B. (2015). Modeling and solving the project selection and scheduling. *Computers & Industrial Engineering*, *83*, 30–38. https://doi.org/10.1016/j.cie.2015.01.012

Tselios, D., Papageorgiou, G., Alamanis, N., & Ipsilandis, P. (2024). Energy project portfolio selection and scheduling. *Energy Systems*, *15*(4), 1471–1480. https://doi.org/10.1007/s12667-022-00503-w

Wang, B. (2018). An integer programming model of reinvestment strategy based project portfolio selection and scheduling with constrained resource. In L. Jia, Y. Qin, J. Suo, J. Feng, L. Diao, & M. An (Eds.), *Lecture Notes in Electrical Engineering* (Vol. 483, pp. 551–559) Springer Singapore. https://doi.org/10.1007/978-981-10-7989-4_56

Wu, L.-H., Wu, L., Shi, J., & Chou, Y.-T. (2021). Project portfolio selection considering uncertainty: Stochastic dominance-based fuzzy ranking. *International Journal of Fuzzy Systems*, *23*(7), 2048–2066. https://dx.doi.org/10.1007/s40815-021-01069-y

Xiong, J., Zhou, Z., Tian, K., Liao, T., & Shi, J. (2017). A multi-objective approach for weapon selection and planning problems in dynamic environments. *Journal of Industrial & Management Optimization*, *13*(3), 1189–1211. https://doi.org/10.3934/jimo.2016068

Zhao, S., & Xu, Z. (2021). New closed-loop approximate dynamic programming for solving stochastic decentralized multi-project scheduling problem with resource transfers. *Expert Systems with Applications*, *185*, 115593. https://doi.org/10.1016/j.eswa.2021.115593

Zhao, S., & Xu, Z. (2022). A sealed bid auction-based two-stage approach for a decentralized multiproject scheduling problem with resource transfers. *Applied Intelligence*, *52*(15), 18081–18100. https://doi.org/10.1007/s10489-022-03424-4

Zhang, H., Ma, R., & He, Z. (2024). Project scheduling cost optimization based on resource transfer costs and robustness. *Computers & Operations Research*, *161*, 106445. https://doi.org/10.1016/j.cor.2023.106445

Zolfaghari, S., & Mousavi, S. M. (2021). A novel mathematical programming model for multi-mode project portfolio selection and scheduling with flexible resources and due dates under interval-valued fuzzy random uncertainty. *Expert Systems with Applications*, *182*, 115207. https://doi.org/10.1016/j.eswa.2021.115207

Capítulo 3

Integrated Stochastic Project Selection and Scheduling for Antarctic Research under Uncertain Project Durations

Integrated Stochastic Project Selection and Scheduling for Antarctic Research under Uncertain Project Durations

Mauricio Vega-Hidalgo.[1] Departamento de Ingeniería Industrial, Universidad de Concepción. Edmundo Larenas 219, Concepción, Chile. mauriciovega@udec.cl. ORCID: 0000-0001-9595-9051

Lorena Pradenas. Departamento de Ingeniería Industrial, Universidad de Concepción. Edmundo Larenas 219, Concepción, Chile. lpradena@udec.cl. ORCID: 0000-0001-5199-7187

Víctor Parada. Departamento de Ingeniería Informática, Universidad de Santiago de Chile. Avenida Ecuador 3659, Santiago, Chile. victor.parada@usach.cl. ORCID: 0000-0002-8649-5694

Héctor Cancela. Instituto de Computación, Universidad de la República. Julio Herrera y Reissig 565, Montevideo, Uruguay. cancela@fing.edu.uy. ORCID: 0000-0001-5015-0988

Abstract

We study the Stochastic Research Project Selection and Scheduling Problem (SRPSAP), motivated by Antarctic field campaigns. The planner must select a portfolio of research projects before uncertainty resolves and, for each scenario, produce feasible multi-site schedules that honor precedence, time windows, scarce renewable resources, and inter-station transfer times. We cast SRPSAP as a two-stage stochastic mixed-integer program that maximizes expected net profit and develop four solution approaches: (i) a direct solve with a commercial solver; (ii) an integer L-shaped decomposition (IL-S); (iii) a hybrid IL-S whose scenario subproblems are evaluated by simulated annealing (IL-S+SA); and (iv) a simulated annealing tailored to the whole stochastic problem (SA-SP). The list-scheduling routines handle resource assignments and transfers explicitly and provide fast feasibility checks. On 36 synthetic instances that reflect realistic Antarctic logistics, we compare solution quality and runtime. Exact procedures tend to require excessive computational effort. IL-S achieves strong average gaps and delivers informative bounds, while SA-SP is typically the fastest and frequently attains the best single-run objective values, at the cost of variability across repetitions. The hybrid IL-S+SA can reduce gaps in specific settings (e.g., shorter horizons or fewer stations), whereas IL-S alone is more stable as the number of stations and resource types increases. These results suggest that explicitly modeling uncertainty and combining decomposition with scalable metaheuristics yield practical, high-value plans for polar operations. We release a reproducible instance set and discuss extensions to multiobjective risk control, correlated uncertainty, and richer air–sea logistics.

Keywords: Stochastic multi-site project portfolio selection and scheduling; Two-stage stochastic mixed-integer programming; Integer L-shaped decomposition; Hybrid decomposition–metaheuristic approach; Simulated annealing metaheuristic; Antarctic research logistics.

---

[1] Corresponding author

1. Introduction

Planning and executing scientific projects in Antarctica is a complex optimization challenge, as it requires balancing the high scientific value of research activities with the severe logistical and environmental constraints of operating in polar regions. Antarctica plays a fundamental role in regulating global climate, ocean circulation, and ecological processes, making sustained research on the continent for understanding global environmental change (Kennicutt et al., 2015). Yet, the extreme remoteness, limited transportation capacity, and unpredictable weather conditions impose significant uncertainty on project execution, often affecting project durations and resource availability (Kennicutt et al., 2014). These unique conditions mean that research planning cannot rely on deterministic schedules but instead demands optimization approaches that explicitly incorporate uncertainty in activity durations and scarce resource allocation.

This study addresses the Stochastic Research Project Selection and Scheduling in Multiple Antarctic Stations Problem (SRPSAP), a novel extension of classical stochastic project portfolio and scheduling models. The problem requires a decision-maker to select, in the first stage, which projects to pursue given their expected scientific and economic value, and to determine, in the second stage, how to schedule the chosen projects across multiple stations once uncertain activity durations are revealed. The model accounts for precedence relationships, limited renewable and non-renewable resources, strict time windows, and the possibility of transferring resources between stations, where transfer times further constrain feasibility. Unlike deterministic formulations, the SRPSAP explicitly incorporates environmental variability such as storms, ice conditions, and operational delays, leading to complex scenarios with different project completion times (Vega-Hidalgo et al., 2026). As a result, the problem arises as a two-stage stochastic mixed-integer program, where first-stage binary selection decisions are coupled with second-stage scenario-dependent scheduling subproblems.

The SRPSAP lies at the intersection of two established research streams: the stochastic project portfolio selection problem (SPPSP) and the stochastic resource-constrained project scheduling problem (SRCPSP) (Habibi et al., 2025). SPPSP studies typically account for uncertainty in project costs, revenues, or success probabilities and focus on maximizing expected portfolio value under budget or risk constraints. However, they usually represent timing and resource effects through surrogate measures such as budgets or due dates, rather than through explicit activity-level schedules (Nascimento et al., 2025). In contrast, the SRCPSP literature emphasizes scheduling precedence-feasible activities under uncertain durations and limited resources. Still, it rarely incorporates first-stage portfolio selection, multi-site station assignments, or inter-station transfer times. To the best of our knowledge, no previous work integrates these two perspectives into a unified two-stage stochastic framework that simultaneously addresses project selection and detailed scheduling across multiple sites under uncertainty.

Solving the SRPSAP is computationally challenging because it combines first-stage binary portfolio decisions with second-stage scenario-dependent scheduling subproblems that are themselves variants of the RCPSP. Each scenario requires constructing precedence-feasible schedules while respecting time windows, resource capacities, and inter-station transfer delays, which significantly increase combinatorial complexity (Bigler et al., 2024). Moreover, the number of

possible scenarios increases with the degree of uncertainty, leading to the well-known “curse of dimensionality” in stochastic programming (Li & Womer, 2015). Standard commercial solvers can handle small instances optimally but scale poorly as the number of projects, stations, and scenarios increases (Dashti et al., 2025). Even decomposition methods, such as integer L-shaped methods, face convergence difficulties when subproblem scheduling is computationally intensive (Fu et al., 2024). These limitations highlight the need for tailored solution strategies that combine exact methods with heuristic and metaheuristic techniques to achieve both solution quality and scalability.

To address the SRPSAP, we propose a two-stage stochastic programming model supported by four complementary solution strategies. First, the stochastic formulation is solved using a state-of-the-art commercial solver to establish optimal benchmarks for small instances. Second, we develop an integer L-shaped decomposition method that exploits the model's separable structure while using branch-and-cut to manage integrality constraints. Third, we design a hybrid approach (IL-S+SA) that combines the decomposition framework with simulated annealing to evaluate recourse subproblems, thereby accelerating convergence in larger instances. Finally, we propose a stand-alone simulated annealing metaheuristic tailored to SRPSAP, which provides high-quality approximate solutions for large-scale cases where exact methods become impractical. Together, these methods represent a balanced strategy that integrates exact optimization with heuristic flexibility to improve both scalability and robustness.

The effectiveness of the proposed methods is demonstrated through computational experiments on 36 synthetic instances that emulate Antarctic logistics with uncertain activity durations. A comparative analysis across four approaches reveals consistent trade-offs between accuracy and scalability. While exact methods provide benchmarks for small-sized problems, hybrid and metaheuristic strategies achieve near-optimal performance within seconds, offering practical advantages for larger cases. These results highlight the main contributions of this study:

(i) The formalization of SRPSAP as a two-stage stochastic optimization model that integrates portfolio selection with multi-site scheduling.
(ii) The development of an integer L-shaped decomposition tailored to this problem.
(iii) The introduction of a hybrid decomposition–metaheuristic approach that balances tractability and quality.
(iv) The construction of a validated testbed that generates managerial insights for the planning of scientific operations in extreme environments. Collectively, these contributions advance the methodological frontier of stochastic project scheduling while providing actionable decision support for Antarctic research management.

Beyond methodological advances, the proposed framework provides direct value for decision-makers responsible for planning Antarctic research programs. By explicitly modeling uncertainty in project durations, resource mobility, and inter-station transfers, the approach enables planners to prioritize projects with the highest expected scientific return while respecting operational feasibility. The framework also supports resource prepositioning and contingency planning, enabling logistics managers to assess trade-offs between scientific value and risk exposure under various scenarios. From a policy perspective, the results offer guidance on allocating limited budgets and transport capacity across competing projects, ensuring that critical research goals are achieved despite severe

environmental variability. More broadly, the methods can be adapted for use in other high-risk environments, such as offshore energy, disaster response, or remote exploration. In this way, the study not only contributes to the optimization literature but also provides actionable insights for managers seeking resilient and scientifically productive operations in extreme settings.

The remainder of this paper is organized as follows: Section 2 reviews related work; Section 3 formalizes the stochastic optimization model; Section 4 presents the solution methods; Section 5 reports the computational experiments; Section 6 discusses the results of the experiments; and Section 7 concludes with a discussion of contributions, managerial implications, and directions for future research.

2. Literature Review

The problem studied in this work is closely related to three established lines of research in project evaluation and scheduling under uncertainty. The stochastic project portfolio selection problem (SPPSP) focuses on choosing which projects to undertake when future outcomes such as costs, benefits, or success probabilities are uncertain, typically aiming to maximize expected value while controlling risk. The stochastic resource-constrained project scheduling problem (SRCPSP) addresses how to construct feasible activity-level schedules when durations or resource availability vary across scenarios, emphasizing precedence feasibility and efficient use of limited renewable resources. A third stream, the project portfolio selection and scheduling problem (PPSSP), integrates these two perspectives by jointly determining which projects to execute and how to schedule their activities. However, it generally assumes single-site settings and limited forms of uncertainty. The SRPSAP studied here extends these formulations by incorporating scenario-dependent activity durations, multi-site station assignments, and transfer times, thereby linking portfolio decisions with detailed schedules in a two-stage stochastic framework suited to Antarctic research operations.

2.1. Stochastic project portfolio scheduling problem.

SPPSP typically address uncertainties in costs, revenues, cash flows, or success probabilities, and often incorporate structural dependencies among projects (Panadero et al., 2020; Ghasemi Bojd & Koosha, 2018). Their objectives frequently combine expected net present value, variance-based risk measures, Value-at-Risk, and Conditional Value-at-Risk within multi-objective formulations that balance profitability, risk, and sustainability (Li et al., 2016; Nielsen et al., 2024). However, these approaches generally abstract away from detailed activity-level scheduling, representing temporal or resource effects through surrogate elements such as budget limits, due dates, or aggregate resource constraints (Panadero et al., 2020; Saiz et al., 2024). In contrast, the SRPSAP framework requires explicit verification of schedule feasibility under scenario-dependent durations while also capturing multi-site resource allocation and transfer-time restrictions—features that extend beyond the modeling scope of traditional SPPSP formulations.

Despite this divergence, there are important points of convergence. First, two-stage stochastic programming structures and risk-sensitive formulations (e.g., CVaR-based) are common across both literatures (Han et al., 2024). Second, simheuristics (metaheuristics enhanced with Monte Carlo

simulation) are widely used for SPPSP (Panadero et al., 2020) and are especially relevant when scenario subproblems are computationally intensive. Third, large-scale SPPSP benchmarks show that exact optimization alone becomes impractical as instance size grows, favoring hybrid exact–metaheuristic approaches; this supports the methodological rationale of SRPSAP (Saiz et al., 2024).

Representative studies include simheuristic variable neighborhood search (VNS) for SPPSP under uncertainty, which shows how deterministic-optimal portfolios can become suboptimal once uncertainty is introduced (Panadero et al., 2020; Saiz et al., 2024), and how correlation structures influence optimal choices. Other contributions suggest a two-stage sustainable SPPSP with fuzzy, robust, and stochastic elements that address project conflicts, splitting, and interruptions (Dong et al., 2025). Large-scale SPPSP benchmarks also highlight the computational trade-offs between exact algorithms and heuristics (Saiz et al., 2024). These results support the use of stochastic programming with decomposition in SRPSAP, complemented by metaheuristics to improve scalability.

2.2. Stochastic resource-constrained project scheduling problem.

SRCPSP studies how to generate precedence-feasible schedules when activity durations and, in some cases, resource availability or transfer times are subject to uncertainty (Chen et al., 2021; Song et al., 2025; Zhang et al., 2025; Chao et al., 2026). Research in this area has developed several complementary strategies. Proactive methods aim to construct robust baseline schedules by incorporating time buffers that mitigate the impact of variability (Ma et al., 2019a). Reactive methods focus on making adjustments during execution in response to deviations from the planned schedule (Xu & Bai, 2024). Hybrid proactive–reactive approaches combine these two perspectives to preserve baseline stability while retaining flexibility to adapt to disruptions (Peng et al., 2023; van den Houten et al., 2025). A wide range of algorithmic techniques used for SRCPSP includes metaheuristics (Gonçalves et al., 2008; Tritschler et al., 2017; Li et al., 2025), approximate dynamic programming (Satic et al., 2024), Markov decision processes with rollout-based policies (Xie et al., 2021; Sadri et al., 2024; Xie et al., 2025), and recent machine learning approaches that design data-driven dispatching rules (Golab et al., 2023).

Compared to SRCPSP, SRPSAP introduces additional complexities, including first-stage project selection, multi-site assignment, site-dependent resource limits, and inter-station transfer times. Therefore, it aligns closely with multi-project (Melchiors et al., 2024; Satic et al., 2024) and multi-site RCPSP variants (Laurent et al., 2017; Bigler et al., 2022). Multi-site RCPSP accounts for geographically spread resources and site-specific capacities, while RCPSP with transfer times includes travel delays during resource redeployment (Zhang et al., 2025). SRPSAP combines both: each selected project must be assigned to a station, and schedules must consider inter-station transfer delays.

Robustness techniques, such as buffer placement (Chen et al., 2025) and time-windowing (Shahabi-Shahmiri et al., 2024), are directly applicable to SRPSAP's scenario-dependent recourse (Ma et al., 2019b). Integrated proactive-reactive methodologies, including critical-chain-based scheduling with online repair, are also adaptable (Peng et al., 2023). These insights suggest that hybridizing exact

decomposition (Laporte & Louveaux, 1993) with metaheuristics for second-stage recourse could yield computationally feasible strategies.

2.3. Stochastic project portfolio selection and scheduling problem.

SRPSAP generalizes the stochastic project portfolio selection and scheduling problem (stochastic PPSSP or SPPSSP) by including multiple Antarctic stations, site-specific resource needs, transfer delays between stations, and uncertain activity durations. It extends the research project selection and scheduling to the multiple Antarctic stations problem (RPSAP), which was previously studied under deterministic assumptions (Vega-Hidalgo et al., 2026). Compared to earlier PPSSP studies, SRPSAP (i) links selection with activity-level scheduling under uncertainty; (ii) adds multi-site logistical and assignment constraints; and (iii) functions in an extreme environment where uncertainty and resource mobility are increased by weather and remoteness.

Existing stochastic PPSSP approaches mainly use expected value optimization, sometimes incorporating risk (Liu et al., 2025) or sustainability criteria (RezaHoseini et al., 2020; Rahimi et al., 2024), and are solved through mixed integer programming (RezaHoseini et al., 2020), two-stage stochastic programming (Rahimi et al., 2024), or multi-objective evolutionary algorithms (Habibi et al., 2025). SRPSAP aligns with this approach in its first-stage formulation but differs in its recourse: second-stage problems are reduced to RCPSP-type schedules with transfer-time and multi-site constraints. This approach supports the use of decomposition techniques (Laporte & Louveaux, 1993; Benders, 1962) and hybrid heuristics (Kirkpatrick et al., 1983).

Prior SPPSSP applications mainly focus on defense (Harrison et al., 2022), research and development (Hesarsorkh et al., 2021), road construction (Habibi et al., 2025), intelligent manufacturing technologies (Liu et al., 2025), or energy portfolios (Tselios et al., 2024). These fields generally involve financial or technological goals, where uncertainty is mainly modeled in economic returns, project dependencies, or technological risks, but rarely in activity-level scheduling. In contrast, SRPSAP undertakes scientific research in Antarctica, a setting that imposes unique constraints due to environmental stewardship, station logistics, and weather-driven variability, resulting in uncertain durations. Unlike defense or manufacturing settings, Antarctic expeditions face extreme weather conditions that directly affect project durations and feasibility. Additionally, resource mobility across remote stations adds another layer of complexity.

Furthermore, environmental concerns are critical, necessitating a balance between scientific value and ecological responsibility, which is uncommon in most PPSP studies. Therefore, SRPSAP represents both a methodological innovation and a pioneering application domain, demonstrating how stochastic portfolio-and-scheduling models can be adapted to sustainability-focused, high-risk environments. To our knowledge, this is the first stochastic framework for selection and scheduling, explicitly designed for Antarctic expeditions, that connects theoretical optimization with the urgent operational challenges of polar science (Vega-Hidalgo et al., 2026).

2.4. Comparative insights and contributions

Table 1 highlights the main differences between SRPSAP and previous work. To the best of our knowledge, SRPSAP contributes to the literature.

- Integrating stochastic PPSP and multi-site RCPSP with transfer times into a unified two-stage framework.
- Representing the first use of such a model in Antarctic scientific operations with time-window constraints (Vega-Hidalgo et al., 2026).
- Advancing methodology through integer L-shaped decomposition combined with branch-and-cut and hybrid recourse solvers (Laporte & Louveaux, 1993; Kirkpatrick et al., 1983).
- Setting up a reproducible test environment aligned with Antarctic logistics.

3. Mathematical model

3.1. Problem description.

The SRPSAP model captures the challenge of allocating scarce heterogeneous resources across multiple Antarctic research stations while accounting for operational constraints and uncertain activity durations represented through multiple scenarios. It integrates elements of stochastic project portfolio selection with resource-constrained scheduling in a geographically distributed setting, where both the choice of projects and the feasibility of their schedules depend on scenario-specific realizations. The problem incorporates several defining features that together reflect the logistical and environmental complexities of Antarctic operations. The main characteristics of SRPSAP are the following:

- *Uncertain durations:* Project timelines can vary across scenarios due to unpredictable factors, such as weather.
- *Resource Allocation:* Distributing fixed and mobile resources among stations to fulfill project needs.
- *Two-stage decisions:* In the initial stage, it is essential to choose projects that can be planned and implemented in each scenario considered during the second stage.
- *Precedence Constraints:* Maintaining logical order in project scheduling according to dependencies.
- *Time Windows:* The chosen projects can only commence within a specified timeframe.
- *Transportation Delays:* Considering the time needed to transfer resources and research project results between stations.
- *Environmental Impact:* Emphasizing projects that support sustainability goals and reduce ecological disruption.

The objective function of the SRPSAP aims to maximize the expected net scientific profit, calculated as the difference between the weighted total costs in each scenario and the total income generated by the selected projects. Costs include resource utilization, transportation, and environmental

mitigation measures. This function aims to achieve the dual goals of enhancing scientific output while minimizing resource expenditures and environmental impact in all scenarios.

Table 1. Comparison between SRPSAP and related literature.

Work / Variant	Uncertainty	MS	TT	First-stage selection	Second-stage scheduling	Risk/robustness focus	Solution method
Simheuristic PPSP (e.g., VNS+MC)	Cash flows/NPV	–	–	✓	–	Risk constraints; correlations	VNS + simulation (Panadero et al., 2020)
Sustainable two-stage PPSP (fuzzy/robust/stochastic)	Costs/ durations; qualitative	–	–	✓	–	Fuzzy-robust-stochastic	Hybrid MCDM + heuristics (Dong et al., 2025)
Large-scale PPSP benchmarks	Multiple (scenarios)	–	–	✓	–	Deterministic or risk-adjusted value	Exact vs. metaheuristic vs. hybrid (Seiz et al., 2024)
Proactive SRCPSP	Durations/ availability	–	–	–	✓	Buffer placement, slack	GA, robust heuristics (Ma et al., 2019a)
Integrated proactive-reactive SRCPSP	Durations	–	–	–	✓	Online repair policies	Critical-chain, MDP rollouts (Peng et al., 2023; van den Houten et al., 2025)
Multi-site RCPSP	Mostly deterministic	✓	✓	–	✓	Makespan	MIP + metaheuristics (Laurent et al., 2017; Bigler et al., 2022)
SRCPSP with transfer times	Stochastic transfer times	–	✓	–	✓	Robustness to travel	Heuristics/MIP (Zang et al., 2025)
RPSAP	Deterministic	✓	✓	✓	✓	Total profit	MIP + metaheuristics (Vega-Hidalgo et al., 2026)
SRPSAP (this study)	**Stochastic durations (scenarios)**	✓	✓	✓	✓	**Expected total profit**	**Integer L-shaped; IL-S+SA; SA**

Abbreviations are defined as follows: MS: Multi-site, TT: Transfer times; PPSP: Project portfolio selection problem; VNS: Variable neighborhood search; MC: Monte Carlo; NPV: Net present value; MCDM: Multi-criteria decision methods; RCPSP: Resource-constrained project scheduling problem; SRCPSP: Stochastic RCPSP; GA: Genetic algorithm; MDP: Markov decision processes; MIP: Mixed integer programming; RPSAP: Research project selection and scheduling in multiple Antarctic stations problem; SRPSAP: Stochastic RPSAP; IL-S: Stochastic problem solved by the integer L-shaped method; IL-S+SA: Stochastic problem solved by the integer L-shaped method using simulated annealing in the second stage subproblems; SA: Simulated annealing.

3.2. Modelling assumptions

To develop a mathematical model, the assumptions of the problem are as follows:

a) The only uncertain parameter is the project duration; all other parameters are deterministic.
b) We consider two dummy projects as the source and sink. These projects have no resource requirements, durations, or incomes. The source precedes all projects with no predecessors, and the sink follows all projects with no successors.
c) All selected projects are carried out without preemption.
d) The selected projects must be completed within the planned timeframe.
e) A monetary benefit represents the scientific impact of each project. Similarly, the environmental impact is defined as the cost of using each unit of a required resource.

3.3. Notation

The study utilizes the following sets, indices, parameters, and decision variables:

Sets:

- $G = (V', E)$ is a directed graph, where:
  - $V = \{1, \dots, |V|\}$ is the set of projects.
  - $V' = V \cup \{0, |V| + 1\}$ consider the dummy projects 0 and $|V| + 1$.
  - E is the set of arcs that represents the precedence relations between projects.
- $\bar{E}$ is the transitive closure of G.
- $L = \{1, \dots, |L|\}$ is the set of stations.
- $R = \{1, \dots, |R|\}$ is the set of types of resources.
- $R_k = \{1, \dots, |R_k|\}$ is the set of units of type $k \in R$ resource.
- $F_k \subseteq R_k$ is the set of fixed units of type $k \in R$ resource.
- $\Pi = \{1, \dots, |\Pi|\}$ is the set of scenarios.

Indices:

- i, j are the indices of projects, $i, j \in V$.
- l, l' are the indices of stations, $l, l' \in L$.
- k is the index of resources, $k \in R$.
- u is the index of units of k-type resource, $u \in R_k$.
- π is the index of scenarios, $\pi \in \Pi$.

Parameters

- T is the horizon time.
- p_i^π is the duration of task $i \in V'$ in scenario $\pi \in \Pi$.
- $\delta_{ll'}$ is the transportation time between stations $l \in L$, and $l' \in L$.
- r_{ik} is the requirement of type $k \in R$ resource by the project $i \in V'$.
- loc_{ku} is the site where the fixed unit $u \in F_k$ of type $k \in R$ resource is located.

- $[\alpha_i, \beta_i]$ are the time windows when the project $i \in V$ can start.
- $M \geq \sum_{i \in V} \sum_{\pi \in \Pi} p_i^\pi + |V| \delta_{max}$ is a big number, where $\delta_{max} = \max\{\delta_{ll'} : l, l' \in L\}$.
- I_i is the income for select the $i \in V$ project.
- c_{ku}^v is the variable cost of using the unit $u \in R_k$ of type $k \in R$ resource.
- c_{ku}^f is the fixed cost of using the unit $u \in R_k$ of type $k \in R$ resource.
- w^π is the probability of scenario $\pi \in \Pi$.

Decision Variables

- x_i indicates if the project $i \in V'$ is selected, where:

$$x_i = \begin{cases} 1, & \text{if the project } i \text{ is selected} \\ 0, & \text{otherwise} \end{cases}$$

- y_{ij}^π indicates if two projects $i, j \in V : (i,j) \notin \bar{E}$ are scheduled consecutively in scenario $\pi \in \Pi$, where:

$$y_{ij}^\pi = \begin{cases} 1, & \text{if the task } i \text{ finish before of the project } j \text{ starts in scenario } \pi \\ 0, & \text{otherwise} \end{cases}$$

- $r_{ik}^{u\pi}$ indicates if the unit $u \in R_k$ of type $k \in R$ resource is assigned to project $i \in V$ in scenario $\pi \in \Pi$, where:

$$r_{ik}^{u\pi} = \begin{cases} 1, & \text{if the unit } u \text{ of type } k \text{ resource is assigned to project } i \text{ in scenario } \pi \\ 0, & \text{otherwise} \end{cases}$$

- $B_i^\pi \in \mathbb{R}_+^0$ is the start time of task $i \in V'$ in scenario $\pi \in \Pi$.
- s_{il}^π indicates if the project $i \in V'$ is scheduled at the station $l \in L$ in scenario $\pi \in \Pi$, where:

$$s_{il}^\pi = \begin{cases} 1, & \text{if the project } i \text{ is scheduled at the station } l \text{ in scenario } \pi \\ 0, & \text{otherwise} \end{cases}$$

3.4. Stochastic Mixed-Integer Model

Using the notation previously defined, equations (1)-(15) propose a stochastic mixed-integer model representing the SRPSAP.

$$\max \sum_{i \in V} I_i x_i - \sum_{\pi \in \Pi} w^\pi \left(\sum_{i \in V} \sum_{k \in R} \sum_{u \in R_k} \left(p_i^\pi c_{ku}^v + c_{ku}^f \right) r_{ik}^{u\pi} \right) \tag{1}$$

Subject to:

$$x_j \leq x_i, \ \forall (i,j) \in E \tag{2}$$

$$B_i^\pi + p_i^\pi + \left(s_{il}^\pi + s_{jl'}^\pi - 1 \right) \delta_{ll'} \leq B_j^\pi + M\left(1 - x_j\right), \ \forall (i,j) \in E; \ l, l' \in L; \ \pi \in \Pi \tag{3}$$

$$r_{ik}^{u\pi} + r_{jk}^{u\pi} \leq y_{ij}^\pi + y_{ji}^\pi + 1, \ \forall (i,j) \notin \bar{E} : i < j; k \in R : r_{ik} r_{jk} > 0; u \in R_k; \ \pi \in \Pi \tag{4}$$

$$B_i^\pi + p_i^\pi + \left(s_{il}^\pi + s_{jl'}^\pi - 1 \right) \delta_{ll'} \leq B_j^\pi + M\left(1 - x_j\right) + M\left(1 - y_{ij}^\pi\right), \quad \forall (i,j) \notin \bar{E} : i \neq j; l, l' \in L; \ \pi \in \Pi \tag{5}$$

$$B_i^\pi \leq (T - p_i^\pi) x_i, \ \forall i \in V \cup \{0\}; \ \pi \in \Pi \tag{6}$$

$$\alpha_i x_i \leq B_i^{\pi} \leq \beta_i x_i, \ \ \forall i \in V \cup \{0\};\ \pi \in \Pi \tag{7}$$

$$\sum_{u \in R_k} r_{ik}^{u\pi} = r_{ik} x_i, \ \ \forall\, i \in V; k \in R : r_{ik} > 0;\ \pi \in \Pi \tag{8}$$

$$r_{ik}^{u\pi} \leq s_{i,loc_{ku}}^{\pi}, \ \ \forall\, i \in V; k \in R; u \in F_k : r_{ik} > 0;\ \pi \in \Pi \tag{9}$$

$$\sum_{l \in L} s_{il}^{\pi} = x_i, \ \ \forall\, i \in V;\ \pi \in \Pi \tag{10}$$

$$x_i \in \{0,1\}, \qquad \forall\, i \in\ V' \tag{11}$$

$$y_{ij}^{\pi} \in \{0,1\}, \qquad \forall\, (i,j) \notin \bar{E};\ \pi \in \Pi \tag{12}$$

$$r_{ik}^{u\pi} \in \{0,1\}, \qquad \forall\, i \in V; k \in R; u \in R_k; \pi \in \Pi \tag{13}$$

$$B_i^{\pi} \geq 0, \qquad \forall\, i \in V'; \pi \in \Pi \tag{14}$$

$$s_{il}^{\pi} \in \{0,1\}, \qquad \forall\, i \in V;\ l \in L; \pi \in \Pi \tag{15}$$

Equation (1) indicates that the goal is to maximize the expected total profit from the selection and to optimize project scheduling, which is given by the difference between the total income from the selected projects and the expected resource cost in each scenario. The constraints in (2) ensure that a project can be selected only if each of its predecessors has been selected. Furthermore, Equation (3) provides that in each scenario, the selected project commences after its predecessors have finished, taking into account any transportation time if both are scheduled at different stations. The constraints in (4) stipulate that if two non-related projects are executed simultaneously in a scenario, the same resource unit cannot be assigned to both projects. Conversely, the constraints in (4) stipulate that when the same resource unit is assigned to two non-related projects in a scenario, this cannot be executed simultaneously. The constraints in (5) ensure that if two non-related projects are selected and scheduled sequentially in a scenario, the successor project begins after the predecessor project has completed, considering any transportation time if both are scheduled at different stations. The constraints in Equation (6) ensure that the end of the selected projects is before the horizon time.

Additionally, Equation (7) indicates that the selected projects can only start within the specified time window. The constraints in (8) require that several units of each resource type be assigned to every selected project in each scenario. Conversely, if a project is not selected, the constraints in (8) ensure that no resource units are allocated to it. The constraints in (9) guarantee that if a unit of a fixed resource at a station is allocated to a selected project, that project will be scheduled at that station. Equation (10) states that a unique station can be assigned only to a select project. Finally, Equations (11)-(15) define the domain of the decision variables.

4. Proposed Algorithms for the SRPSAP

4.1. The integer L-shaped method

Initially, we will present the integer L-shaped method introduced by Laporte & Louveaux (1993). This approach relies on Benders' decomposition (Benders, 1962) combined with a branch-and-cut technique (Padberg & Rinaldi, 1991) to address the non-integrality of decision variables.

The problem posed by Equations (1)-(15) can be expressed as the formula provided by Equation (16).

$$\max\left\{\sum_{i\in V} I_i x_i + Q(\boldsymbol{x}), \text{s.t. } x_j \le x_i, \forall\,(i,j)\in E, x_i \in \{0,1\}, \forall\, i \in V'\right\} \tag{16}$$

The function $Q(\boldsymbol{x})$ in Equation (16), where $\boldsymbol{x} = \left(x_0, \dots, x_{|V|+1}\right)$, called the recourse function, is given by Equation (17).

$$Q(\boldsymbol{x}) = \sum_{\pi\in\Pi} w^{\pi} Q(\boldsymbol{x}, \pi) \tag{17}$$

The recourse function represents the expected cost of the recourse used by the selected projects $\boldsymbol{x}$ in each scenario π, $Q(\boldsymbol{x}, \pi)$, given by Equation (18).

$$Q(\boldsymbol{x}, \pi) = \max\left\{-\sum_{i\in V}\sum_{k\in R}\sum_{u\in R_k}\left(p_i^{\pi} c_{ku}^{v} + c_{ku}^{f}\right) r_{ik}^{u\pi}, \text{s.t. } (3)-(10), (12)-(15)\right\} \tag{18}$$

Please note that the problem in Equation (16) is equivalent to the problem in Equation (19).

$$\begin{cases} \max & \displaystyle\sum_{i\in V} I_i x_i + \theta \\ \text{s. t.} & \theta \le Q(\boldsymbol{x}) \\ & x_j \le x_i, \qquad \forall\,(i,j)\in E \\ & x_i \in \{0,1\}, \qquad \forall\, i \in V' \\ & \theta \in \mathbb{R} \end{cases} \tag{19}$$

Thus, the integer L-shaped method involves solving the *current problem* (CP) recursively, as given by Equation (20).

$$\text{(CP)}\begin{cases} \max & \displaystyle\sum_{i\in V} I_i x_i + \theta \\ \text{s. t.} & x_j \le x_i, \qquad \forall\,(i,j)\in E \\ & \boldsymbol{D}_q \boldsymbol{x} \ge d_q, \qquad \forall\, q = 1, \dots, t \\ & \boldsymbol{E}_q \boldsymbol{x} + \theta \le e_q, \qquad \forall\, q = 1, \dots, v \\ & x_i \in \{0,1\}, \qquad \forall\, i \in V' \\ & \theta \in \mathbb{R} \end{cases} \tag{20}$$

Let be $(\boldsymbol{x}^{\boldsymbol{\gamma}}, \theta^{\gamma})$ an optimal solution of CP, and let be S^{γ} the set of projects selected in the optimal solution, i.e., $i \in S^{\gamma}$ if $x_i^{\gamma} = 1$ and $i \notin S^{\gamma}$ otherwise. The $\boldsymbol{D}_q \boldsymbol{x} \ge d_q$ are called *feasibility cuts* and

have the form given by Equation (21). Feasibility cuts are added whenever given an optimal solution of CP, the subproblem $Q(\boldsymbol{x}^{\gamma}, \pi)$ is not feasible in some scenario $\pi \in \Pi$.

$$\sum_{i \in S^{\gamma}} x_i \leq |S^{\gamma}| - 1 \tag{21}$$

An optimal solution of CP and feasible on each subproblem $Q(\boldsymbol{x}^{\gamma}, \pi)$ is an optimal solution for SRASAP if it satisfies the condition $\theta^{\gamma} \leq Q(\boldsymbol{x}^{\gamma})$ in Equation (19). Otherwise, an *optimality cut* $\boldsymbol{E}_q \boldsymbol{x} + \theta \leq e_q$, having the form given by Equation (22), where $\mathcal{L}$ is an upper bound on the recourse function given by Equation (23), is added.

$$\theta \leq (Q(\boldsymbol{x}^{\gamma}) - \mathcal{L}) \left(\sum_{i \in S^{\gamma}} x_i - \sum_{i \notin S^{\gamma}} x_i \right) - (Q(\boldsymbol{x}^{\gamma}) - \mathcal{L})(|S^{\gamma}| - 1) + \mathcal{L} \tag{22}$$

$$\mathcal{L} \geq \max\{Q(\boldsymbol{x}), \text{s.t. } x_j \leq x_i, \forall\, (i,j) \in E, x_i \in \{0,1\}, \forall\, i \in V'\} \tag{23}$$

If no projects are selected in the first stage, there will be no associated costs for programming these projects or for utilizing resources to execute them, regardless of the scenario. In other words, if $\boldsymbol{x}^{\gamma} = \boldsymbol{0}$, then $Q(\boldsymbol{0}, \pi) = 0$, for each scenario $\pi \in \Pi$. Therefore, $Q(\boldsymbol{0}) = 0$ is the maximum value for the recourse function. Thus, for the optimality cuts, we choose $\mathcal{L} = 0$.

To solve CP, we use a branch-and-cut procedure. Initially, we define an initial node that solves a linear relaxation of the CP, ignoring θ in the computation unless an optimality cut arises (we can initially set $\theta = \infty$). If the optimal solution of CP is non-integer, i.e., there exists $i \in V'$ such that x_i^{γ} is non-integer, then we generate two new nodes, adding the *integrality cuts* $x_i = 0$ and $x_i = 1$, respectively. Otherwise, we add a feasibility cut or an optimality cut, as described previously. The optimality and feasibility cuts added in each node are used in every node explored during the branch-and-cut procedure. A node is fathomed if one of the following cases occurs:

- When CP does not have a feasible solution.
- When $\sum_{i \in V} I_i x_i^{\gamma} + \theta^{\gamma} < \bar{z}$, where $\bar{z}$ is the best objective value of the stochastic problem found at the current node.
- When the optimal solution of CP satisfies the condition $\theta^{\gamma} \leq Q(\boldsymbol{x}^{\gamma})$ from Equation (19).

The integer L-shaped method finishes when all generated nodes are fathomed. Algorithm 1 shows this method.

In Algorithm 1, the values t, r, and γ in Line 1 count the feasibility and optimality cuts, as well as the integrality nodes. The family $BCNodeSet$ defined in Line 2 contains the sets of integrality cuts that represent each node of the branch-and-cut procedure. A set $Node \in BCNodeSet$ contains all integrality cuts that define a node of the branch-and-cut procedure. The sets $OptCutSet$ and $FeaCutSet$ in Line 2 contain the optimality and feasibility cuts, respectively. The values $\bar{z}$ and $\bar{\boldsymbol{x}}$ in Line 1 are the best objective value and the best solution for the stochastic problem, found at the current node.

Algorithm 1. The integer L-shaped method

Input: Parameters of SRPSAP

Set $t \leftarrow 0, r \leftarrow 0, \gamma \leftarrow 0,\ \bar{z} \leftarrow -\infty,\ \bar{\boldsymbol{x}} \leftarrow NULL$
Set $BCNodeSet \leftarrow \{\emptyset\}, OptCutSet \leftarrow \emptyset, FeaCutSet \leftarrow \emptyset$
while $BCNodeSet \neq \emptyset$ **do**:
  $\gamma \leftarrow \gamma + 1$
  Select $Node \in BCNodeSet$ *# Select a node from the branch-and-cut procedure*
  $BCNodeSet \leftarrow BCNodeSet - \{Node\}$
  $(\boldsymbol{x}^\gamma, \theta^\gamma) \leftarrow$ Solve$(CP, Node, OptCutSet, FeaCutSet)$
  while $(\boldsymbol{x}^\gamma, \theta^\gamma)$ is feasible **and** $\sum_{i \in V} I_i x_i^\gamma + \theta^\gamma \geq \bar{z}$ **do**:
    $Integrality \leftarrow$ **True**
    for $i \in V$ **do**:
      if x_i^γ is non-integer **do**:
        $BCNodeSet \leftarrow BCNodeSet \cup \{Node \cup \{x_i = 0\}\} \cup \{Node \cup \{x_i = 1\}\}$
        $Integrality \leftarrow$ **False**
        break
    if $Integrality =$ **True do**:
      $Feasibility \leftarrow$ **True**
      for $\pi \in \Pi$ **do:**
        $Solution^\pi \leftarrow$ Solve$(\mathcal{Q}(\boldsymbol{x}^\gamma, \pi))$
        if $Solution^\pi$ is non-feasible **do:**
          $FeaCutSet \leftarrow FeaCutSet \cup \{\sum_{i \in S^\gamma} x_i \leq |S^\gamma| - 1\}$
          $Feasibility \leftarrow$ **False**
          $t \leftarrow t + 1$
          break *# Add a feasibility cut and solve the current problem*
      if $Feasibility =$ **True do:**
        $\mathcal{Q}(\boldsymbol{x}^\gamma) \leftarrow \sum_{\pi \in \Pi} w^\pi \mathcal{Q}(\boldsymbol{x}^\gamma, \pi)$
        $z^\gamma \leftarrow \sum_{i \in V} I_i x_i^\gamma + \mathcal{Q}(\boldsymbol{x}^\gamma)$
        if $z^\gamma > \bar{z}$ **do:**
          $\bar{z} \leftarrow z^\gamma, \bar{\boldsymbol{x}} \leftarrow \boldsymbol{x}^\gamma$
        if $\theta^\gamma \leq \mathcal{Q}(\boldsymbol{x}^\gamma)$ **do:**
          break *# Fathom this node*
        else:
          $OptCutSet \leftarrow OptCutSet \cup \{\theta \leq (\mathcal{Q}(\boldsymbol{x}^\gamma) - \mathcal{L})(\sum_{i \in S^\gamma} x_i - \sum_{i \notin S^\gamma} x_i) - (\mathcal{Q}(\boldsymbol{x}^\gamma) - \mathcal{L})(|S^\gamma| - 1) + \mathcal{L}\}$
          $r \leftarrow r + 1$
    else:
      break *# Fathom this node*
  $(\boldsymbol{x}^\gamma, \theta^\gamma) \leftarrow$ Solve$(CP, Node, OptCutSet, FeaCutSet)$
Output: $\bar{z},\ \bar{\boldsymbol{x}}$

Note that the optimal solution of CP in the first iteration of the integer L-shaped algorithm is $x_i^\gamma = 1$, for all $i \in V'$. That is, initially, all projects are selected because there are no cuts from the second-stage subproblems. When the set of candidate projects is large relative to the projects that can be scheduled due to resource constraints, the algorithm can generate a large number of feasibility cuts before adding an optimality cut. Furthermore, the complexity of the second-stage subproblems

slows each step in the search for a feasibility or an optimality cut. To address these complexities, we propose the following:

- Generate an initial solution using heuristic methods before Line 3 of Algorithm 1. This solution introduces an optimality cut to the CP, thereby reducing the search space.
- When the algorithm generates two integrality nodes, we prioritize the branch where $x_i = 0$ (Line 5 of Algorithm 1). This decreases the number of selected projects, thereby enhancing the likelihood of achieving a feasible solution in the second-stage subproblems.
- We employ a metaheuristic approach to address the second-stage subproblems (Line 18 of Algorithm 1). This speeds up the computation of the recourse function, but results in non-optimal solutions to the second-stage subproblems.

4.2. The list-scheduling-based algorithm for the stochastic problem

To generate an initial solution for the integer L-shaped method, we introduce the list-scheduled-based algorithm for the stochastic problem (LSBA-SP), as a modification of the algorithm proposed by Vega-Hidalgo et al. (2026). The input of the LSBA-SP considers a list of non-dummy projects $\boldsymbol{\sigma} = (\sigma_1, \dots, \sigma_{|V|})$. The LSBA-SP reviews this list and chooses the projects that can be scheduled in all scenarios. The outputs of this algorithm are the set of selected projects $S(\boldsymbol{\sigma})$, the expected total profit $P(\boldsymbol{\sigma})$, the completion time $f_i^{\pi}(\boldsymbol{\sigma})$, the site assigned $l_i^{\pi}(\boldsymbol{\sigma})$ and the set or resources allocated $ResourceAllocation_i^{\pi}(\boldsymbol{\sigma})$, of each selected project i in each scenario π. Algorithm 2 shows the LSBA-SP.

For each project on the list, where all its predecessors are selected, or the project has no predecessors, the LSBA-SP in Algorithm 2 performs the following:

- Lines 5 to 17 calculate the availability time of each resource unit $a_{ku}^{l\pi}$ at each station and scenario. The availability time of each fixed resource is set to infinity to ensure that this resource is not allocated to a project at a different station.
- Line 18 temporarily allocates to project i the first r_{ik} resources available at each station and scenario.
- Lines 19 to 21 calculate the temporal completion time for each project at each station and scenario, taking into account precedence relations, the availability time of each resource unit assigned to this project, and the earliest starting time.
- Line 22 assigns a station to the project in each scenario.
- In each scenario, if the completion time does not exceed the horizon time and the project starts before the latest time of the time windows, then Lines 23 to 29 add the project to the set of selected projects, calculate the total income, the completion time, and the resource allocation in each scenario, and the total expected cost.
- Line 30 calculates the expected total profit.

Algorithm 2. The list-scheduling-based algorithm for the stochastic problem

Input: $\boldsymbol{\sigma}$, parameters of SRCPSC

Set $S(\boldsymbol{\sigma}) \leftarrow \emptyset$, $Income \leftarrow 0$, $ExpectedCost \leftarrow 0$

for $i \in \boldsymbol{\sigma}$ **such that** $j \in S(\boldsymbol{\sigma})$, $\forall\,(j,i) \in E$ **or** $(0,i) \in E$ **do**:

  for $\pi \in \Pi$ **do:**

    for $l \in L$ **do:**

      for $k \in R$ **such that** $r_{ik} > 0$ **do:**

        for $u \in F_k$ **do:**

          if $loc_{ku} \neq l$ **do:**

            $a_{ku}^{l\pi} \leftarrow \infty$

          else if u is not assigned to any project **do:**

            $a_{ku}^{l\pi} \leftarrow 0$

          else:

            $a_{ku}^{l\pi} \leftarrow f_j^{\pi}(\boldsymbol{\sigma})$, where j is the last project assigned to u.

        for $u \notin F_k$ **do:**

          if u is not assigned to any project **do:**

            $a_{ku}^{l\pi} \leftarrow 0$

          else:

            $a_{ku}^{l\pi} \leftarrow f_j^{\pi}(\boldsymbol{\sigma}) + \delta_{l_j l}$, where j is the last project assigned to u.

      $TemporallyAllocation_{il}^{\pi} \leftarrow \{(k,u) : u$ are the first $r_{ik} > 0$ units ordered by $a_{ku}^{l\pi}\}$

      $A_i^{l\pi} \leftarrow \begin{cases} \max\{d_j + \delta_{l_j l} : (j,i) \in E\}, & \text{if } i \text{ has a predecessor} \\ 0, & \text{otherwise} \end{cases}$

      $B_i^{l\pi} \leftarrow \max\{a_{ku}^{l\pi} : (k,u) \in TemporallyAllocation_{il}^{\pi}\}$

      $f_i^{l\pi} \leftarrow \max\{A_i^{l\pi}, B_i^{l\pi}, \alpha_i\} + p_i^{\pi}$

    $l_i^{\pi}(\boldsymbol{\sigma}) \leftarrow \operatorname{argmin}\{f_i^{l\pi} : l \in L\}$

  if $\max\{\min\{f_i^{l\pi} : l \in L\} : \pi \in \Pi\} \leq T$ **and** $\max\{\min\{f_i^{l\pi} - p_i^{\pi} : l \in L\} : \pi \in \Pi\} \leq \beta_i$ **do:**

    $S(\boldsymbol{\sigma}) \leftarrow S(\boldsymbol{\sigma}) \cup \{i\}$

    $Income \leftarrow Income + I_i$

    for $\pi \in \Pi$ **do:**

      $f_i^{\pi}(\boldsymbol{\sigma}) \leftarrow \min\{f_i^{l\pi} : l \in L\}$

      $ResourceAllocation_i^{\pi}(\boldsymbol{\sigma}) \leftarrow TemporallyAllocation_{il_i^{\pi}(\boldsymbol{\sigma})}^{\pi}$

      $ExpectedCost \leftarrow ExpectedCost + w^{\pi} \sum_{(k,u) \in ResourceAllocation_i^{\pi}(\boldsymbol{\sigma})} (p_i^{\pi} c_{ku}^{v} + c_{ku}^{f})$

$P(\boldsymbol{\sigma}) \leftarrow Income - ExpectedCost$ *# Expected Profit*

Output: $S(\boldsymbol{\sigma})$, $P(\boldsymbol{\sigma})$, $f_i^{\pi}(\boldsymbol{\sigma})$, $l_i^{\pi}(\boldsymbol{\sigma})$, $ResourceAllocation_i^{\pi}(\boldsymbol{\sigma})$

4.3. The list-scheduling-based algorithm and simulated annealing metaheuristic for the recourse function

Given the complexity of the subproblems in the L-shaped method, we propose a simulated annealing metaheuristic (Kirkpatrick et al., 1983) to compute the recourse function (SA-RF). This method mimics the cooling process in pure substances, which guides SA-RF toward equilibrium. Each solution in SA-RF is represented by a list of selected projects $\boldsymbol{\sigma}^{\gamma} = (\sigma_1, \ldots, \sigma_{|S^{\gamma}|})$, where S^{γ} is the set of projects that are selected by the CP. Then, SA-RF uses a list-scheduling-based algorithm to evaluate $\boldsymbol{\sigma}^{\gamma}$ in each scenario $\pi \in \Pi$ (LSBA-RF). The LSBA-RF is an adaptation of the LSBA-SP (Algorithm 2). The main differences arise in:

- LSBA-SP evaluates a list of projects for the entire stochastic problem; LSBA-RF evaluates a list of selected projects in a specific scenario π.
- LSBA-SP selects and schedules projects from a list; LSBA-RF only schedules selected projects if it is possible.
- LSBA-SP calculates the expected total profit for the entire stochastic problem; LSBA-RF calculates the total cost of using allocated resources in a specific scenario π.

When the LSBA-RF cannot schedule all selected projects from the list $\boldsymbol{\sigma}^{\gamma}$, the solution is declared infeasible. Otherwise, the LSBA-RF returns the total cost $Cost(\boldsymbol{\sigma}^{\gamma}, \pi)$, the completion time $f_i(\boldsymbol{\sigma}^{\gamma}, \pi)$, the site assigned $l_i(\boldsymbol{\sigma}^{\gamma}, \pi)$, the resource set allocated for each project $ResourceAllocation_i(\boldsymbol{\sigma}^{\gamma}, \pi)$, and an indicator of feasibility of the solution $Feasibility(\boldsymbol{\sigma}^{\gamma}, \pi)$. Algorithm 3 shows the LSBA-RF.

For each project on the list, if a non-scheduled predecessor exists, the solution is declared infeasible, and the total cost is set to infinity (Lines 32-35 in Algorithm 3). Otherwise, the LSBA-RF performs the following:

- Lines 5 to 17 calculate the availability time of each resource unit a_{ku}^{l} at each station. The availability time of each fixed resource is set to infinity to ensure that this resource is not allocated to a project at a different station.
- Line 18 temporarily allocates to project i the first r_{ik} resources available at each station.
- Lines 19 to 21 calculate the temporal completion time for each project at each station, taking into account precedence relations, the availability time of each resource unit assigned to the project, and the earliest starting time.
- Line 22 assigns a station to the project.
- If the completion time does not exceed the horizon time and the project starts before the latest time of the time windows, then Lines 23 to 27 add the project to the set of scheduled projects, calculate the completion time, the resource allocation, and the total cost. Otherwise, Lines 28-31 declare the solution infeasible and set the total cost to infinity.

Therefore, the LSBA-RF is essential for the operation of the SA-RF, as demonstrated in Algorithm 4.

The SA-RF in Algorithm 4 explores the solution space using a neighborhood system $\mathcal{N}(\boldsymbol{\sigma}^{\gamma})$, which involves swapping two projects in the list $\boldsymbol{\sigma}^{\gamma}$. Therefore, we specify the SA-RF operators as follows. A list of selected projects $\boldsymbol{\sigma}'$ is created from S^{γ} as an initial solution (Line 1). These projects are ordered by the earliest start time α_i, and the list is considered the best solution (Line 2). The initial temperature is randomly generated by creating a set of random lists and calculating the average of the absolute differences between the solution costs $\overline{\Delta} > 0$. Since $Prob_0 = \exp(-\overline{\Delta}/Temp)$, then the initial temperature is calculated by $Temp = -\overline{\Delta}/\ln(Prob_0)$ (Line 3). In each iteration of the algorithm, a random neighbor $\boldsymbol{\sigma}'' \in \mathcal{N}(\boldsymbol{\sigma}')$ of the current solution $\boldsymbol{\sigma}'$ is selected (Line 5). If the cost of the new solution is lower than the current one, it is accepted as the new current solution (Line 8). Otherwise, the new solution is accepted with a probability of $\exp(-\Delta/Temp)$ (Line 10). Next, if the cost of the current solution is lower than that of the best solution, the current solution becomes the new best solution (Line 12). Finally, the Temperature Update operator updates the temperature $Temp$ and the cooling ratio a as follows: the temperature is updated by a geometric schedule

$Temp \leftarrow \varepsilon\, Temp$, where the cooling ratio is $\varepsilon = \varepsilon_{fast}$, if the best solution is not updated after a certain number of iterations, or $\varepsilon = \varepsilon_{slow}$, otherwise (Line 13). The algorithm stops after the minimum temperature $Temp_{min}$ or the execution time limit is reached. The outputs of the SA-RF are the best-found solution $\boldsymbol{\sigma}^{\gamma}$, the total cost $Cost(\boldsymbol{\sigma}^{\gamma}, \pi)$, and the feasibility $Feasibility(\boldsymbol{\sigma}^{\gamma}, \pi)$ of this solution in the scenario π.

Algorithm 3. The list-scheduling-based algorithm for the recourse function

Input: $\boldsymbol{\sigma}^{\gamma}$, parameters of SRCPSC in scenario π

1: Set $S \leftarrow \emptyset$, $Cost(\boldsymbol{\sigma}^{\gamma}, \pi) \leftarrow 0$, $Feasibility(\boldsymbol{\sigma}^{\gamma}, \pi) \leftarrow$ **True**
2: **for** $i \in \boldsymbol{\sigma}^{\gamma}$ **do**:
3: **if** $j \in S$, $\forall\, (j,i) \in E$ **or** $(0,i) \in E$ **do:**
4: **for** $l \in L$ **do:**
5: **for** $k \in R$ **such that** $r_{ik} > 0$ **do:**
6: **for** $u \in F_k$ **do:**
7: **if** $loc_{ku} \neq l$ **do:**
8: $a_{ku}^{l} \leftarrow \infty$
9: **else if** u is not assigned to any project **do:**
10: $a_{ku}^{l} \leftarrow 0$
11: **else:**
12: $a_{ku}^{l} \leftarrow f_j(\boldsymbol{\sigma}^{\gamma}, \pi)$, where j is the last project assigned to u.
13: **for** $u \notin F_k$ **do:**
14: **if** u is not assigned to any project **do:**
15: $a_{ku}^{l} \leftarrow 0$
16: **else:**
17: $a_{ku}^{l} \leftarrow f_j(\boldsymbol{\sigma}^{\gamma}, \pi) + \delta_{l_j l}$, where j is the last project assigned to u.
18: $TemporallyAllocation_{il} \leftarrow \{(k,u) : u$ are the first $r_{ik} > 0$ units ordered by $a_{ku}^{l}\}$
19: $A_i^l \leftarrow \begin{cases} \max\{d_j + \delta_{l_j l} : (j,i) \in E\}, & \text{if } i \text{ has a predecessor} \\ 0, & \text{otherwise} \end{cases}$
20: $B_i^l \leftarrow \max\{a_{ku}^{l} : (k,u) \in TemporallyAllocation_{il}\}$
21: $f_i^l \leftarrow \max\{A_i^l, B_i^l, \alpha_i\} + p_i^{\pi}$
22: $l_i(\boldsymbol{\sigma}^{\gamma}, \pi) \leftarrow \operatorname{argmin}\{f_i^l : l \in L\}$
23: **if** $\min\{f_i^l : l \in L\} \leq T$ **and** $\min\{f_i^l - p_i^{\pi} : l \in L\} \leq \beta_i$ **do:**
24: $S \leftarrow S \cup \{i\}$
25: $f_i(\boldsymbol{\sigma}^{\gamma}, \pi) \leftarrow \min\{f_i^l : l \in L\}$
26: $ResourceAllocation_i(\boldsymbol{\sigma}^{\gamma}, \pi) \leftarrow TemporallyAllocation_{il_i(\boldsymbol{\sigma}^{\gamma}, \pi)}$
27: $Cost(\boldsymbol{\sigma}^{\gamma}, \pi) \leftarrow Cost(\boldsymbol{\sigma}^{\gamma}, \pi) + \sum_{(k,u) \in ResourceAllocation_i(\boldsymbol{\sigma}^{\gamma}, \pi)} \left(p_i^{\pi} c_{ku}^{v} + c_{ku}^{f}\right)$
28: **else:**
29: $Cost(\boldsymbol{\sigma}^{\gamma}, \pi) \leftarrow \infty$
30: $Feasibility(\boldsymbol{\sigma}^{\gamma}, \pi) \leftarrow$ **False**
31: **break**
32: **else:**
33: $Cost(\boldsymbol{\sigma}^{\gamma}, \pi) \leftarrow \infty$
34: $Feasibility(\boldsymbol{\sigma}^{\gamma}, \pi) \leftarrow$ **False**
35: **break**

Output: $Cost(\boldsymbol{\sigma}^{\gamma}, \pi)$, $Feasibility(\boldsymbol{\sigma}^{\gamma}, \pi)$, $f_i(\boldsymbol{\sigma}^{\gamma}, \pi)$, $l_i(\boldsymbol{\sigma}^{\gamma}, \pi)$, $ResourceAllocation_i(\boldsymbol{\sigma}^{\gamma}, \pi)$

Algorithm 4. The simulated annealing metaheuristic for the recourse function

Input: parameters of the SRPSAP in scenario π, parameters of SA, S^{γ}
1: Generate an initial solution $\boldsymbol{\sigma}'$ from S^{γ}
2: $\boldsymbol{\sigma}^{\gamma} \leftarrow \boldsymbol{\sigma}'$
3: Generate initial temperature $Temp$ such that the probability of acceptance is near to $Prob_0$
4: **while** the stopping criterion is not satisfied, **do**:
5: $\quad \boldsymbol{\sigma}'' \leftarrow \boldsymbol{\sigma} \in \mathcal{N}(\boldsymbol{\sigma}')$ *# selected randomly*
6: $\quad \Delta \leftarrow Cost(\boldsymbol{\sigma}'', \pi) - Cost(\boldsymbol{\sigma}', \pi)$ *# $Cost$ is calculated by LSBA-RF*
7: $\quad$ **if** $\Delta < 0$ **then:**
8: $\quad\quad \boldsymbol{\sigma}' \leftarrow \boldsymbol{\sigma}''$
9: $\quad$ **else if** $random(0,1) < \exp(-\Delta/Temp)$ **then:**
10: $\quad\quad \boldsymbol{\sigma}' \leftarrow \boldsymbol{\sigma}''$
11: $\quad$ **if** $Cost(\boldsymbol{\sigma}', \pi) < Cost(\boldsymbol{\sigma}^{\gamma}, \pi)$ **then:**
12: $\quad\quad \boldsymbol{\sigma}^{\gamma} \leftarrow \boldsymbol{\sigma}'$
13: $\quad Temp,\ \varepsilon \leftarrow$ Temperature Update $(Temp, \boldsymbol{\sigma}^{\gamma}, \boldsymbol{\sigma}', \varepsilon, \pi)$
Output: $\boldsymbol{\sigma}^{\gamma}$, $Cost(\boldsymbol{\sigma}^{\gamma}, \pi)$, $Feasibility(\boldsymbol{\sigma}^{\gamma}, \pi)$ *# $Feasibility$ is calculated by LSBA-RF*

4.4. The simulated annealing metaheuristic for the stochastic problem

As an alternative to the integer L-shaped method, we propose a simulated annealing metaheuristic to solve the SRPSAP (SA-SP). The solution in SA-SP is represented by a list of non-dummy projects, denoted by $\boldsymbol{\sigma} = (\sigma_1, \dots, \sigma_{|V|})$, and evaluated by the LSBA-SP, as shown in Section 4.2. Therefore, the SA-SP searches for a near-optimal list using a neighborhood system $\mathcal{N}(\boldsymbol{\sigma})$, which involves swapping two projects; meanwhile, the LSBA-SP selects projects from the list and schedules them in each scenario. Algorithm 5 illustrates the SA-SP procedure.

Algorithm 5. The simulated annealing metaheuristic for the stochastic problem

Input: parameters of the SRPSAP, parameters of SA
1: Generate initial solution $\boldsymbol{\sigma}'$
2: $\boldsymbol{\sigma}^{*} \leftarrow \boldsymbol{\sigma}'$
3: Generate initial temperature $Temp$ such that the probability of acceptance is near to $Prob_0$
4: **while** the stopping criterion is not satisfied, **do**:
5: $\quad \boldsymbol{\sigma}'' \leftarrow \boldsymbol{\sigma} \in \mathcal{N}(\boldsymbol{\sigma}')$ *# selected randomly*
6: $\quad \Delta \leftarrow P(\boldsymbol{\sigma}'') - P(\boldsymbol{\sigma}')$ *# P calculated by LSBA-SP*
7: $\quad$ **if** $\Delta > 0$ **then:**
8: $\quad\quad \boldsymbol{\sigma}' \leftarrow \boldsymbol{\sigma}''$
9: $\quad$ **else if** $random(0,1) < \exp(\Delta/Temp)$ **then:**
10: $\quad\quad \boldsymbol{\sigma}' \leftarrow \boldsymbol{\sigma}''$
11: $\quad$ **if** $P(\boldsymbol{\sigma}') > P(\boldsymbol{\sigma}^{*})$ **then:**
12: $\quad\quad \boldsymbol{\sigma}^{*} \leftarrow \boldsymbol{\sigma}'$
13: $\quad Temp,\ \varepsilon \leftarrow$ Temperature Update $(Temp, \boldsymbol{\sigma}^{*}, \boldsymbol{\sigma}', \varepsilon)$
Output: $\boldsymbol{\sigma}^{*}$, $P(\boldsymbol{\sigma}^{*})$

We specify the SA-SP operators presented in Algorithm 5 as follows. A list of projects $\boldsymbol{\sigma}'$ is created as an initial solution, ordered by the earliest start time α_i and considered the best solution (Lines 1-2). The initial temperature is randomly determined by generating several random lists and computing

the average of the absolute differences between their solution profits $\overline{\Delta} > 0$ (Line 3). Since $Prob_0 = \exp(-\overline{\Delta}/Temp)$, then the initial temperature is calculated by $Temp = -\overline{\Delta}/\ln(Prob_0)$. In each iteration of the algorithm a random neighbor $\boldsymbol{\sigma}'' \in \mathcal{N}(\boldsymbol{\sigma}')$ of the current solution $\boldsymbol{\sigma}'$ is selected (Line 5). If the profit from the new solution exceeds that of the current one, it is accepted as the new current solution (Line 8). Otherwise, the new solution is accepted with a probability of $\exp(\Delta/Temp)$ (Line 10). Next, if the current solution's profit exceeds that of the best solution, it becomes the new best solution (Line 12). Finally, the Temperature Update operator updates the temperature $Temp$ and the cooling ratio a as follows: the temperature is updated by a geometric schedule $Temp \leftarrow \varepsilon\, Temp$, where the cooling ratio is $\varepsilon = \varepsilon_{fast}$, if the best solution is not updated after a certain number of iterations, or $\varepsilon = \varepsilon_{slow}$, otherwise (Line 13). The algorithm stops after the minimum temperature $Temp_{min}$ or the execution time limit is reached. The outputs of the SA-SP are the best-founded solution $\boldsymbol{\sigma}^*$, and the profit of the selection and scheduling $P(\boldsymbol{\sigma}^*)$.

5. Computational experiments and results

This section shows the results from testing the stochastic model in Section 3 and the algorithms in Section 4. Specifically, we compare four methods to solve the SRPSAP:

- *CS*: Solve the stochastic model given by Equations (1)-(15) using a commercial solver.
- *IL-S*: Solve the stochastic model using the integer L-shaped method presented in Algorithm 1, generate an initial feasible solution with the LBSA-SP shown in Algorithm 2, and use a commercial solver to tackle the CP and subproblems in each scenario.
- *IL-S+SA*: Solve the stochastic model using the integer L-shaped method presented in Algorithm 1, generate an initial feasible solution with the LBSA-SP shown in Algorithm 2, employ a commercial solver to address the CP, and utilize SA-RF presented in Algorithm 4 to solve subproblems in each scenario.
- *SA*: Solve the SRPSAP using the SA-SP metaheuristic shown in Algorithm 5.

All methods are implemented using Python 3.9.13, while the stochastic model, CP, and subproblems are solved with Gurobi 12.0.1. The process runs on a 64-bit Windows 10 Pro system, equipped with an Intel Core i7-4790S processor at 3.2 GHz and 8 GB of RAM. For each instance, we run Gurobi until the relative gap between the upper bound (UB) and lower bound (LB) is less than 10^{-4} (default value provided by Gurobi), or a time limit of 3600 seconds (to solve the stochastic model or CP) or 600 seconds (to solve each subproblem) is reached. The integer L-shaped method runs until all branch-and-cut nodes are fathomed or a time limit of 3600 seconds is reached. The SA-SP and SA-RF metaheuristics run until the temperature reaches 10^{-5} (Vega-Hidalgo et al., 2026). In the subproblems, if Gurobi or SA-RF cannot find a feasible solution by the time the stopping criteria are met, the first-stage solution is declared infeasible. The initial temperature is calculated using three random lists, so the initial probability of accepting the worst solution is approximately 95% (Vega-Hidalgo et al., 2026). We use a fast-cooling ratio $\varepsilon_{fast} = 0.95$, where the best solution is not improved after 10 iterations, and a slow-cooling ratio $\varepsilon_{slow} = 0.99$, otherwise (Vega-Hidalgo et al., 2026). We run SA-SP 20 times per instance (Laurent et al., 2017; Vega-Hidalgo et al., 2026) to ensure

the reliability and consistency of the results, thereby accounting for variability and improving the accuracy of our findings.

5.1 Test instances

To test the proposed methods on this new problem, we need to create instances. Thus, we adapt 36 test instances created by Vega-Hidalgo et al. (2025). These instances consider $|V| \in \{10, 20, 30\}$ projects, $|L| \in \{2,\ 3\}$ sites, $|R| \in \{20, 30\}$ resource types and horizon time $T \in \{30, 60, 120\}$ periods. Each instance is denoted by V$|V|$-L$|L|$-R$|R|$-TT, for example, the instance with 10 projects, 2 sites, 30 resource types and a horizon time of 60 periods is denoted by V10-L2-R30-T60. Additionally, we abbreviate each test instance as $\mathcal{I}_m$, with $m \in \{1, \ldots, 36\}$, according to Table 2. In this paper, we examine three scenarios in each instance. Let $U_{\mathbb{Z}}(a, b)$ represents the uniform distribution of integer numbers on the interval (a, b). Thus, the instances from Vega-Hidalgo et al. (2026) are adapted for the stochastic problem as follows:

- *Uncertain durations for each scenario*. We consider the durations in the original instances for the most probable scenario $\pi = 2$. For the other scenarios, let $e \sim U_{\mathbb{Z}}(0, 1)$ and $p \sim U_{\mathbb{Z}}(0, 2)$. Then, $p_i^{\pi} = p_i^2 + (-1)^e p$, for $\pi \neq 2$ (Rahimi et al., 2024).
- *Time windows.* Initially, time windows are set as $\alpha_i = 0$, and $\beta_i = T - p_i^{max}$, where $p_i^{max} = \max\{p_i^{\pi} : \pi \in \Pi\}$, for all projects i. Then, for each project i, we randomly select $\alpha_i \sim U_{\mathbb{Z}}\left(0, \beta_i^{min}\right)$ and $\beta_i \sim U_{\mathbb{Z}}(\alpha_i^{max} + 1, T - p_i^{max})$, where $\alpha_i^{max} = \max\left\{\{\alpha_j : (j, i) \in E\} \cup \{\alpha_i\}\right\}$, and $\beta_i^{min} = \min\left\{\{\beta_j : (i, j) \in E\} \cup \{T - p_i^{max}\}\right\}$. This guarantees that the time windows are consistent with the precedence relations.
- *Probability of each scenario*. We randomly select three values $w^{\pi} \in (0,1)$ such that $\sum_{\pi \in \Pi} w^{\pi} = 1$ and $w^2 \geq w^1, w^3$.

Table 2. List of notations for each instance.

$\mathcal{I}_m$	Instance	$\mathcal{I}_m$	Instance	$\mathcal{I}_m$	Instance	$\mathcal{I}_m$	Instance
$\mathcal{I}_1$	V10-L2-R10-T30	$\mathcal{I}_{10}$	V10-L3-R20-T30	$\mathcal{I}_{19}$	V20-L3-R10-T30	$\mathcal{I}_{28}$	V30-L2-R20-T30
$\mathcal{I}_2$	V10-L2-R10-T60	$\mathcal{I}_{11}$	V10-L3-R20-T60	$\mathcal{I}_{20}$	V20-L3-R10-T60	$\mathcal{I}_{29}$	V30-L2-R20-T60
$\mathcal{I}_3$	V10-L2-R10-T120	$\mathcal{I}_{12}$	V10-L3-R20-T120	$\mathcal{I}_{21}$	V20-L3-R10-T120	$\mathcal{I}_{30}$	V30-L2-R20-T120
$\mathcal{I}_4$	V10-L2-R20-T30	$\mathcal{I}_{13}$	V20-L2-R10-T30	$\mathcal{I}_{22}$	V20-L3-R20-T30	$\mathcal{I}_{31}$	V30-L3-R10-T30
$\mathcal{I}_5$	V10-L2-R20-T60	$\mathcal{I}_{14}$	V20-L2-R10-T60	$\mathcal{I}_{23}$	V20-L3-R20-T60	$\mathcal{I}_{32}$	V30-L3-R10-T60
$\mathcal{I}_6$	V10-L2-R20-T120	$\mathcal{I}_{15}$	V20-L2-R10-T120	$\mathcal{I}_{24}$	V20-L3-R20-T120	$\mathcal{I}_{33}$	V30-L3-R10-T120
$\mathcal{I}_7$	V10-L3-R10-T30	$\mathcal{I}_{16}$	V20-L2-R20-T30	$\mathcal{I}_{25}$	V30-L2-R10-T30	$\mathcal{I}_{34}$	V30-L3-R20-T30
$\mathcal{I}_8$	V10-L3-R10-T60	$\mathcal{I}_{17}$	V20-L2-R20-T60	$\mathcal{I}_{26}$	V30-L2-R10-T60	$\mathcal{I}_{35}$	V30-L3-R20-T60
$\mathcal{I}_9$	V10-L3-R10-T120	$\mathcal{I}_{18}$	V20-L2-R20-T120	$\mathcal{I}_{27}$	V30-L2-R10-T120	$\mathcal{I}_{36}$	V30-L3-R20-T120

5.2 Numerical results

To compare the methods' performance, we use a variety of indicators. The relative gap is defined as the difference between the upper bound of the optimal solution and the objective function value obtained by a method, as given by Equation (24).

$$GAP_{ins}^{met} = \frac{UB_{ins} - OF_{ins}^{met}}{OF_{ins}^{met}}, \forall\, met \in \{CS, IL-S, IL-S+SA\} \cup \{SA_{rep}\}_{rep=1}^{20}, ins = 1, \dots, 36 \quad (24)$$

In Equation (24), GAP_{ins}^{met} indicates the relative gap of the instance ins solved by method met. Here, methods include the CS, the integer L-shaped method (IL-S or IL-S+SA), or a repetition rep of the simulated annealing metaheuristic (SA_{rep}). OF_{ins}^{met} denotes the objective function value, and UB_{ins} is the best upper bound provided by commercial software when solving the stochastic programming model.

Table 3 summarizes the results of experiments conducted for CS, IL-S, IL-S+SA, and SA in both the best and worst repetitions. Each row displays the performance indicators for each method. The first column indicates the method. The next four columns show the minimum, average, standard deviation, and maximum relative gap for each method, respectively. The following column shows the number of instances in which each method reaches the optimal solution (NOS), and the next displays the number of instances in which each method achieves the best relative gap (NBG). The last four rows present the minimum, average, standard deviation, and maximum execution time for each method, respectively. Additionally, Figure 1 compares the relative gaps between the methods in each instance.

Table 3. Summary of each method run.

Method	Relative Gap				NOS	NBG	Execution Time (s)			
	Min.	Ave.	SD	Max.			Min.	Ave.	SD	Max.
CS	2.52e-15	0.8001	1.1752	4.8759	**7**	11	**0.700**	2947.656	1359.666	3602.643
IL-S	**2.15e-15**	0.6095	0.7497	2.7135	**7**	**13**	6.908	3370.179	1845.447	6794.241
IL-S+SA	0.0307	0.6096	0.6606	2.5253	0	1	1.088	2565.932	1470.564	3687.489
SA-B	0.0238	**0.4999**	**0.6037**	**2.2139**	0	12	9.549	**35.663**	21.615	99.688
SA-W	0.0307	0.7285	0.7739	2.9473	0	0	9.176	36.346	**20.732**	**87.312**

Abbreviations are defined as follows: CS: Stochastic problem solved by the commercial solver; IL-S: Stochastic problem solved by the integer L-shaped method; IL-S+SA: Stochastic problem solved by the integer L-shaped method using simulated annealing in the second stage subproblems; SA-B: Stochastic problem solved by simulated annealing (best repetition); SA-B: Stochastic problem solved by simulated annealing (worst repetition); Min.: Minimum value; Ave.: Average value; SD: Standard deviation; Max.: Maximum value; NOS: Number of instances when the method obtains an optimal solution; NBG: Number of instances when the method obtains the best relative gap (this number includes draws with other methods).

In terms of the relative gap, IL-S outperforms other methods. Indeed, Table 3 shows that in 13 of 36 test instances, IL-S has a better relative gap than the other methods. The minimum, average, and maximum relative gap of IL-S is better than CS and IL-S+SA (2.15e-15, 0.6095, and 2.7135, respectively). In addition, CS and IL-S obtain an optimal solution in 7 instances (with a relative gap less than 10^{-4}). Nevertheless, in the best repetition of SA, the average and maximum relative gaps (0.4999 and 2.2139) are better than those of the other methods. Furthermore, SA outperforms other methods in execution time. In the best SA repetition, the maximum execution time is 99.688 seconds, which is 36.139 times lower than the second-best time among other methods (CS with 3602.643 seconds). Moreover, in the best repetition, SA achieves the best relative gap in 12 instances (one fewer than IL-S). These results suggest that SA outperforms the other methods in terms of the

relative gap and execution time. However, in the worst case, SA only outperforms the average, and the maximum relative gap of CS is still larger.

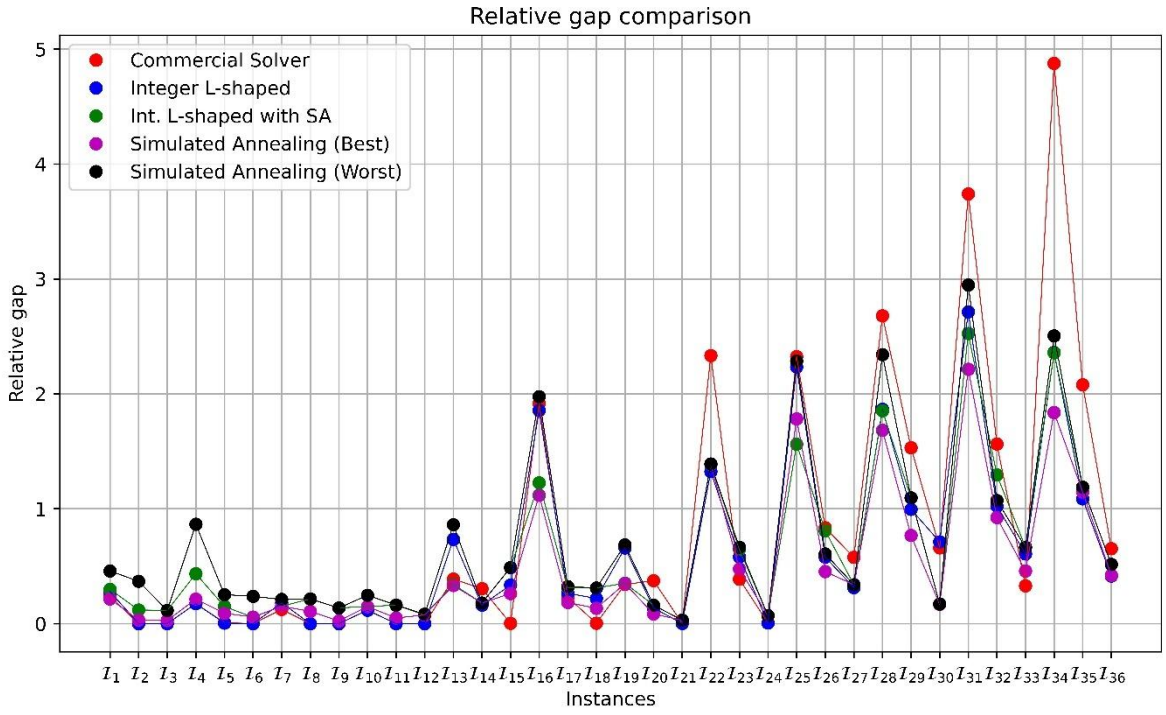

Figure 1. Relative gap of each method

Table 4 summarizes the average relative gap and average execution time for each method, grouped by control parameters. Thus, the first column displays the control parameters. The following five columns show the average relative gaps achieved by CS, IL-S, IL-S+SA, and SA in both the best and worst repetitions, respectively. The last five columns show the average execution time for each method. The first three rows categorize instances by the number of projects. The following two rows classify instances by the number of stations and resource types, respectively, and the last three rows group instances by time horizon.

Table 4 shows that CS achieves the best average relative gap only in instances with 10 projects. The average relative gap of CS is 0.0555, which is 1.1351 times smaller than the second-best average relative gap (IL-S) and 5.0342 times smaller than the worst average relative gap (SA-W). However, in the 20- and 30-project instances, the best SA repetition achieves the smallest average relative gap (0.3839 and 1.0162, respectively). This result is 1.1948 and 1.1789 times smaller than the second-best average relative gap (IL-S+SA), respectively. In these groups, IL-S and IL-S+SA have a better average relative gap than CS. The average relative gap does not exceed 0.5233 (for 20-project instances) or 1.2420 (for 30-project instances). SA obtains a near-optimal solution in less than 60 seconds, with the best and worst repetitions in each instance group. For the SA's best repetition, the average execution time in each group is 34.450, 98.825, and 66.547 times smaller than the second-best average execution time, respectively (IL-S+SA in the first two groups and CS in the last one).

Table 4. Results of each method when the control parameters group the instances.

CP	Average Relative Gap					Average Execution Time (s)				
	CS	IL-S	IL-S+SA	SA-B	SA-W	CS	IL-S	IL-S+SA	SA-B	SA-W
V10	**0.0555**	0.0630	0.1720	0.0996	0.2794	1642.111	1735.228	609.210	**17.684**	19.496
V20	0.5241	0.5233	0.4587	**0.3839**	0.5950	3600.374	4184.972	3478.945	**35.203**	36.378
V30	1.8206	1.2420	1.1980	**1.0162**	1.3111	3600.485	4190.338	3609.642	54.104	**53.164**
L2	0.6598	0.5961	0.5479	**0.4467**	0.7376	3028.216	3621.086	2670.368	**29.318**	30.976
L3	0.9403	0.6228	0.6712	**0.5531**	0.7194	2867.097	3119.273	2461.497	42.008	**41.716**
R10	0.6194	0.5533	0.5436	**0.4426**	0.6571	2892.901	3356.534	2405.891	24.343	**23.754**
R20	0.9807	0.6656	0.6755	**0.5573**	0.7999	3002.412	3383.825	2725.973	**46.984**	48.938
T30	1.6045	1.2082	1.0531	**0.9527**	1.3976	3600.456	4084.694	2683.231	35.557	**34.243**
T60	0.6094	0.4026	0.5278	**0.3746**	0.5242	2840.160	3473.220	2618.278	**36.427**	36.907
T120	0.1862	0.2177	0.2478	**0.1723**	0.2637	2402.354	2552.623	2396.288	**35.007**	37.888

Abbreviations are defined as follows: CP: Control parameter; CS: Stochastic problem solved by the commercial solver; IL-S: Stochastic problem solved by the integer L-shaped method; IL-S+SA: Stochastic problem solved by the integer L-shaped method using simulated annealing in the second stage subproblems; SA-B: Stochastic problem solved by simulated annealing (best repetition); SA-B: Stochastic problem solved by simulated annealing (worst repetition).

The best SA repetition achieves the best average relative gap when instances are split by station groups. Indeed, the average relative gap in the instances with two and three stations is 0.4467 and 0.5531, respectively. IL-S+SA achieves the second-best average relative gap in the two-station instances (0.5479) and IL-S in the three-station instances (0.6228). The worst average relative gap is achieved by the worst repetition of SA in two-station instances (0.7376) and by the worst repetition of CS in three-station instances (0.9403). Near-optimal solutions are achieved by SA in under 45 seconds on average for each instance group. Indeed, the best SA repetition finds a solution in 29.318 seconds and 42.008 seconds for instances with two and three stations, respectively. Compared to the second-best execution time (IL-S+SA), these times are 91.083 and 58.596 times longer than the best SA repetition.

When resource types group test instances, SA's best repetition produces the lowest average relative gap in instances with 10 and 20 resource types. The average relative gap in these groups is 0.4426 and 0.5573, respectively. In the 10-resource type instances, the second-best average relative gap is 0.5436 (IL-S+SA), and in the 20-resource type instances, it is 0.6656 (IL-S). The worst repetition of SA results in the highest average relative gap in instances with 10 resource types (0.6571) and the CS in instances with 20 resource types (0.9807). On average, SA requires less than 50 seconds to find near-optimal solutions in both instance groups. The best SA repetition finds a near-optimal solution in 24.343 seconds for 10 resource types and 46.984 seconds for 20 resource types. These times are 98.833 and 58.019 times less than the second-best average execution time achieved by IL-S+SA.

SA's best repetition has the lowest average relative gap across all groups and horizons. Indeed, the average relative gaps in these groups are 0.9527, 0.3746, and 0.1723, respectively, for horizons of 30, 60, and 120 periods. The second-best average relative gap for IL-S+SA in the 30-period instances is 1.0531. In the 60-period instances, IL-S achieves a second-best average relative gap of 0.4026, and

in the 120-period instances, CS attains a second-best average relative gap of 0.1862. The CS has the highest average relative gap in instances with 30 and 60 periods (1.6045 and 0.6094, respectively), and the SA experiences its worst repetition in instances with 120 periods (0.2637). On average, SA can obtain near-optimal solutions in less than 40 seconds for all groups. The SA's best repetitions take 35.557, 36.427, and 35.007 seconds in each group, respectively. These average execution times are 75.463, 71.887, and 68.452 times shorter than the second-best average times achieved by IL-S+SA.

The results in Table 4 suggest that SA exceeds other methods in terms of the relative gap and execution time. However, Table 4 shows results for only two repetitions per instance (the best and worst). Moreover, in nearly all groups of instances, the SA's worst repetition results in the worst or second-worst average relative gap, meaning not all repetitions of SA outperform the average relative gap of other methods. Table 5 compares SA's performance with other methods in terms of the number and percentage of repetitions in which SA outperforms the others. The first column displays the control parameters. The following three columns show the average number of repetitions per instance when SA has a better relative gap than CS, IL-S, and IL-S+SA, respectively. Likewise, the last three columns present the average percentage of repetitions per instance when SA outperforms other methods in terms of relative gap. Similar to Table 4, Table 5 groups instances by control parameters. The last row in Table 5 shows the results for all instances.

Table 5. Performance comparison when a simulated annealing repetition has a better relative gap than other methods in each instance, across control-parameter groups.

CP	Average number of repetitions			Av. percentage of repetitions (%)		
	vs. CS	vs. IL-S	vs. IL-S+SA	vs. CS	vs. IL-S	vs. IL-S+SA
V10	0.1667	0.4167	8.6667	**0.8333**	2.0833	43.3333
V20	7.3333	5.5000	9.1667	36.6667	**27.5000**	45.8333
V30	18.3333	5.6667	12.0000	91.6667	**28.3333**	60.0000
L2	9.4444	4.8333	8.1667	47.2222	**24.1667**	40.8333
L3	7.7778	2.8889	11.7222	38.8889	**14.4444**	58.6111
R10	8.2778	4.2222	11.0000	41.3889	**21.1111**	55.0000
R20	8.9444	3.5000	8.8889	44.7222	**17.5000**	44.4444
T30	10.6667	7.0833	2.7500	53.3333	35.4167	**13.7500**
T60	10.1667	1.6667	13.6667	50.8333	**8.3333**	68.3333
T120	5.0000	2.8333	13.4167	25.0000	**14.1667**	67.0833
All	**8.6111**	**3.8611**	**9.9444**	**43.0556**	**19.3056**	**49.7222**

Abbreviations are defined as follows: CP: Control parameter; CS: Stochastic problem solved by the commercial solver; IL-S: Stochastic problem solved by the integer L-shaped method; IL-S+SA: Stochastic problem solved by the integer L-shaped method using simulated annealing in the second-stage subproblems.

SA outperforms other methods in fewer than 50% of repetitions per instance. Table 5 shows that SA has a better relative gap than CS in 43.0556% of repetitions (between 8 and 9 repetitions out of 20), than IL-S in 19.3056% of repetitions (between 3 and 4 repetitions), and than IL-S+SA in 49.7222% of

repetitions (between 9 and 10 repetitions). Regarding the relative gap, this result indicates a low likelihood that SA will outperform IL-S in a repeat.

CS and IL-S outperform SA in nearly all repetitions per instance with 10 projects. According to Table 5, only 0.8333% and 2.0833% of SA repetitions outperform CS and IL-S, respectively, in these instances. Additionally, 43.3333% of SA repetitions (between 8 and 9 repetitions) outperform IL-S+SA in 10-project instances. In instances with 20 projects, 27.5000% of SA repetitions surpass the relative gap of IL-S (between 5 and 6 repetitions). Furthermore, 36.6667% (between 7 and 8 repetitions) and 45.8333% (between 9 and 10 repetitions) of SA repetitions exceed the gaps of CS and IL-S+SA, respectively. The performance of SA notably improves compared to CS in instances with 30 projects, with 91.6667% (across 18-19 repetitions) of SA repetitions having a smaller relative gap than CS. However, SA performs similarly to IL-S in instances with 20 and 30 projects. Indeed, in 28.3333% of SA repetitions per 30-project instance (between 5 and 6 repetitions), the relative gap surpasses the IL-S relative gap. Additionally, 60.0000% of SA repetitions per 30-project instance exceed the IL-S+SA relative gap (12 repetitions).

SA's performance decreases compared to CS and IL-S but improves relative to IL-S+SA as the number of stations grows. Table 5 shows that the percentage of SA repetitions when the gap is better than CS drops from 47.2222% (between 9 and 10 repetitions) in 2-station instances to 38.8889% (between 7 and 8 repetitions) in 3-station instances. In the case of IL-S, the percentage decreases from 24.1667% (between 4 and 5 repetitions) in 2-station instances to 14.4444% (between 2 and 3 repetitions) in 3-station instances. Nevertheless, in the case of IL-S+SA, this percentage increases from 40.8333% (between 8 and 9 repetitions) to 58.6111% (between 11 and 12 repetitions) as the number of stations increases from 2 to 3.

As the number of resource types increases, SA's performance declines compared to IL-S and IL-S+SA but increases for CS. Table 5 indicates that the percentage of SA repetitions when the gap is less than IL-S decreases from 21.1111% (between 4 and 5 repetitions) in 10-resource-type instances to 17.5000% (between 3 and 4 repetitions) in 20-resource-type instances. In the IL-S+SA case, the percentage drops from 55.0000% (11 repetitions) in 10 resource-type instances to 44.4444% (between 8 and 9 repetitions) in 20 resource-type instances. However, in the CS case, this percentage increases from 41.3889% to 44.7222% when the number of resource types increases from 10 to 20. This growth is insignificant because the number of repetitions when the SA gap is larger than the CS gap remains around 8-9 per instance.

SA outperforms IL-S in a small percentage of 60-period instances. Table 5 indicates that SA has a better relative gap than IL-S in only 8.3333% of repetitions per 60-period instance (between 1 and 2 repetitions). However, SA outperforms CS and IL-S+SA in over 50% of repetitions in this group. Indeed, SA exhibits a greater relative gap than CS and IL-S+SA at 50.8333% (between 10 and 11 repetitions) and 68.3333% (between 13 and 14 repetitions), respectively, in this group. In 30-period instances, SA outperforms IL-S+SA in just 13.7500% of repetitions per instance (between 2 and 3 repetitions), IL-S in 35.4167% (between 7 and 8 repetitions), and CS in 53.3333% of repetitions per instance (between 10 and 11 repetitions). In 120-period instances, SA surpasses IL-S+SA in 67.0833%

(between 13 and 14 repetitions), CS in 25.0000% (5 repetitions), and 14.1667% (between 2 and 3 repetitions) of repetitions per instance.

Finally, Table 6 presents a comparison of the performance between IL-S and IL-S+SA, grouping the instances by control parameters. The first column of Table 6 indicates the control parameter. The following columns are grouped in pairs, where each pair represents an indicator of IL-S and IL-S+SA. The first pair of columns indicates the average relative gap for each method. The next pair indicates the number of nodes explored by the branch-and-cut method. The last two pairs indicate the number of feasibility cuts and optimality cuts generated by the integer L-shaped method, respectively. Like Table 5, Table 6 groups rows by control parameters, and the last row shows the results for all instances.

Table 6. Comparison between the integer L-shaped method and this variant with simulated annealing solving the second-stage subproblems, when the instances are grouped by control parameter.

CP	ARG		ABCN		AFC		AOC	
	IL-S	IL-S+SA	IL-S	IL-S+SA	IL-S	IL-S+SA	IL-S	IL-S+SA
V10	**0.0630**	0.1720	36.833	101.667	28.083	51.500	4.833	13.333
V20	0.5233	**0.4587**	264.000	4381.750	183.917	1961.750	7.000	97.833
V30	1.2420	**1.1980**	143.000	4555.000	159.500	1907.333	3.583	120.083
L2	0.5961	**0.5479**	148.833	2982.333	120.278	1195.556	5.944	85.889
L3	**0.6228**	0.6712	147.056	3043.278	127.389	1418.167	4.333	68.278
R10	0.5533	**0.5436**	197.111	3116.722	158.222	1375.222	5.222	37.944
R20	**0.6656**	0.6755	98.778	2908.889	89.444	1238.500	5.056	116.222
T30	1.2082	**1.0531**	222.167	2296.250	171.250	1013.667	7.750	64.917
T60	**0.4026**	0.5278	72.167	3216.917	74.000	1437.500	2.917	70.833
T120	**0.2177**	0.2478	149.500	3525.250	126.250	1469.417	4.750	95.500
All	**0.6095**	**0.6096**	**147.944**	**3012.806**	**123.833**	**1306.861**	**5.139**	**77.083**

Abbreviations are defined as follows: CP: Control parameter; IL-S: Stochastic problem solved by the integer L-shaped method; IL-S+SA: Stochastic problem solved by the integer L-shaped method using simulated annealing in the second stage subproblems; ARG: Average relative gap; ABCN: Average number of nodes explored by the branch-and-cut procedure; AFC: Average Feasibility cuts; AOC: Average optimality cuts.

In general, IL-S and IL-S+SA perform similarly in terms of average relative gap. Table 6 shows that the average relative gap of IL-S is 0.0001 units better than IL-S+SA. Regarding the number of nodes explored during the first-stage relaxation, IL-S explores 147-148 nodes per instance, whereas IL-S+SA explores 3,012-3,013 nodes per instance. Furthermore, IL-S generates 123-124 feasibility cuts and 5-6 optimality cuts, while IL-S+SA produces 1306-1307 feasibility cuts and 77-78 optimality cuts. In both procedures, the number of optimality cuts includes the cut from the initial solution generated by LBSA-SP. The differences between IL-S and IL-S+SA in terms of the number of nodes and feasibility and optimality cuts are related to the method used in the second stage. Because IL-S uses the commercial solver in the second stage (a complex scheduling problem) and is limited to ten minutes per scenario, the method requires more time to explore nodes and generate feasibility and optimality cuts. On the other hand, IL-S+SA takes less time to evaluate the recourse function with

SA-RF, but the evaluation is less accurate. Therefore, IL-S+SA produces a large number of feasibility and optimality cuts.

In terms of average relative gap, IL-S performs better than IL-S+SA in 10-project instances but worse in 20- and 30-project instances. Table 6 shows that the average relative gap is 0.0630, 0.5233, and 1.2420 for IL-S, and 0.1720, 0.4587, and 1.1980 for IL-S+SA, for 10, 20, and 30 projects, respectively. Note the widening of the relative gap for both methods as the number of projects increases. The branch-and-cut procedure in IL-S explores between 36 and 37 nodes, 264 nodes, and 143 nodes in instances with 10, 20, and 30 projects, respectively, while IL-S+SA explores between 101 and 102 nodes, between 4381 and 4382 nodes, and 4555 nodes in instances with 10, 20, and 30 projects, respectively. IL-S generates between 28 and 29, 183 and 184, and 159 and 160 feasibility cuts in 10-, 20-, and 30-project instances, respectively, while IL-S+SA generates between 51 and 52, 1961 and 1962, and 1907 and 1908 feasibility cuts in the same instances. Moreover, IL-S produces between 4 and 5 optimality cuts in 10-project instances, 7 in 20-project instances, and 3 and 4 30-project instances, while IL-S+SA generates between 13 and 14, 97 and 98, and 120 and 121 optimality cuts in those same instances, respectively.

IL-S+SA has a better average relative gap than IL-S in 2-site instances, but a worse one in 3-site instances. Table 6 shows that the average relative gap for 2-site instances is 0.5961 and 0.5436, while for 3-site instances it is 0.6228 and 0.6712 for IL-S and IL-S+SA, respectively. The number of nodes explored by the branch-and-cut procedure in 2-site instances varies from 148 to 149 for IL-S and from 2982 to 2983 for IL-S+SA. For 3-site instances, the ranges are 147-148 for IL-S and 3043-3044 for IL-S+SA. In two-site instances, IL-S generates 120-121 feasibility cuts, and IL-S+SA generates 1,195-1,196. For 3-site instances, IL-S produces 127-128 feasibility cuts, while IL-S+SA produces 1418-1419. Furthermore, IL-S produces 5-6 optimality cuts, and IL-S+SA produces 85-86 in 2-site instances. For 3-site instances, IL-S produces 4-5 optimality cuts, and IL-S+SA produces 68-69.

In the 10-resource-type instances, the average relative gap for IL-S+SA is lower than that for IL-S; however, it is higher in the 20-resource-type instances. Table 6 indicates that the average relative gap of IL-S is 0.5533 for instances with 10 resource types and 0.6656 for those with 20 resource types. For IL-S+SA, these values are 0.5436 and 0.6755, respectively. In instances with 10 resource types, IL-S explores between 197 and 198 branch-and-cut nodes, and between 98 and 99 in instances with 20 resource types, while IL-S+SA explores between 3116 and 3117 nodes, and between 2908 and 2909 nodes, respectively. IL-S generates approximately 158 to 159 feasibility cuts in 10-resource-type instances and 89 to 90 in 20-resource-type instances, whereas IL-S+SA produces around 1735 to 1736 cuts and 1238 to 1239 cuts, respectively. Additionally, IL-S produces between 5 and 6 optimality cuts for instances with 10 and 20 resource types. IL-S+SA generates between 37 and 38 cuts, and between 116 and 117 cuts, respectively.

IL-S+SA outperforms IL-S in 30-period instances but performs worse in 60- and 120-period instances, with respect to the average relative gap. Table 6 indicates that the average relative gaps are 1.2082, 0.4026, and 0.2177 for IL-S across instances with 30, 60, and 120 periods. Meanwhile, IL-S+SA achieves an average relative gap of 1.0531, 0.5278, and 0.2478, respectively. IL-S examines between

222 and 223, 72 and 73, and 149 and 150 branch-and-cut nodes in 30-, 60-, and 120-period instances, respectively. In the meantime, IL-S+SA explores nodes between 2296 and 2297, 3216 and 3217, and 3525 and 3526, respectively. IL-S produces approximately 171 to 172 feasibility cuts for 30-period instances, 74 cuts for 60-period instances, and between 126 and 127 cuts for 120-period instances. In contrast, IL-S+SA generates around 1013 to 1014 cuts for 30-period instances, between 1437 and 1438 cuts for 60-period instances, and approximately 1469 to 1470 cuts for 120-period instances. Additionally, IL-S produces 7 to 8, 2 to 3, and 4 to 5 optimality cuts for instances with 30, 60, and 120 periods, respectively. Meanwhile, IL-S+SA generates 64-65, 70-71, and 95-96 cuts for these instances.

5.3. Performance Analysis

Table 3 shows that, overall, CS is outperformed by the other methods in terms of the average relative gap. However, the best SA repetition achieves a low relative gap in less than 100 seconds. IL-S has the second-best average relative gap; however, in some instances, the execution time exceeds 113 minutes (approximately 1 hour and 53 minutes). IL-S+SA completes in under 62 minutes but has a worse average relative gap than IL-S. Despite this result, CS and IL-S achieve optimal solutions in 7 out of 10 instances.

Table 4 presents an analysis of the instances, grouped by control parameters. The average relative gap and average execution time increase as instance size increases. Although the problem size does not depend on the horizon, the average relative gap and average execution time decrease in almost all methods as the horizon increases. A high horizon time provides more options for selecting and scheduling projects. Note that only in 10-project instances does CS have a better relative gap. In the other groups, the best SA repetition yields the lowest average relative gap. The second-lowest average relative gap is contested between IL-S and IL-S+SA.

Previous results suggest that SA outperforms other methods. However, Table 5 shows that overall, SA outperforms other methods in less than 50% of the repetitions per instance. In small instances (10-project), SA can outperform CS in fewer than 1% of repetitions per instance, meaning fewer than 2 out of 240 (20 repetitions across 12 instances). In other groups, SA outperforms IL-S in less than 29% of the repetitions per instance, except in the 30-period instances, where SA cannot outperform IL-S+SA by more than 14%. These results suggest a low likelihood that a straightforward execution of SA can outperform other methods, particularly IL-S. Nevertheless, for these test instances, a single SA repetition requires only 100 seconds, compared with the almost 6,800 seconds required by the slowest IL-S execution. This means that we can run SA 68 times to find a better solution than IL-S in the same time.

Finally, Table 6 compares the performance between IL-S and IL-S+SA. Generally, both methods yield similar results in terms of average relative gap. However, a difference appears when the control parameters group the instances. When instances are grouped by project or horizon time, IL-S shows a larger relative gap on the easiest instances. Nevertheless, if stations or resource types group instances, IL-S+SA has a larger relative gap on the easiest instances. The difference between IL-S and IL-S+SA is notable in the number of branch-and-cut nodes explored and the counts of feasibility and

optimality cuts. In general, the number of branch-and-cut nodes explored by IL-S+SA is roughly 20 times greater than IL-S.

Additionally, the number of feasibility and optimality cuts produced by IL-S+SA is about 10 and 15 times higher than those generated by IL-S, respectively. Those differences vary when the results are analyzed based on the control parameter, but the methods remain consistent. Clearly, those differences are due to the method used to evaluate the recourse function. Because SA-RF is used in IL-S+SA, the method can quickly determine whether a first-stage solution is feasible in the second-stage subproblems, albeit less precisely. Thus, the first-stage problem can be solved again by adding a new cut from the subproblems and generating new branch-and-cut nodes if the solution is non-integer. This cycle repeats quickly, resulting in numerous cuts and nodes. On the other hand, because IL-S uses Gurobi to solve the second-stage subproblems (limited to 10 minutes), cut generation and node exploration are slower than those of IL-S+SA (in the worst case, the first-stage solution can be infeasible at the last scenario). Because the lower precision of IL-S+SA is offset by a higher number of cuts and nodes, the average relative gaps are similar to those of IL-S.

6. Discussion

The computational results reveal clear performance trade-offs among the four solution strategies and illustrate how different modeling choices influence both accuracy and scalability. Overall, the simulated annealing (SA) metaheuristic provides high-quality solutions within very short execution times, confirming its suitability for large instances where exact approaches become intractable. Although the cyclical search behavior of SA does not consistently outperform the exact or decomposition-based methods in every repetition, its rapid runtime enables multiple independent runs to be executed within the time required for a single IL-S or IL-S+SA solve. This behavior increases the probability of reaching solutions that are competitive with, or superior to, those of the more computationally intensive methods.

The comparison between the two decomposition strategies, IL-S and IL-S+SA, highlights the impact of the recourse evaluation mechanism on overall performance. IL-S relies on exact second-stage solves, which improves precision but introduces substantial computational overhead, especially when the scenario set is large. By contrast, IL-S+SA replaces the exact recourse evaluation with a SA-based approximation. This surrogate recourse model dramatically accelerates feasibility checks and cut generation, resulting in a considerably larger number of branch-and-cut nodes, feasibility cuts, and optimality cuts. Although this increases algorithmic activity, the reduced precision of the approximate recourse function results in slightly larger average optimality gaps. Nevertheless, IL-S+SA remains competitive when multiple stations or resource types introduce additional combinatorial complexity, because the faster recourse evaluation allows the decomposition framework to explore the search space more aggressively.

A further insight from the experiments is that problem characteristics, such as the number of projects and time horizon, strongly influence the relative performance of the methods. Instances with a longer horizon offer more temporal flexibility, expanding the feasible solution space and enabling both exact and heuristic approaches to obtain better solutions with reduced computational

effort. Conversely, instances with more projects or limited temporal slack yield tighter feasibility regions, thereby amplifying the difficulty of the second-stage scheduling tasks. In these settings, SA's flexibility and speed are particularly advantageous. In contrast, IL-S may struggle due to the computational cost of solving many constrained-scheduling subproblems to optimality in each scenario.

These findings align with observations reported in the stochastic scheduling and project portfolio optimization literature, where hybrid approaches often provide a balance between solution quality and computational scalability. The behavior of IL-S+SA is consistent with simheuristic frameworks that combine stochastic optimization with heuristic search to handle problematic recourse structures. At the same time, the strong performance of pure SA confirms the relevance of metaheuristics for large-scale resource-constrained scheduling. The results reinforce the idea that the value of decomposition is sensitive to the cost of evaluating second-stage subproblems, a pattern also identified in earlier studies on integer L-shaped and Benders-type methods applied to complex scheduling models.

From a managerial perspective, the results demonstrate that SA is particularly well-suited for operational environments that require rapid, good-quality solutions, such as daily or weekly adjustments to Antarctic research plans. When higher accuracy is required for strategic planning or high-impact project selection, IL-S remains a strong choice, provided that computational resources are available. The IL-S+SA hybrid method offers an intermediate alternative, yielding solutions of acceptable quality within more moderate runtimes, and is especially useful when multi-site resource transfers introduce additional complexity that hinders the tractability of exact decomposition.

Taken together, the computational analysis shows that no single method dominates across all instance types. Instead, each approach offers a distinct balance between precision and computational effort. SA provides excellent performance for large, time-sensitive problems; IL-S offers accurate solutions for smaller or strategic planning instances; and IL-S+SA bridges the gap when problem complexity or time constraints make exact recourse evaluation less viable. These insights guide both methodological choices for future research and practical decision-making in Antarctic research operations, where uncertainty, limited resources, and environmental constraints require flexible, scalable optimization tools.

7. Conclusions

This study has developed a comprehensive stochastic programming framework for selecting and scheduling scientific research projects across Antarctic stations under substantial operational uncertainty. The contributions are fourfold. First, it introduces a two-stage stochastic model that links first-stage project selection with second-stage scenario-dependent scheduling to maximize expected scientific benefit. Second, it proposes an exact decomposition approach based on the integer L-shaped method to exploit the problem's structural properties and improve computational tractability. Third, it develops a hybrid method that integrates simulated annealing into the recourse evaluation to accelerate convergence. Fourth, it implements a tailored metaheuristic that produces high-quality solutions for large-scale instances where exact approaches become impractical.

Together, these contributions expand the methodological toolkit for addressing stochastic project scheduling in remote and resource-constrained environments.

From a theoretical perspective, the study emphasizes the significance of stochastic programming in areas where logistical constraints intersect with high levels of environmental uncertainty. The integration of hybrid and metaheuristic approaches contributes to the growing body of literature focused on scalable solutions for large stochastic scheduling problems, particularly when traditional exact methods encounter computational limitations. From a managerial perspective, the findings offer guidance to institutions and policymakers involved in Antarctic operations, where resource allocation and project prioritization must balance scientific value, operational practicality, and unpredictable climate conditions. The methods presented here can support more robust portfolio planning and resource deployment, thereby improving the effectiveness and resilience of polar research efforts.

However, the study has several limitations. First, the model has not yet been validated with real Antarctic project data, which limits the immediate applicability of the results. Second, the decomposition strategy struggles to efficiently solve large second-stage subproblems, especially as the size of the scenario set grows. Third, relying on list-based scheduling heuristics within the metaheuristic framework creates rigidity and limits exploration of more diverse solution spaces. These issues highlight opportunities to improve both the modeling approach and the solution methods.

Future research could explore many new directions. Multi-objective approaches would enable the simultaneous optimization of additional factors, such as risk exposure, fair resource use, and environmental impact. Using different ways to handle uncertainty, such as robust or distributionally robust optimization, could improve how solutions withstand variations. Adding objectives such as reducing operational costs, increasing project diversity, or promoting fairness among research groups would enhance the framework's decision-making power. Methodologically, using more advanced decomposition methods, machine-learning-based heuristics, or sophisticated hybrid metaheuristics would improve scalability and solution quality. Lastly, applying the research to real-world Antarctic case studies is a crucial step to validate its practical usefulness and managerial relevance.

Acknowledgements

This work is funded by ANID—Subdirección de Capital Humano/Doctorado Nacional/2021 under Grant 21210451.

Declaration of Interest statement

The authors declare that they have no conflict of interest.

CRediT authorship contribution statement

M. Vega-Hidalgo: Conceptualization, Methodology, Software, Formal Analysis, Writing – Original Draft, Visualization, Funding acquisition**. L. Pradenas:** Conceptualization, Writing – Review & Editing, Supervision**. V. Parada:** Validation, Writing – Review & Editing**. H. Cancela:** Conceptualization, Supervision.

Data availability statement

The data supporting this study's findings are openly available in Zenodo at https://doi.org/10.5281/zenodo.17849465

References

Benders, J. F. (1962). Partitioning procedures for solving mixed-variables programming problems. *Numerische Mathematik*. 4, 238–252. https://doi.org/10.1007/BF01386316

Bigler, T., Gnägi, M., & Trautmann, N. (2024). MIP-based solution approaches for multi-site resource-constrained project scheduling. *Annals of Operations Research*, 337(2), 627–647. https://doi.org/10.1007/s10479-022-05109-0

Chao, Y., Zhuang, C., Guo, H., & Liu, J. (2026). A genetic programming hyper-heuristic with whale optimization algorithm for the dynamic resource-constrained multi-project scheduling problems. *Expert Systems with Applications*, 295, 128881. https://doi.org/10.1016/j.eswa.2025.128881

Chen, H., Ding, G., Qin, S., & Zhang, J. (2021). A hyper-heuristic based ensemble genetic programming approach for stochastic resource constrained project scheduling problem. *Expert Systems with Applications*, 167, 114174. https://doi.org/10.1016/j.eswa.2020.114174

Chen, Y., Demeulemeester, E., & Matuschke, J. (2025). A purely buffer-based approach to the proactive and reactive resource-constrained project scheduling problem. *Annals of Operations Research*. https://doi.org/10.1007/s10479-025-06688-4

Dashti, F., Fallahi, A., & Mokhtari, H. (2025). A sustainable multiobjective multi-site resource-constrained project scheduling problem. *Computers and Industrial Engineering*, 203, 110968. https://doi.org/10.1016/j.cie.2025.110968

Dong, Y., Zheng, W., Ma, Z., & He, Z. (2025). Two-stage robust optimization for public health emergency project scheduling with uncertain activity durations. *Computers and Operations Research*, 182, 107135. https://doi.org/10.1016/j.cor.2025.107135

Fu, F., Liu, Q., & Yu, G. (2024). Robustifying the resource-constrained project scheduling against uncertain durations. *Expert Systems with Applications*, 238, 122002. https://doi.org/10.1016/j.eswa.2023.122002

Ghasemi Bojd, F., & Koosha, H. (2018). A robust goal programming model for the capital budgeting problem. *Journal of the Operational Research Society*, 69(7), 1105–1113. https://doi.org/10.1080/01605682.2017.1389673

Golab, A., Gooya, E. S., Al Falou, A., & Cabon, M. (2023). A convolutional neural network for the resource-constrained project scheduling problem (RCPSP): A new approach. *Decision Science Letters*, 12(2), 225–238. https://doi.org/10.5267/j.dsl.2023.2.002

Gonçalves, J. F., Mendes, J. J. M., & Resende, M. G. C. (2008). A genetic algorithm for the resource constrained multi-project scheduling problem. *European Journal of Operational Research*, 189(3), 1171–1190. https://doi.org/10.1016/j.ejor.2006.06.074

Habibi, F., Chakrabortty, R. K., Servranckx, T., Abbasi, A., & Vanhoucke, M. (2025). Project portfolio selection and scheduling problem under material supply uncertainty. *Operations Management Research*, 18(1), 226–256. https://doi.org/10.1007/s12063-024-00532-x

Han, R., Li, X., Shen, Z., & Jia, D. (2024). A framework of robust project portfolio selection problem under strategic objectives considering the risk propagation. *Engineering, Construction and Architectural Management*, 31(12), 4872–4896. https://doi.org/10.1108/ECAM-08-2022-0801

Harrison, K. R., Elsayed, S. M., Weir, T., Garanovich, I. L., Boswell, S. G., & Sarker, R. A. (2022). Solving a novel multi-divisional project portfolio selection and scheduling problem. *Engineering Applications of Artificial Intelligence*, 112, 104771. https://doi.org/10.1016/j.engappai.2022.104771

Hesarsorkh, A. H., Ashayeri, J., & Naeini, A. B. (2021). Pharmaceutical R&D project portfolio selection and scheduling under uncertainty: A robust possibilistic optimization approach. Computers & Industrial Engineering, 155(January 2020), 107114. https://doi.org/10.1016/j.cie.2021.107114

Kennicutt, M. C., Chown, S. L., Cassano, J. J., Liggett, D., Massom, R., Peck, L. S., Rintoul, S. R., Storey, J. W. V., Vaughan, D. G., Wilson, T. J., & Sutherland, W. J. (2014). Polar research: Six priorities for Antarctic science. *Nature*, 512(7512), 23–25. https://doi.org/10.1038/512023a

Kennicutt, M. C., Chown, S. L., Cassano, J. J., Liggett, D., Peck, L. S., Massom, R., Rintoul, S. R., Storey, J., Vaughan, D. G., Wilson, T. J., Allison, I., Ayton, J., Badhe, R., Baeseman, J., Barrett, P. J., Bell, R. E., Bertler, N., Bo, S., Brandt, A., … Sutherland, W. J. (2015). A roadmap for Antarctic and Southern Ocean science for the next two decades and beyond. *Antarctic Science*, 27(1), 3–18. https://doi.org/10.1017/S0954102014000674

Kirkpatrick, S., Gelatt, C. D., & Vecchi, M. P. (1983). Optimization by simulated annealing. *Science*, 220(4598), 671–680. https://doi.org/10.1126/science.220. 4598.671

Laporte, G., & Louveaux, F. V. (1993). The integer L-shaped method for stochastic integer programs with complete recourse. *Operations Research Letters*, 13(3), 133–142. https://doi.org/10.1016/0167-6377(93)90002-X

Laurent, A., Deroussi, L., Grangeon, N., & Norre, S. (2017). A new extension of the RCPSP in a multi-site context: Mathematical model and metaheuristics. *Computers and Industrial Engineering*, 112, 634–644. https://doi.org/10.1016/j.cie.2017.07.028

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

Li, L., Zhang, J., Zhang, H., & Leus, R. (2025). A novel hyper-heuristic based on surrogate genetic programming for the three-dimensional spatial resource-constrained project scheduling problem under uncertain environments. *Computers and Operations Research*, 179, 107013. https://doi.org/10.1016/j.cor.2025.107013

Li, X., Fang, S.-C., Guo, X., Deng, Z., & Qi, J. (2016). An extended model for project portfolio selection with project divisibility and interdependency. *Journal of Systems Science and Systems Engineering*, 25(1), 119–138. https://doi.org/10.1007/s11518-015-5281-1

Liu, X., Liang, J., Zhang, Z., Yang, S., Peukert, S., & Lanza, G. (2025). Project selection and scheduling with multiplicative enhancement effects and delay risk: An application in intelligent manufacturing technologies. IISE Transactions, 57(8), 873–889. https://doi.org/10.1080/24725854.2024.2374090

Ma, Z., He, Z., Wang, N., Yang, Z., & Demeulemeester, E. (2019a). A Genetic Algorithm for the Proactive Resource-Constrained Project Scheduling Problem With Activity Splitting. *IEEE Transactions on Engineering Management*, 66(3), 459–474. https://doi.org/10.1109/TEM.2018.2819689

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

Melchiors, P., Kolisch, R., & Kanet, J. J. (2024). The performance of priority rules for the dynamic stochastic resource-constrained multi-project scheduling problem: an experimental investigation. *Annals of Operations Research*, 338(1), 569–595. https://doi.org/10.1007/s10479-024-05841-9

Nascimento, C. R. S. de M. S., Almeida-Filho, A. T. de, & Perez Palha, R. (2025). A TOPSIS-based framework for construction projects' portfolio selection in the public sector. *Engineering, Construction and Architectural Management*, 32(4), 2553–2570. https://doi.org/10.1108/ECAM-05-2023-0534

Nielsen, M. K., Jacobsen, A. M. S. Ø., Carstensen, J. L., Nielsen, M. T., & Tambo, T. (2024). Industrial R&D project portfolio selection method using a multi-objective optimization program: A conceptual quantitative framework. *Journal of Industrial Engineering and Management*, 17(1), 217. https://doi.org/10.3926/jiem.6552

Padberg, M., & Rinaldi, G. (1991). A Branch-and-Cut Algorithm for the Resolution of Large-Scale Symmetric Traveling Salesman Problems. *SIAM Review*, 33(1), 60–100. https://doi.org/10.1137/1033004

Panadero, J., Doering, J., Kizys, R., Juan, A. A., & Fito, A. (2020). A variable neighborhood search simheuristic for project portfolio selection under uncertainty. *Journal of Heuristics*, 26(3), 353–375. https://doi.org/10.1007/s10732-018-9367-z

Peng, W., Lin, X., & Li, H. (2023). Critical chain based Proactive-Reactive scheduling for Resource-Constrained project scheduling under uncertainty. *Expert Systems with Applications*, 214, 119188. https://doi.org/10.1016/j.eswa.2022.119188

Rahimi, F., Davari-Ardakani, H., Ameli, M., & Kabiri Beheshtkhah, M. (2024). Sustainable project selection and scheduling using scenario-based stochastic programming: a case study of industrial projects. *Stochastic Environmental Research and Risk Assessment*, 38(2), 593–619. https://doi.org/10.1007/s00477-023-02589-9

RezaHoseini, A., Ghannadpour, S. F., & Hemmati, M. (2020). A comprehensive mathematical model for resource-constrained multi-objective project portfolio selection and scheduling considering sustainability and projects splitting. *Journal of Cleaner Production*, 269, 122073. https://doi.org/10.1016/j.jclepro.2020.122073

Sadri, S., Ghomi, S. M. T. F., & Dehghanian, A. (2024). Analysis of a time–cost trade-off in a resource-constrained GERT project scheduling problem using the Markov decision process. Annals of Operations Research, 338(1), 535–568. https://doi.org/10.1007/s10479-024-05896-8

Saiz, M., Calvet, L., Juan, A. A., & Lopez-Lopez, D. (2024). A simheuristic for project portfolio optimization combining individual project risk, scheduling effects, interruptions, and project risk correlations. *Computers & Industrial Engineering*, 198, 110694. https://doi.org/10.1016/j.cie.2024.110694

Satic, U., Jacko, P., & Kirkbride, C. (2024). A simulation-based approximate dynamic programming approach to dynamic and stochastic resource-constrained multi-project scheduling problem. *European Journal of Operational Research*, 315(2), 454–469. https://doi.org/10.1016/j.ejor.2023.10.046

Shahabi-Shahmiri, R., Tavakkoli-Moghaddam, R., Dolgui, A., Mirnezami, S. A., Ghasemi, M., & Ahmadi, M. (2024). Preemptive and non-preemptive multi-skill multi-mode resource-constrained project scheduling problems considering sustainability and energy consumption: A comprehensive mathematical model. *Journal of Environmental Management*, 367, 121986. https://doi.org/10.1016/j.jenvman.2024.121986

Song, J., Song, J., & Vanhoucke, M. (2025). Automatic selection of the best performing control point approach for project control with resource constraints. *European Journal of Operational Research*, 322(1), 15–38. https://doi.org/10.1016/j.ejor.2024.10.025

Tritschler, M., Naber, A., & Kolisch, R. (2017). A hybrid metaheuristic for resource-constrained project scheduling with flexible resource profiles. *European Journal of Operational Research*, 262(1), 262–273. https://doi.org/10.1016/j.ejor.2017.03.006

Tselios, D., Papageorgiou, G., Alamanis, N., & Ipsilandis, P. (2024). Energy project portfolio selection and scheduling. Energy Systems, 15(4), 1471–1480. https://doi.org/10.1007/s12667-022-00503-w

van den Houten, K., Planken, L., Freydell, E., Tax, D. M. J., & de Weerdt, M. (2025). Proactive and Reactive Constraint Programming for Stochastic Project Scheduling with Maximal Time-Lags. *Proceedings of the AAAI Conference on Artificial Intelligence*, 39(25), 26534–26541. https://doi.org/10.1609/aaai.v39i25.34854

Vega-Hidalgo, M., Pradenas, L., & Parada, V. (2026). Optimizing research in Antarctica: a novel approach to project selection and scheduling across multiple stations. *Journal of the Operational Research Society*, 77(1), 286–312. https://doi.org/10.1080/01605682.2025.2495766

Xie, F., Li, H., & Xu, Z. (2021). An approximate dynamic programming approach to project scheduling with uncertain resource availabilities. *Applied Mathematical Modelling*, 97, 226–243. https://doi.org/10.1016/j.apm.2021.03.048

Xie, F., Li, Y., & Li, H. (2025). Integrating simulation, optimization and reinforcement learning for a general class of stochastic scheduling problems. *Expert Systems with Applications*, 279, 127488. https://doi.org/10.1016/j.eswa.2025.127488

Xu, J., Bai, S. (2024). A reactive scheduling approach for the resource-constrained project scheduling problem with dynamic resource disruption. Kybernetes, 53(6), 2007-2028. https://doi.org/10.1108/K-09-2022-1339

Zhang, H., Li, L., Demeulemeester, E., Bai, S., & Zhang, J. (2025). Heuristic Methods for Dynamic Multiproject Scheduling Problems with Stochastic Transfer Times. *Journal of Construction Engineering and Management*, 151(2), 1–15. https://doi.org/10.1061/JCEMD4.COENG-15342

Capítulo 4

Discusión

En los Capítulos 2 y 3 se aborda el problema de la selección y programación de proyectos en múltiples bases antárticas desde dos perspectivas complementarias. En primer lugar, el Capítulo 2 estudia la versión determinística del problema (RPSAP), formulándola como un modelo de programación entera mixta y proponiendo metaheurísticas orientadas a obtener soluciones de alta calidad en tiempos computacionales acotados. A continuación, el Capítulo 3 desarrolla una extensión estocástica en dos etapas (SRPSAP), incorporando la incertidumbre en la duración de los proyectos mediante escenarios: la primera etapa determina decisiones de selección, mientras que la segunda etapa ajusta la calendarización de los proyectos seleccionados; para ello, se consideran enfoques de descomposición (integer L-shaped) y esquemas híbridos con metaheurísticas.

Según la evidencia computacional reportada en los Capítulos 2 y 3, puede sostenerse una conclusión transversal: en el contexto antártico (caracterizado por restricciones logísticas, recursos renovables distribuidos por base, tiempos de traslado y relaciones de precedencia) la resolución exacta mediante MIP resulta adecuada solo en instancias pequeñas, mientras que el desempeño robusto en escalas realistas requiere estrategias metaheurísticas y/o híbridas (descomposición con heurísticas).

En el Capítulo 2, el RPSAP se aborda mediante una formulación MIP y tres metaheurísticas (ILS, VNS y SA), evaluadas en 480 instancias. El resultado más consistente es la superioridad de las metaheurísticas respecto de los modelos matemáticos en instancias medianas y grandes, tanto en términos de brecha relativa como de escalabilidad; en particular, se reporta que las brechas relativas promedio de los modelos (CTM e ITM) se sitúan alrededor de 1.310 y 1.277, respectivamente, mientras que en ejecuciones metaheurísticas la brecha promedio por repetición no supera 0.405, lo que constituye una mejora sustantiva en calidad relativa bajo presupuestos de tiempo razonables.

Un matiz relevante es que la ventaja del enfoque exacto se limita al régimen de instancias pequeñas. En la clasificación por clúster, los modelos superan a las metaheurísticas solo en el clúster de menor tamaño (clúster 0), donde las brechas son comparables y el modelo logra con mayor frecuencia la mejor solución relativa; fuera de este régimen, las metaheurísticas dominan de manera sistemática, y en clústeres de mayor dificultad se observa incluso que los modelos no alcanzan la mejor brecha en ninguna instancia, sugiriendo un quiebre de escalabilidad propio de formulaciones MIP frente a incrementos en tamaño y complejidad combinatoria (selección, programación y asignación de recursos).

Respecto de la comparación interna entre metaheurísticas, los resultados muestran una compensación estable entre calidad y tiempo: ILS tiende a concentrar el mejor desempeño en la calidad de la solución (y en la escalabilidad), mientras que SA destaca por tiempos de ejecución considerablemente menores, aun cuando exhibe, en promedio, brechas más altas que ILS y VNS. En clústeres difíciles, SA mantiene tiempos promedio del orden de decenas de segundos, significativamente inferiores a los tiempos promedio de ILS y, especialmente, a los de VNS; además, se evidencia que, a

medida que crece el tamaño de instancia, la tasa de crecimiento del tiempo de VNS es mayor que la de ILS, lo cual posiciona a ILS como alternativa más estable al aumentar la dificultad combinatoria, y a SA como alternativa rápida para decisiones con fuerte restricción temporal.

En síntesis, el Capítulo 2 ofrece una sólida justificación empírica para privilegiar (i) ILS cuando el objetivo principal es minimizar la brecha relativa de calidad y (ii) SA cuando el objetivo operacional es obtener soluciones buenas en tiempos acotados, particularmente en contextos donde la planificación requiere iteraciones frecuentes o análisis de sensibilidad. Esta lectura es clave porque en el Capítulo 3 la dificultad se amplifica al incorporar escenarios, lo que vuelve aún más crítico el costo de resolver repetidamente subproblemas de scheduling.

El Capítulo 3 formula el SRPSAP como un problema estocástico de dos etapas con duraciones inciertas (escenarios) y ventanas de tiempo, y propone la descomposición integer L-shaped (IL-S) para explotar la estructura maestro–subproblemas, además de una variante IL-S+SA que utiliza SA para la resolución aproximada de los subproblemas de segunda etapa, y un esquema SA directo para el problema estocástico. La evaluación se realiza sobre 36 instancias adaptadas del caso determinístico, comparando cuatro enfoques: solver comercial (CS), IL-S, IL-S+SA y SA.

A nivel agregado, los resultados reportan que CS es superado por los métodos propuestos en términos de brecha relativa, y que tanto IL-S como IL-S+SA logran brechas competitivas frente al solver directo, lo que confirma que la descomposición (y la hibridación) capturan eficientemente la estructura del problema. En paralelo, SA puede alcanzar brechas bajas en tiempos inferiores a 100 segundos cuando se considera su mejor repetición, lo que lo convierte en una alternativa atractiva para decisiones rápidas o como mecanismo para generar soluciones iniciales de alta calidad.

Sin embargo, el análisis con parámetros de control arroja dos hallazgos metodológicamente relevantes. Primero, la brecha relativa y el tiempo aumentan con el tamaño (p. ej., número de proyectos), tal como se espera; pero, pese a que el tamaño del modelo no depende del horizonte de tiempo, se observa que el aumento del horizonte tiende a reducir brechas y tiempos en casi todos los métodos, lo que sugiere que horizontes más amplios incrementan la holgura de programación y facilitan la factibilidad/recourse, estabilizando la búsqueda en los escenarios (más opciones de calendarización).

Segundo, la comparación IL-S vs IL-S+SA explicita un costo estructural de la aproximación: aunque ambas variantes presentan brechas promedio prácticamente indistinguibles en el agregado, IL-S+SA explora un orden de magnitud mayor de nodos y genera sustantivamente más cortes (factibilidad y optimalidad). Este fenómeno se presenta como consecuencia directa de evaluar más rápido, pero con menor precisión, la función de recurso cuando SA reemplaza al solver exacto en subproblemas: la velocidad acelera el ciclo "probar–cortar–ramificar", pero la menor precisión induce más iteraciones y cortes para estabilizar el maestro y cerrar la brecha de manera consistente.

Finalmente, se enfatiza un aspecto de robustez estadística de SA: aunque su mejor repetición puede dominar en ciertos grupos, en términos de frecuencia SA no supera a los demás métodos en la mayoría de las repeticiones (es decir, su superioridad no es determinística y depende de la variabilidad estocástica de la búsqueda). No obstante, este riesgo se mitiga operacionalmente porque una repetición de SA es muy barata en comparación con ejecuciones lentas de IL-S; por ello, la estrategia natural es ejecutar múltiples repeticiones de SA (multi-start) dentro del presupuesto de tiempo de un método exacto/híbrido, incrementando la probabilidad de obtener soluciones comparables o mejores en el mismo tiempo total.

La lectura conjunta de ambos capítulos sugiere una arquitectura de decisión coherente: (i) usar metaheurísticas (en especial SA) como mecanismo rápido para obtener soluciones iniciales y/o apoyar la evaluación intensiva bajo escenarios; (ii) usar ILS (determinístico) o IL-S (estocástico) cuando el énfasis sea la calidad y estabilidad de la solución, aceptando tiempos mayores; y (iii) adoptar variantes híbridas (IL-S+SA) cuando se requiera acelerar la evaluación del recourse, asumiendo un

incremento sustantivo en la exploración del árbol y generación de cortes producto de evaluaciones aproximadas. Esta conclusión es coherente con la motivación del manuscrito: dada la dificultad de programación y la evidencia del Capítulo 2, se justifica resolver subproblemas con SA dentro de la descomposición estocástica, en una lógica de equilibrio entre precisión y costo computacional.

Capítulo 5

Conclusiones

Esta tesis ha desarrollado un marco de modelamiento y resolución para apoyar la toma de decisiones en la planificación de expediciones científicas en la Antártica chilena, integrando explícitamente la selección y la programación de proyectos de investigación bajo restricciones logísticas y de recursos altamente limitados. El trabajo se estructura en dos ejes complementarios: (i) un problema determinístico de selección y programación en múltiples bases antárticas (RPSAP), y (ii) una extensión estocástica de dos etapas (SRPSAP) que incorpora incertidumbre —principalmente— en la duración de los proyectos mediante escenarios.

La evidencia computacional obtenida permite sostener una conclusión transversal: los modelos exactos de programación lineal entera mixta son apropiados como referencia metodológica y para instancias pequeñas, mientras que la obtención de planes robustos y oportunos en escalas realistas exige metaheurísticas y/o esquemas híbridos, particularmente cuando se requiere resolver repetidamente problemas de programación (determinísticos o por escenario) con restricciones interbase, precedencias y recursos renovables distribuidos.

En términos agregados, la contribución de la tesis consiste en proponer modelos matemáticos y algoritmos escalables para una familia de problemas de planificación científica en contextos extremos, cerrando la brecha entre las formulaciones de optimización y su uso como herramientas efectivas para apoyar la gestión operativa de recursos y de portafolios de investigación. En detalle, las contribuciones de la tesis se sintetizan en los siguientes resultados: (i) se formuló un modelo MIP determinístico para el RPSAP que integra recursos por base, traslados interbase, precedencias y beneficio neto; (ii) se diseñaron ILS, VNS y SA para el RPSAP, observándose un patrón consistente donde ILS tiende a liderar en calidad y SA en rapidez; (iii) se extendió el problema al SRPSAP mediante un modelo estocástico de dos etapas (selección y programación por escenario); (iv) se desarrolló un integer L-shaped (IL-S) con estructura maestro–subproblemas; (v) se propuso IL-S+SA para aproximar la función de recourse mediante SA-RF y acelerar la iteración; y (vi) se implementó SA-SP para resolver directamente el SRPSAP vía list-scheduling, útil como solución rápida o como inicialización en esquemas de descomposición.

5.1. Implicancias teóricas

5.1.1. Modelo determinístico para el RPSAP

Desde una perspectiva teórica, el RPSAP contribuye a extender modelos clásicos de selección y programación de proyectos con restricciones de recursos hacia un entorno multisitio donde (i) los recursos se encuentran distribuidos por base, (ii) existen tiempos de transferencia que acoplan decisiones logísticas con la calendarización, y (iii) la factibilidad y el rendimiento dependen de manera

sustantiva de la interacción entre precedencias, duración y disponibilidad interbase. La formalización mediante programación entera mixta provee una base rigurosa para analizar propiedades del problema, definir cotas y caracterizar quiebres de escalabilidad, en particular al aumentar el número de proyectos y los tipos de recursos.

Además, la tesis evidencia empíricamente que el rendimiento de los modelos exactos se degrada con rapidez al pasar de instancias pequeñas a medianas y grandes, lo que motiva la necesidad de estrategias de solución aproximada o híbrida. Este resultado refuerza un patrón ampliamente observado en la programación de proyectos de gran escala: la calidad de los planes depende tanto del modelamiento como de la capacidad de resolverlo en plazos compatibles con la toma de decisiones.

5.1.2. Metaheurísticas para el RPSAP

La contribución metodológica de las metaheurísticas radica en dos elementos: (i) el diseño de mecanismos de exploración/explotación adaptados a la estructura combinatoria del RPSAP (selección, programación y asignación de recursos/bases), y (ii) la robustez computacional obtenida en un conjunto amplio de instancias. En particular, los resultados sugieren que:

- ILS constituye una alternativa estable al aumentar la dificultad combinatoria, preservando la calidad de la solución cuando otras estrategias incrementan desproporcionadamente el tiempo.
- SA logra soluciones competitivas en tiempos reducidos, lo que la hace adecuada para escenarios que requieren iteraciones frecuentes, análisis de sensibilidad o replanificación.

Esta evidencia sustenta una implicancia teórica relevante: para problemas NP-difíciles multisitio con recursos renovables distribuidos, el "diseño algorítmico" (vecindarios, perturbaciones, mecanismos de aceptación y evaluación) es tan determinante como la formulación exacta, especialmente cuando el objetivo es la escalabilidad.

5.1.3. Modelo estocástico de dos etapas para el SRPSAP

La formulación de dos etapas para el SRPSAP constituye una contribución al integrar, en un mismo marco, decisiones de portafolio (selección) con decisiones de programación por escenario, en presencia de ventanas temporales y de transferencias interbase. Este acoplamiento es conceptualmente importante: la incertidumbre en las duraciones no solo afecta las fechas de término, sino que también puede inducir la inviabilidad debido a restricciones de recursos, precedencias y ventanas, lo que transforma la selección en una decisión inherentemente dependiente de las decisiones de segunda etapa.

En consecuencia, la tesis refuerza la relevancia de la programación estocástica en contextos extremos donde la incertidumbre operativa (clima, accesibilidad, retrasos) se traduce en cambios estructurales en la factibilidad y en el desempeño esperado.

5.1.4. *Integer L-shaped* para el SRPSAP

El desarrollo del método IL-S formaliza un enfoque de descomposición que explota el carácter de dos etapas del problema, separando la decisión de selección del esfuerzo de calendarización por escenario. Teóricamente, ello permite:

- Estructurar el problema como un proceso iterativo de generación de cortes, en el que la información de la segunda etapa retroalimenta la selección.

- Estudiar el impacto del número de escenarios sobre la convergencia y la carga computacional, explicitando que el crecimiento del conjunto de escenarios incrementa el costo de resolver subproblemas y, por extensión, la necesidad de aproximaciones

5.1.5. Hibridación mediante SA en la función de recourse

La propuesta IL-S+SA aporta a la literatura sobre métodos híbridos al mostrar cómo una metaheurística puede integrarse en el núcleo de una descomposición estocástica, reemplazando (parcialmente) la resolución exacta de subproblemas por una evaluación aproximada. Esta idea es particularmente relevante en problemas en los que la segunda etapa es la fuente dominante de complejidad: una evaluación rápida, aunque menos precisa, puede permitir una exploración más amplia del espacio de selección y acelerar la obtención de soluciones de buena calidad.

La tesis evidencia, además, un fenómeno metodológico importante: cuando la evaluación de recourse pierde precisión, la descomposición puede requerir más iteraciones/cortes para estabilizar el problema maestro. Este equilibrio entre precisión y velocidad es clave para diseñar algoritmos escalables y sugiere líneas de investigación concretas para la estabilización y el fortalecimiento de cortes.

5.1.6. SA para el SRPSAP

El diseño de SA-SP constituye una alternativa directa (no basada en un solver) para resolver el SRPSAP bajo restricciones temporales estrictas. Más allá de su desempeño en términos de brecha relativa, su valor teórico incluye la posibilidad de actuar como generador de soluciones iniciales en esquemas de descomposición o como componente de matheurísticas que combinan búsqueda estocástica con estructuras exactas.

5.2. Implicancias prácticas

Las implicancias prácticas se sitúan en la gestión de instituciones responsables de coordinar la investigación antártica (p. ej., organismos nacionales y unidades logísticas asociadas), donde la planificación debe compatibilizar el valor científico, la factibilidad logística, el uso eficiente de los recursos y la minimización de la intervención humana y ambiental. En particular, los modelos y algoritmos desarrollados pueden apoyar:

- **Priorización transparente de proyectos:** selección coherente con las restricciones reales de recursos y las precedencias, evitando portafolios inviables.
- **Planificación operativa de campañas:** calendarizaciones por base y por horizonte que internalizan los tiempos de traslado y la disponibilidad de recursos.
- **Análisis de sensibilidad y replanificación:** uso de metaheurísticas (especialmente SA) para evaluar rápidamente escenarios alternativos (cambios de recursos, retrasos, ventanas).
- **Gestión bajo incertidumbre:** mediante SRPSAP, evaluar ex ante el desempeño esperado y el riesgo operativo derivado de duraciones inciertas, lo que ayuda a construir planes más resilientes.

Asimismo, la comparación sistemática entre enfoques exactos, de descomposición y metaheurísticos aporta criterios operativos para seleccionar herramientas según el propósito:

- **Planeamiento estratégico con alta precisión:** IL-S (si el tamaño o los escenarios lo permiten).
- **Planeamiento táctico con un buen equilibrio entre precisión y tiempo:** IL-S+SA.
- **Decisiones rápidas y exploración inicial:** SA y SA-SP, particularmente útiles cuando se requieren iteraciones frecuentes.

5.3. Validación empírica

La tesis incorpora una validación empírica extensa, orientada a evaluar tanto calidad de solución como esfuerzo computacional:

- Para el RPSAP, se diseñaron 480 instancias de prueba basadas en la literatura, cubriendo múltiples clases y niveles de dificultad, lo que permitió analizar quiebres de escalabilidad, el desempeño por clúster y los trade-offs entre métodos.
- Para el SRPSAP, se adaptaron 36 instancias a partir del conjunto determinístico, incorporando escenarios y ventanas de tiempo, así como factores de control que incluyen distintos tamaños de conjuntos de proyectos, bases, tipos de recursos y horizontes temporales.

Esta estrategia de experimentación permitió comparar, en condiciones controladas, enfoques exactos, metaheurísticos y de descomposición, y extraer conclusiones robustas sobre su idoneidad en función del tamaño del problema y la severidad de la incertidumbre.

5.4. Limitaciones del estudio

Pese a las contribuciones reportadas, la tesis presenta limitaciones que delimitan el alcance de sus resultados y abren oportunidades de mejora:

1. **Ausencia de validación con instancias reales completas.** Aunque las instancias fueron diseñadas para ser realistas y se basan en la literatura, el marco propuesto no ha sido evaluado en conjuntos de datos operacionales integrales (proyectos, recursos, logística y calendarios reales), lo que limita su transferibilidad inmediata.
2. **Escalabilidad de los modelos exactos.** Tanto el modelo MIP determinístico como el modelo estocástico de dos etapas presentan restricciones de escalabilidad al aumentar el número de proyectos, de recursos o de escenarios.
3. **Representación parcial de la complejidad operativa.** Los supuestos del problema y el diseño de instancias no logran capturar completamente fenómenos del mundo real (p. ej., interrupciones de transporte, fallas, prioridades institucionales dinámicas, restricciones ambientales con granularidad temporal fina, correlaciones entre eventos climáticos y duraciones).
4. **Estructuras de costos simplificadas.** Diversos componentes de costo se agregan a los parámetros únicos, lo que reduce el nivel de detalle de los análisis económicos y ambientales.
5. **Rigidez de los algoritmos basados en listas.** Los procedimientos de evaluación de list-scheduling introducen restricciones estructurales (órdenes fijos de bases/recursos o reglas de asignación) que pueden limitar la exploración de soluciones alternativas, en especial en instancias en las que la calidad depende de reasignaciones no triviales.

6. **Configuración paramétrica no optimizada.** El uso de parámetros "clásicos" de la literatura para metaheurísticas aporta comparabilidad, pero no garantiza un desempeño ideal específico para RPSAP/SRPSAP.

7. **Complejidad de la segunda etapa y crecimiento por escenarios**. En SRPSAP, la dificultad de los subproblemas de programación se agrava con el número de escenarios, convirtiéndose en el principal cuello de botella y condicionando la efectividad práctica de la descomposición exacta.

5.5. Líneas de trabajo futuro

A partir de los hallazgos, se proponen líneas de investigación futuras en cuatro direcciones principales.

1. **Extensiones de modelamiento.**

   - **Modelos multiobjetivo** para balancear simultáneamente: beneficio científico/económico, impacto ambiental, riesgo operativo, equidad en la asignación de recursos entre grupos y robustez del plan.
   - **Medidas de riesgo** en la selección bajo incertidumbre y restricciones de confiabilidad.

2. **Nuevos enfoques para programar bajo incertidumbre.** Además del paradigma de dos etapas con escenarios, es pertinente explorar enfoques alternativos y/o complementarios, por ejemplo:

   - **Optimización robusta:** seleccionar y programar considerando conjuntos de incertidumbre para duraciones y/o tiempos de traslado, buscando soluciones viables para todas (o la mayoría) de las realizaciones del conjunto.
   - **Optimización robusta distribucional:** modelar la incertidumbre cuando la distribución es parcialmente conocida (p. ej., intervalos de momentos o divergencias), obteniendo planes menos sensibles a los errores de estimación.
   - **Restricciones probabilísticas (chance constraints):** imponer que ciertas restricciones críticas (ventanas de inicio, disponibilidad de recursos, límites ambientales) se cumplan con alta probabilidad.
   - **Modelos multietapa y horizonte rodante:** permitir decisiones adaptativas (reselección, reprogramación) al revelarse información durante la campaña, aproximando las políticas de decisión y no solo los planes estáticos.
   - **Incertidumbre correlacionada y endógena:** incorporar correlaciones entre duraciones, clima y tiempos de traslado, o considerar que algunas decisiones (p. ej., asignación de bases) afectan la incertidumbre efectiva (endogeneidad).

3. **Mejoras algorítmicas y de descomposición.**

   - **Calibración de parámetros** mediante herramientas como ParamILS (Hutter et al, 2009) o iRace (López-Ibáñez et al, 2016) y/o estrategias bayesianas de configuración automática para adaptar metaheurísticas a familias de instancias RPSAP/SRPSAP.
   - **Descomposiciones más avanzadas:** multicut L-shaped, estabilización (p. ej., regularización/level sets), cortes más fuertes (Pareto-optimal cuts), selección y gestión de cortes, paralelización por escenarios y estrategias híbridas de ramificación.

- **Matheurísticas y grandes vecindarios:** integrar MIP parcial para subestructuras (p. ej., reasignación local de recursos y bases) en una búsqueda metaheurística (Large Neighborhood Search, Fix-and-Optimize, etc.).

4. **Metaheurísticas híbridas y enfoques basados en aprendizaje.**

   - **Metaheurísticas híbridas** que combinen fortalezas complementarias (p. ej., ILS para intensificación, SA para diversificación y operadores evolutivos para recombinación).
   - **Metaheurísticas basadas en el aprendizaje automático:** selección adaptativa de vecindarios, aprendizaje de políticas de aceptación, predicción de componentes críticos (cuellos de botella) y/o aprendizaje por refuerzo para guiar la construcción de listas y asignaciones.
   - **Generación automática de algoritmos:** explorar hiperheurísticas y enfoques de diseño automatizado que construyan (o configuren) estrategias de búsqueda específicamente para familias de instancias RPSAP/SRPSAP.

5. **Validación aplicada y transferencia a otros contextos extremos.**

   - **Aplicación del marco a instancias reales** en colaboración con actores del ecosistema antártico (p. ej., INACH), incluyendo datos sobre recursos, logística y calendarios de campaña.
   - **Adaptación a otros contextos extremos** con restricciones severas y alta incertidumbre, tales como la planificación de proyectos espaciales, misiones científicas submarinas o campañas en desiertos polares, preservando la estructura de selección–programación con recourse.

5.6. Cierre

En conjunto, la tesis establece un marco integrado de modelamiento determinístico y estocástico para la planificación de proyectos científicos en múltiples bases, complementado con algoritmos metaheurísticos, métodos de descomposición y esquemas híbridos que permiten su aplicación en escalas relevantes. Los resultados sostienen que no existe un método universalmente dominante; más bien, la elección depende del equilibrio requerido entre la precisión y el tiempo de cómputo. En este sentido, la tesis ofrece un repertorio metodológico coherente que puede utilizarse tanto para la investigación futura como para el diseño de herramientas de apoyo a la toma de decisiones en operaciones antárticas bajo severas restricciones logísticas y ambientales.

Bibliografía

ATS (1959) *Antarctic Treaty*. Secretariat of Antarctic Treaty, Buenos Aires, Argentina

COMNAP (2017) *Antarctic Station Catalogue*. Council of Managers of National Antarctic Programs, Christchurch, New Zealand

Hutter F, Hoos HH, Leyton-Brown K, Stützle T (2009) ParamILS: an automatic algorithm configuration framework. *Journal of Artificial Intelligence Research* 36(1):267–306, DOI https://doi.org/10.1613/jair.2861

Kennicutt M, Chown S, Cassano J, Liggett D, Peck L, Massom R, Rintoul S, Storey J, Vaughan D, Wilson T, et al (2015) A roadmap for Antarctic and Southern Ocean science for the next two decades and beyond. *Antarctic Science* 27(1):3–18, DOI https://doi.org/10.1017/S0954102014000674

Kennicutt MC, Chown SL, Cassano JJ, Liggett D, Massom R, Peck LS, Rintoul SR, Storey JWV, Vaughan DG, Wilson TJ, Sutherland WJ (2014) Polar research: Six priorities for Antarctic science. *Nature* 512(7512):23–25, DOI https://doi.org/10.1038/512023a

Liu X, Liang J, Zhang Zh, Yang S, Peukert S, Lanza G (2025) Project selection and scheduling with multiplicative enhancement effects and delay risk: An application in intelligent manufacturing technologies. *IISE Transactions* 57(8):873–889, DOI 10.1080/24725854.2024.2374090

López-Ibáñez M, Dubois-Lacoste J, Pérez Cáceres L, Birattari M, Stützle T (2016) The irace package: Iterated racing for automatic algorithm configuration. *Operations Research Perspectives* 3:43–58, DOI https://doi.org/10.1016/j.orp.2016.09.002

Seroussi H (2019) Fate and future climatic role of polar ice sheets. *Nature* 566:48–49, DOI https://doi.org/10.1038/d41586-019-00330-7

Tofighian AA, Naderi B (2015) Modeling and solving the project selection and scheduling. *Computers & Industrial Engineering* 83:30–38, DOI https://doi.org/10.1016/j.cie.2015.01.012

Vega-Hidalgo M, Pradenas L, Parada V (2026) Optimizing research in Antarctica: a novel approach to project selection and scheduling across multiple stations. *Journal of the Operational Research Society* 77(1):286–312, DOI https://doi.org/10.1080/01605682.2025.2495766