Revista Científica Estelí
Medio ambiente, tecnología y desarrollo humano
Universidad Nacional Autónoma de Nicaragua, UNAN-Managua
ISSN: 3006-6700

180

https://revistas.unan.edu.ni/index.php/Cientifica
DOI: https://doi.org/10.5377/esteli.v14i55.21306

Revisión del estado del arte de la Optimización Multiobjetivo en Proyectos Ágiles de Desarrollo de Software

Review of the State of the Art in Multi-Objective Optimization in Agile Software Development Projects

Erick Noel Lanzas Martínez
Universidad Nacional Autónoma de Nicaragua, Centro Universitario Regional de Matagalpa. UNAN-Managua/CUR-Matagalpa, Nicaragua.
https://orcid.org/0000-0003-2557-2833
elanzas@unan.edu.ni

RECIBIDO
08/04/2025

ACEPTADO
22/09/2025

RESUMEN

En la gestión de proyectos de software ágiles, es común enfrentar decisiones complejas con múltiples objetivos en conflicto, tales como minimizar el tiempo de entrega y el costo, al tiempo que se maximiza la calidad del producto y la satisfacción de los interesados. La optimización multiobjetivo ofrece un enfoque sistemático para equilibrar estos criterios contrapuestos, buscando soluciones de compromiso (óptimas de Pareto) en lugar de un único plan óptimo. En este artículo se presenta una revisión del estado del arte (período 2020-2025) sobre la aplicación de técnicas de optimización multiobjetivo en proyectos ágiles de desarrollo de software. Las Bases de datos revisadas fueron principalmente IEEE, ACM, Springer, Elsevier, Arxiv. Se analizaron, de forma sistemática, los algoritmos evolutivos más usados fueron: NSGA-II, NSGA-III, SPEA2, MOEA/D, algunas metaheurísticas de enjambre y métodos exactos como la programación entera y por metas. Se identificaron sus aplicaciones más comunes en entornos ágiles (planificación de versiones, asignación de tareas y priorización de requisitos), así como las herramientas más citadas y los principales desafíos: incertidumbre, escalabilidad, preferencias de los interesados y adopción industrial. Los resultados indican que los algoritmos evolutivos multiobjetivo, como NSGA-II y variantes, dominan la literatura reciente, aplicándose exitosamente a problemas como el Next Release Problem y la calendarización de proyectos, aunque surgen enfoques híbridos y exactos para mejorar la calidad de las soluciones. En conclusión, la optimización multiobjetivo se consolida como un marco riguroso para apoyar la toma de decisiones en proyectos ágiles, con beneficios demostrados en casos de estudio.

PALABRAS CLAVE

Optimización multiobjetivo; software; proyectos ágiles; planificación; metaheurísticas.

ABSTRACT

In agile software project management, it is common to face complex decisions with multiple conflicting objectives, such as minimizing delivery time and cost while maximizing product quality and stakeholder satisfaction. Multi-objective optimization offers a systematic approach to balancing these conflicting criteria, seeking compromise solutions (Pareto optimal) rather than a single optimal plan. This article presents a review of the state of the art (2020-2025) on the application of multi-objective optimization techniques in agile software development projects. The databases reviewed were mainly IEEE, ACM, Springer, Elsevier, and Arxiv. The most used evolutionary algorithms were systematically analyzed: NSGA-II, NSGA-III, SPEA2, MOEA/D, some swarm metaheuristics, and exact methods such as integer and goal programming. Their most common applications in agile environments (version planning, task assignment, and requirement prioritization) were identified, as well as the most cited tools and the main challenges: uncertainty, scalability, stakeholder preferences, and industrial adoption. The results indicate that multi-objective evolutionary algorithms, such as NSGA-II and variants, dominate recent literature, being successfully applied to problems such as the Next Release Problem and project scheduling, although hybrid and exact approaches are emerging to improve the quality of solutions. In conclusion, multi-objective optimization is consolidating itself as a rigorous framework to support decision-making in agile projects, with proven benefits in case studies.

KEYWORDS

Multi-objective optimization; software; agile projects; planning; metaheuristics.

INTRODUCCIÓN

En la gestión de proyectos de desarrollo de software, es frecuente enfrentarse a decisiones complejas que implican múltiples objetivos en conflicto. Los gerentes de proyecto deben equilibrar criterios como el tiempo de entrega, el costo de desarrollo, la calidad del producto (por ilustrar, número de defectos) e incluso la satisfacción de los interesados. Tradicionalmente, esta problemática se ha abordado mediante técnicas de planificación que reducen estos factores a un único criterio (como el costo total) o imponiendo restricciones (como un presupuesto fijo).

Sin embargo, en los entornos reales, optimizar un solo objetivo suele ser insuficiente porque las decisiones de planificación conllevan *trade-offs* inevitables entre objetivos contrapuestos (Amaral & Elias, 2018). En este contexto, la optimización multiobjetivo proporciona un enfoque sistemático para abordar decisiones con múltiples criterios. A diferencia de la optimización tradicional donde se busca optimizar un solo objetivo, la optimización multiobjetivo busca conjuntos de soluciones óptimas que representen diferentes equilibrios entre los objetivos en conflicto, conocidas como *soluciones Pareto óptimas*. Ninguna de estas soluciones es estrictamente mejor que las otras en todos los criterios, sino que cada una ofrece un compromiso distinto. Este enfoque es especialmente valioso en ingeniería de software, donde abundan problemas de decisión con criterios múltiples (Zhang, Harman, & Mansouri, 2007).

En los últimos años, la Ingeniería de Software Basada en Búsqueda (Search-Based Software Engineering, SBSE) ha promovido el uso de técnicas de optimización, particularmente metaheurísticas, para abordar desafíos complejos de software (como planificación de proyectos, asignación de requisitos, pruebas) tratándolos como problemas de búsqueda en espacios de soluciones.

En la literatura de las últimas dos décadas (Zitzler & Künzli, Indicator-Based Selection in Multiobjective Search, 2004), (Zhang, Harman, & Mansouri, 2007), (Amaral & Elias, 2018)), los algoritmos evolutivos multiobjetivo han destacado como herramienta eficaz para abordar estos problemas. Técnicas como NSGA-II, SPEA2, MOEA/D, entre otras, han permitido aproximar conjuntos de soluciones de Pareto en problemas de planificación de proyectos con relativo éxito. No obstante, existen también enfoques exactos basados en programación entera multiobjetivo o programación por metas que han sido aplicados para encontrar soluciones óptimas o semi-óptimas a problemas de menor escala (Dong, Xue, Brinkkemper, & Li, 2022).

Dentro de SBSE, muchas dificultades se enmarcan de forma natural como problemas multiobjetivo, ya que se deben optimizar simultáneamente varias métricas de desempeño del proyecto. De hecho, la planificación de versiones (también conocida como el Next Release Problem, NRP) implica decidir qué características o requisitos incluir en la siguiente versión de un software, buscando maximizar la satisfacción de los clientes o el valor del negocio a la vez que se minimizan el esfuerzo o costo de desarrollo (Amaral & Elias, 2018).

De forma similar, la calendarización de proyectos de software (planificación de tareas y asignación de personal) requiere asignar desarrolladores a tareas a lo largo del tiempo procurando terminar lo antes posible, sin exceder el presupuesto y evitando sobrecargar al equipo. En este contexto, Schwaber & Sutherland (2020) describe que Scrum estructura este trabajo en un Scrum Team formado por un Product Owner, un

Scrum Master y desarrolladores. Las tareas se gestionan en Sprints con eventos formales como la Planificación del Sprint, el Daily Scrum, la Revisión y la Retrospectiva, así como artefactos como el Product Backlog, el Sprint Backlog y el Incremento.

En este tipo de metodologías, estas decisiones deben revisarse en iteraciones cortas, adaptándose a cambios de requisitos o recursos disponibles. Esto introduce una capa adicional de complejidad, ya que el plan óptimo puede cambiar con el tiempo y es necesario mantener un equilibrio entre criterios en cada replanteamiento, además de atender la mejora continua promovida por las retrospectivas al final de cada Sprint (SCRUM.ORG, 2025).

Por otra parte, se han propuesto métodos híbridos (como memetic algorithms (Shen, Minku, Marturi, Guo, & Han, 2018), algoritmos evolutivos con aprendizaje) y enfoques novedosos de inteligencia colectiva o bio-inspirados (como variantes de optimización por enjambre de partículas, colonias de hormigas, optimización basada en búhos o buitres (Asha, Rajesh, Verma, & Poonguzhali, 2023)) para mejorar la eficiencia o incorporar consideraciones adicionales (como riesgo, incertidumbre o equidad entre partes interesadas).

Este artículo pretende servir como base y motivación para continuar explorando y refinando la optimización multiobjetivo en el apasionante ámbito de la gestión de proyectos de software. También pretende dar claridad y contexto sobre este tema. Desde el plano teórico, organiza y sintetiza los conceptos y técnicas más recientes, mostrando cómo algoritmos evolutivos, enfoques exactos e híbridos se han aplicado al Next Release Problem, la priorización de requisitos y la asignación de recursos. Al reunir esta información dispersa, ofrece a investigadores y estudiantes una base comprensible para profundizar en el área y comparar enfoques con criterios comunes.

Desde el plano metodológico y práctico, la revisión describe de forma reproducible la estrategia de búsqueda y selección de estudios, identifica tendencias y vacíos y señala los retos para transferir estas técnicas a la industria. Con ello busca que, gestores y desarrolladores conozcan las posibilidades reales de la optimización multiobjetivo y cuenten con un punto de partida para integrar herramientas y enfoques en sus prácticas ágiles cotidianas, facilitando decisiones más informadas y equilibradas.

En este artículo se presenta una revisión del estado del arte (período 2020-2025) sobre la aplicación de técnicas de optimización multiobjetivo en proyectos ágiles de desarrollo de software. Se cubren de forma amplia múltiples algoritmos y estrategias, en lugar de centrarse en una técnica particular, e incluyendo las aplicaciones frecuentes documentadas (por ejemplo, planificación de versiones, priorización de requisitos, asignación de tareas en equipos ágiles). También se identifican las herramientas de software comúnmente utilizadas para implementar estos enfoques y se discuten los desafíos actuales y tendencias emergentes en la investigación del área. Para lograr lo anterior, se realizó una búsqueda sistemática de literatura reciente en bases de datos reconocidas (IEEE, ACM, Springer, Elsevier, Arxiv, entre otras):

MATERIALES Y MÉTODOS

Tipo de investigación

Para realizar esta revisión se llevó a cabo una investigación de tipo documental donde se siguió un enfoque metodológico sistemático, basado en las directrices propuestas por Kitchenham & Charters (2007) para revisiones sistemáticas en ingeniería de software, complementado con criterios de análisis narrativo. Su sustento teórico radica en que las revisiones sistemáticas permiten identificar, evaluar y sintetizar de manera reproducible la evidencia disponible sobre un tema ofreciendo así un panorama integral y crítico.

Universo y muestra

La búsqueda bibliográfica inicial identificó aproximadamente 150 documentos (que constituyeron el universo) entre artículos, ponencias y literatura gris relacionados con optimización multiobjetivo aplicada a proyectos de software. Tras aplicar los criterios de inclusión y exclusión descritos más abajo, se seleccionaron 28 estudios para el análisis final (muestra).

Criterios de selección de los estudios

Se incluyeron publicaciones:

- En inglés o español.
- Que abordaran explícitamente problemas de optimización con múltiples objetivos en algún aspecto de la gestión de proyectos de software (planificación temporal, asignación de recursos, selección de requisitos, optimización de procesos).
- Publicadas en el período 2020-2025, salvo trabajos seminales ampliamente citados, o de primera línea.
- Se excluyeron fuentes no académicas (artículos de opinión, páginas web populares, Wikipedia) para asegurar la confiabilidad. También trabajos que se alejaran demasiado de la temática principal.

Técnicas e instrumentos de recolección de datos

Se utilizaron las siguientes técnicas e instrumentos de recolección de datos:

- Bases de datos: IEEE Xplore, ACM Digital Library, Scopus, arXiv, Google Académico, repositorios institucionales de la UNAN Managua y bibliotecas universitarias de acceso abierto, específicamente el RIUMA de UNAN Managua.
- Términos de búsqueda: combinaciones en inglés y español de “multi-objective optimization”, “software project scheduling”, “multiobjective software engineering”, “optimización multi-criterio en software”, etc.
- Instrumento de registro y análisis: hoja de cálculo en Microsoft Excel para tabular autor, año, contexto del problema, algoritmos empleados y hallazgos principales. Microsoft Word para elaborar el informe y redactar resúmenes críticos, así como su gestor bibliográfico incluido.

Etapas de la investigación

Etapa 1. Definición de la estrategia de búsqueda

En primer lugar, se definió una estrategia de búsqueda bibliográfica orientada a identificar estudios pertinentes sobre optimización multiobjetivo aplicada a proyectos de software. Se delimitaron como fuentes de información las bases de datos académicas de alto impacto en el área de informática y ciencias de la computación: IEEE Xplore, ACM Digital Library, Scopus, arxiv y Google Académico. También se consultaron repositorios institucionales y bibliotecas universitarias para obtener tesis doctorales y literatura gris relevante. Los términos de búsqueda incluyeron combinaciones en inglés y español de palabras clave como "multi-objective optimization", "software project", "software project scheduling", "multiobjective software engineering", "búsqueda multiobjetivo proyectos software", "optimización multi-criterio en software", etc.

Etapa 2. Recopilación inicial de documentos

Se aplicaron filtros de selección para garantizar la calidad y relevancia de las fuentes. Principalmente se consideraron publicaciones recientes (período 2015 - 2025**)** para reflejar el estado del arte actual, aunque también se incluyeron trabajos seminales más antiguos ampliamente citados que sentaron las bases del campo (por ejemplo, algoritmos fundamentales de optimización multiobjetivo). Se excluyeron fuentes no académicas (artículos de opinión, páginas web populares, Wikipedia) para asegurar la confiabilidad de la información.

Cada resultado de búsqueda fue evaluado mediante sus resúmenes y palabras clave, seleccionando aquellos estudios que abordaban explícitamente problemas de optimización con múltiples objetivos en algún aspecto de la gestión de proyectos de software (planificación temporal, asignación de recursos, selección de requisitos, optimización de procesos).

Etapa 3. Extracción y tabulación de datos

La revisión siguió un proceso iterativo de lectura y síntesis. Inicialmente, se recopiló un conjunto amplio de referencias relevantes. Posteriormente, mediante lectura crítica, se extrajeron de cada fuente los objetivos del estudio, el contexto del problema de proyecto de software abordado, los algoritmos o métodos de optimización empleados y los hallazgos o conclusiones principales. Se llevó un registro sistemático de esta información, tabulándola para facilitar comparaciones. En particular, se confeccionaron tablas de síntesis para categorizar los algoritmos y aplicaciones identificados en la literatura. Estas tablas permitieron identificar patrones, similitudes y diferencias entre enfoques, así como vacíos de investigación o tendencias emergentes.

Se presenta un ejemplo de las tablas síntesis utilizadas:

Tabla X:
Ejemplo de tabla de síntesis de algoritmos y aplicaciones identificados en la literatura revisada

Algoritmo/ Técnica	Tipo de enfoque	Características y usos relevantes	Referencias representativas
NSGA-II (Deb et al.)	Evolutivo (Genético)	Algoritmo genético de ranking no-dominado. Muy usado como baseline; buena diversidad y convergencia. Aplicado en NRP, planificación de proyectos.	(Deb, Pratap, Agarwal, & Meyarivan, 2002) (Rahimi, Gandomi, Nikoo, & Chen, 2023)

Nota: Elaboración propia a partir de los estudios seleccionados.

Adicionalmente, se empleó un enfoque inductivo-deductivo durante el análisis. De manera inductiva, a partir de los datos recopilados se identificaron temas recurrentes. Luego, en forma deductiva, se contrastaron estos hallazgos con el marco teórico existente, verificando su coherencia con principios conocidos de optimización y gestión de proyectos. Este proceso garantizó una cobertura integral del tema, combinando la amplitud (cubrimiento de numerosas fuentes) con la profundidad de análisis (discusión crítica de enfoques y resultados).

Etapa 4. Síntesis y categorización

Cabe destacar que, dado el enfoque documental de esta investigación, no se realizó trabajo experimental propio; los resultados y discusión se basan en la integración de resultados reportados por otros autores. No obstante, se procuró evaluar la calidad de cada estudio incluido (considerando, por ejemplo, si publicaron en revistas o conferencias prestigiosas, número de citas, rigor metodológico, etc.) para dar mayor peso en la síntesis a aquellos de alta confiabilidad. En la sección siguiente, se presentan los hallazgos de la revisión, organizados según los ejes temáticos identificados: algoritmos, aplicaciones y tendencias.

A partir de la literatura analizada, se identificaron cuatro componentes principales para estructurar la discusión: **(**1) Algoritmos y estrategias de optimización multiobjetivo aplicadas a proyectos de software, (2) Aplicaciones frecuentes en la gestión de proyectos ágiles (problemas específicos abordados), (3) Herramientas utilizadas para implementar y evaluar estas optimizaciones, y (4) Desafíos actuales y tendencias emergentes en este campo de investigación.

RESULTADOS Y DISCUSIÓN

A continuación, se desarrolla cada componente con apoyo de ejemplos representativos extraídos de estudios recientes. Se incluyen tablas resumidas para brindar una visión comparativa de los algoritmos (Tabla 1) y de las aplicaciones abordadas (Tabla 2) en los trabajos revisados.

1. Algoritmos y Estrategias de Optimización Multiobjetivo

Algoritmos Evolutivos Multiobjetivo (MOEAs). La técnica predominante en la literatura reciente son los algoritmos evolutivos de optimización multiobjetivo, derivados de la inteligencia evolutiva (genética) y bio-inspirada. Estos algoritmos buscan aproximar el frente de Pareto (conjunto de soluciones no-dominadas) en una sola ejecución, manteniendo una población de soluciones. Entre los MOEAs más utilizados se encuentran NSGA-II (Non-dominated Sorting Genetic Algorithm II) y su sucesor NSGA-III, así como

SPEA2 (Strength Pareto Evolutionary Algorithm 2), MOEA/D (Algoritmo evolutivo multiobjetivo basado en descomposición) y algoritmos basados en indicadores como IBEA (Indicator-Based EA).

Por ejemplo, NSGA-II (propuesto por (Deb, Pratap, Agarwal, & Meyarivan, 2002)) ha sido empleado en numerosos estudios como base para resolver el Next Release Problem y variantes de planificación de proyectos, gracias a su equilibrio entre eficacia y simplicidad (Rahimi, Gandomi, Nikoo, & Chen, 2023).

En un estudio comparativo reciente, Rahimi et al. (2023) evaluaron el desempeño de varios algoritmos evolutivos (incluyendo NSGA-II, NSGA-III, MOEA/D, entre otros) para el problema de planificación de la próxima versión, encontrando que NSGA-II y MOEA/D ofrecían una buena diversidad de soluciones, mientras que algoritmos basados en indicadores podían mejorar en métricas específicas de calidad (como hipervolumen).

Adicionalmente, NSGA-III ha sido aplicado cuando el número de objetivos es alto (cuatro o más), ya que introduce un conjunto de puntos de referencia para mantener la diversidad en problemas de muchos-objetivos (por ejemplo, planificación de proyectos considerando costo, tiempo, calidad, riesgo y satisfacción simultáneamente). SPEA2 e IBEA también aparecen en la literatura como alternativas; por ejemplo, Dukhan, Mohamed, Amer, Zanaty, & Reyad (2022) emplean SPEA2 en combinación con otras heurísticas para la selección de requisitos óptima, argumentando ventajas en la convergencia hacia soluciones de alto valor para el cliente.

La tabla 1 resume algunos de los algoritmos relevantes en esta categoría, indicando su naturaleza y aplicaciones destacadas.

Metaheurísticas basadas en enjambre y otras heurísticas. Aunque en menor medida que los algoritmos genéticos, otras metaheurísticas multiobjetivo han sido exploradas. La Optimización por *Enjambre de Partículas multiobjetivo (MOPSO)* y la Optimización basada en Colonias de Hormigas multiobjetivo se han aplicado ocasionalmente a problemas de calendarización y asignación de recursos en proyectos de software. Por ejemplo, algoritmos híbridos GA-*PSO* se han propuesto para asignación de tareas, aprovechando la rápida convergencia de PSO con la diversidad del GA (Sharma, Gaur, & Mittal, 2014).

También han surgido algoritmos bio-inspirados novedosos adaptados al contexto de requisitos de software: (Liu, Zhou, Luo, & Wei, 2024) introducen un algoritmo de “buitres africanos” multiobjetivo cuántico (QMO-AVOA) para la selección de requisitos, reportando que supera a NSGA-II y MOEA/D en ciertas instancias de NRP al mejorar el equilibrio entre exploración y explotación del espacio de soluciones. Estas aproximaciones exóticas (inspiradas en comportamientos animales) reflejan la tendencia de buscar optimizadores personalizados para problemas de software, aunque su adopción no es aún generalizada.

Otra línea relacionada son los *algoritmos meméticos* (evolutivos con búsqueda local incorporada). (Shen, Minku, Marturi, Guo, & Han, 2018) propusieron un algoritmo memético con aprendizaje por refuerzo (Q-Learning) para un problema de calendarización de proyectos dinámico con múltiples objetivos, demostrando que la incorporación de búsqueda local adaptativa podía mejorar la calidad del frente de Pareto encontrado, especialmente en entornos cambiantes. En general, las metaheurísticas de enjambre y los enfoques híbridos se usan para casos donde se busca aprovechar heurísticas particulares del problema (por ejemplo, reglas de prioridad para asignar tareas, o conocimiento sobre dependencias entre requisitos) dentro del marco evolutivo multiobjetivo.

Programación entera multiobjetivo y métodos exactos. Aunque la mayoría de los problemas en proyectos de software son NP-duros y de gran tamaño (lo que dificulta el uso de métodos exactos), algunos trabajos han explorado enfoques de optimización exacta multiobjetivo. Dong, Xue, Brinkkemper, & Li (2022) desarrollaron modelos de programación entera multi-objetivo para el Next Release Problem, capaces de encontrar el frente de Pareto completo en instancias pequeñas a medianas. Para lograrlo, emplean técnicas de mejora de eficiencia en la exploración completa del espacio de soluciones, como algoritmos de branch-and-bound con podas informadas por estimaciones multiobjetivo. Estos enfoques garantizan encontrar soluciones óptimas en sentido de Pareto, pero escalan pobremente cuando el número de requisitos o decisiones crece.

Otra técnica exacta aplicada es la programación por metas (*goal programming*), que transforma un problema multiobjetivo en uno de satisfacción de metas con diferentes prioridades o ponderaciones Kaur, Singh, Anand, & Agarwal (2023) aplican programación por metas al resource allocation en proyectos ágiles, utilizando tres variantes: (a) método de suma ponderada (programación por metas Archimediana), (b) método lexicográfico (metas preemptivas) y (c) método Chebyshev (minimax de desviaciones). Con estas técnicas lograron obtener soluciones factibles que satisfacen en la mayor medida posible los objetivos de minimizar costo, minimizar retraso y maximizar uso de capacidad en un contexto de asignación de personal a tareas.

La ventaja de los enfoques de programación exacta es que ofrecen soluciones óptimas o cuasi-óptimas con certidumbre sobre su optimalidad; sin embargo, su uso práctico está limitado a problemas de menor escala o requiere reformular el problema (por ejemplo, relajando ciertos requerimientos o considerando versiones agregadas).

En la práctica, es común que estudios comparativos combinen métodos exactos con heurísticos: por ejemplo, en planificación de versiones, se ha comparado la eficiencia de un modelo MILP multiobjetivo resuelto con CPLEX frente a algoritmos genéticos, observando que los genéticos encuentran buenas soluciones con menor tiempo, mientras que el modelo MILP puede validar la calidad de dichas soluciones en casos pequeños (Rahimi et al., 2023; Dong et al., 2022).

Otras estrategias y enfoques de búsqueda. Además de los anteriores, existen enfoques basados en búsqueda local multiobjetivo (como Pareto Local Search), algoritmos de recocido simulado multiobjetivo o búsqueda tabú multiobjetivo. No se hallaron en la literatura muy recientes aplicaciones directas de recocido o tabú multiobjetivo en proyectos ágiles, posiblemente debido a que suelen requerir un diseño cuidadoso de funciones de vecindad en espacios combinatoriales complejos. Sin embargo, algunos estudios más antiguos demostraron que variantes de recocido simulado podían generar aproximaciones de Pareto para la selección de requisitos, aunque con menor eficiencia que los algoritmos evolutivos (Zhang et al., 2007).

Finalmente, cabe mencionar los enfoques de múltiples fases o heurísticas constructivas multiobjetivo: por ejemplo, algoritmos que primero construyen una solución factible inicial (pudiendo usar heurísticas golosas) y luego aplican una mejora multiobjetivo. Un caso es el de branch-and-bound con criterio de Pareto, aplicado en combinación con heurísticas fuzzy para priorización de requisitos (Dukhan, Mohamed, Amer, Zanaty, & Reyad, 2022). Estos enfoques combinados buscan aprovechar la rapidez de métodos heurísticos con la mejoría incremental guiada por múltiples objetivos.

La tabla 1 presenta un compendio de los principales algoritmos y técnicas de optimización multiobjetivo identificados, junto con su naturaleza (evolutivo, exacto, híbrido, etc.) y ejemplos de aplicaciones reportadas en proyectos de software.

Tabla 1:
Algoritmos y enfoques de optimización multiobjetivo aplicados a proyectos de software ágiles.

Algoritmo/ Técnica	Tipo de enfoque	Características y usos relevantes	Referencias representativas
NSGA-II (Deb et al.)	Evolutivo (Genético)	Algoritmo genético de ranking no-dominado. Muy usado como baseline; buena diversidad y convergencia. Aplicado en NRP, planificación de proyectos.	(Deb, Pratap, Agarwal, & Meyarivan, 2002) (Rahimi, Gandomi, Nikoo, & Chen, 2023)
NSGA-III	Evolutivo (Genético)	Variante para muchos-objetivos (>3). Mantiene diversidad mediante puntos de referencia. Usado en casos con múltiples métricas (p.ej. añadir riesgo, satisfacción, etc.).	(Deb & Jain, 2014) (Shen, Minku, Marturi, Guo, & Han, 2018)
SPEA2	Evolutivo (Genético)	Utiliza un archivo externo para conservar Pareto; apto para mantener diversidad. Empleado en selección de requisitos híbrida.	(Zitzler, Laumanns, & Thiele, SPEA2: Improving the strength pareto evolutionary algorithm, 2001) (Dukhan, Mohamed, Amer, Zanaty, & Reyad, 2022)
MOEA/D	Evolutivo (Descomposición)	Descompone el problema en múltiples subproblemas escalarizados (p. ej., combinación lineal de objetivos). Demostró buen desempeño en calendarización ágil.	(Zhang, Harman, & Mansouri, 2007) (Zapotecas-Martínez & Ponsich, 2020)
IBEA	Evolutivo (Indicadores)	Optimiza directamente un indicador (ej. hipervolumen). Útil para controlar la calidad del frente Pareto. Aplicado en algunos estudios comparativos (p.ej., NRP).	(Zitzler & Künzli, Indicator-Based Selection in Multiobjective Search, 2004); (Rahimi, Gandomi, Nikoo, & Chen, 2023)
Algoritmos de Enjambre (MOPSO, MO-ACO)	Metaheurístico (Enjambre)	Extensiones multiobjetivo de PSO y ACO. Menos comunes, pero usados en asignación de tareas con restricciones, a veces combinados con GA.	(Kumari, Singh, & Ranjan, 2019) (Liu & Dai, 2020)
Algoritmos bio-inspirados recientes (AVOA, GWO)	Metaheurístico (Bio-inspirado)	Nuevos algoritmos basados en comportamientos animales (buitres, lobos, etc.) adaptados a múltiples objetivos. Buscan mejorar exploración/ explotación.	(Liu, Zhou, Luo, & Wei, 2024) (Asha, Rajesh, Verma, & Poonguzhali, 2023)

Algoritmo/ Técnica	Tipo de enfoque	Características y usos relevantes	Referencias representativas
Meméticos / Híbridos	Evolutivo + Búsqueda local	Combinan MOEAs con heurísticas locales o aprendizaje. Excelentes para problemas dinámicos o con estructuras especiales (p.ej., Q-learning memético en scheduling).	(Shen, Minku, Marturi, Guo, & Han, 2018); (Tong, Minku, Menzel, Sendhoff, & Yao, 2021)
Programación Entera Multiobjetivo	Exacto (MILP)	Formulación matemática con múltiples funciones objetivo (resueltas típicamente via epsilon-restricciones o métodos de Pareto óptimo). Garantiza Pareto óptimo en instancias pequeñas.	(Dong, Xue, Brinkkemper, & Li, 2022)
Programación por Metas	Exacto (GP)	Transforma objetivos en metas con prioridades o pesos. Obtiene una solución de compromiso según preferencias predefinidas. Usado en asignación de recursos con objetivos de costo/tiempo.	(Kaur, Singh, Anand, & Agarwal, 2023)
Búsqueda Local Pareto	Heurístico (Local)	Extensión de búsqueda local guiada a optimizar varios objetivos. Requiere vecindarios bien diseñados. Pocas aplicaciones recientes directas; a veces integrada dentro de meméticos.	(Govil & Sharma, 2021)
Recocido Simulado Multiobjetivo	Heurístico (Metaheurístico)	Enfría gradualmente aceptando soluciones peores probabilísticamente para escapar óptimos locales, adaptado a multiobjetivo (con criterio de dominancia). Menos frecuente en literatura reciente.	(Zhang, Harman, & Mansouri, 2007)
Enfoques híbridos por fases	Combinado	Por ejemplo, heurística golosa inicial + mejora multiobjetivo (GA o Búsqueda Local). Aprovechan conocimiento problema en construcción y optimización posterior.	(Ferrucci, Harman, Ren, & Sarro, 2013) (García-Nájera, Zapotecas-Martínez, Falcón-Cardona, & Cervantes, 2021)

Nota: Las referencias listadas son ilustrativas, combinando trabajos clásicos (pre-2019) y recientes (2020–2025) que documentan el uso de cada enfoque.

2. Aplicaciones Frecuentes en Proyectos Ágiles de Software

En el dominio de proyectos ágiles, los problemas de decisión donde se ha aplicado optimización multiobjetivo típicamente corresponden a planificación y gestión de alcance/ recursos. A continuación, se describen las aplicaciones más reportadas, junto con los objetivos considerados y enfoques utilizados en cada caso. La tabla 2 sintetiza estos problemas, objetivos y técnicas asociadas.

2.1 Planificación de Versiones (Next Release Problem)

El Next Release Problem (NRP) es quizá la aplicación más estudiada de optimización multiobjetivo en ingeniería de requisitos. En el contexto ágil, la planificación de versiones o releases consiste en seleccionar, de entre un conjunto de historias de usuario o requisitos pendientes, cuáles se implementarán en la próxima iteración o versión del producto. Esto debe hacerse equilibrando objetivos como maximizar el valor para el cliente o la satisfacción de los stakeholders (a menudo medido en puntuaciones de importancia o valor de negocio) y minimizar el esfuerzo o costo de desarrollo, bajo restricciones de presupuesto o capacidad del equipo. Trabajos clásicos reformularon el NRP como un problema bi-objetivo (valor vs. costo) resuelto con algoritmos evolutivos, obteniendo un conjunto de posibles planes de lanzamiento que representan distintos trade-offs (Amaral & Elias, 2018).

En años recientes, se han enriquecido estos modelos con objetivos adicionales y consideraciones propias de entornos ágiles:

- **Riesgos y Calidad:** Amaral y Elias (2018) añadieron la consideración de riesgo del software como un tercer factor, proponiendo un enfoque evolutivo multiobjetivo riesgo-sensible. Su algoritmo (un GA multiobjetivo) incorporó un análisis de riesgo para estimar el impacto potencial de cada requisito en el costo y la satisfacción, entregando así soluciones que balancean valor, costo y riesgo. En su experimento, observaron que ignorar los riesgos podía conducir a planes subóptimos (por ejemplo, incluir muchas funcionalidades de alto riesgo), mientras que su enfoque multiobjetivo generó planes más robustos.

- **Satisfacción de múltiples interesados**: Varios autores han notado que el "valor" no es uniforme, sino que diferentes interesados valoran distintos subconjuntos de requisitos. Por ello, se han formulado variantes multiobjetivo del NRP donde se consideran por separado las satisfacciones de diferentes grupos de usuarios o stakeholders. (Rahimi, Gandomi, Nikoo, & Chen, 2023) propusieron un modelo con objetivos de maximizar satisfacción para dos perfiles de clientes distintos, además de minimizar costo total, usando NSGA-II para obtener frentes de Pareto que muestren, por ejemplo, cómo mejorar la satisfacción de un segmento de usuarios puede reducir la de otro si el presupuesto es fijo.

- **Justicia o equidad:** Un tema emergente es la equidad en la selección de requisitos, intentando que las decisiones sean "justas" entre los distintos interesados Asha, Rajesh, Verma, & Poonguzhal (2023) introducen como objetivo explícito minimizar la disparidad de satisfacción entre stakeholders, junto con los objetivos tradicionales de valor promedio y costo. Utilizaron algoritmos bio-inspirados (optimización de enjambre y de lobos grises multiobjetivo) y reportaron que, al incluir este objetivo de equidad, las soluciones ofrecían compromisos más balanceados en donde ningún grupo de interesados quedaba sistemáticamente relegado.

- **Formulaciones exactas:** Además de los enfoques heurísticos, se han aplicado modelos exactos al NRP multiobjetivo. Por ejemplo, Dong, Xue, Brinkkemper, & Li (2022) presentan modelos de programación entera que encuentran todas las soluciones Pareto-óptimas para conjuntos de requisitos de tamaño moderado (decenas de requisitos). Asimismo, Dukhan, Mohamed, Amer, Zanaty, & Reyad, (2022) combinaron métodos de *branch-and-bound* con lógica difusa para manejar incertidumbre en

estimaciones de esfuerzo, obteniendo soluciones óptimas en instancias pequeñas del NRP.

La planificación de versiones sigue siendo un problema central donde la optimización multiobjetivo ha demostrado su utilidad. Objetivos típicos incluyen valor/satisfacción (a maximizar), costo/esfuerzo (a minimizar), riesgo (a minimizar) y eventualmente otros como el equilibrio de trabajo o la precedencia de requisitos. Enfoques evolutivos (NSGA-II, SPEA2, etc.) dominan este espacio, aunque reforzados por heurísticas de negocio (por ejemplo, penalizaciones por dependencias entre historias) y complementados por modelos exactos para validación. Los resultados suelen presentarse como conjuntos de posibles *release plans* de Pareto, entre los cuales los gestores pueden elegir según sus preferencias.

2.2 Priorización de Requisitos y Selección de Características

Relacionado con el NRP, pero más general es el problema de priorizar requisitos o *features* en un backlog ágil. Mientras que el NRP da una solución binaria (elegir o no cada requisito para la próxima versión), la priorización puede entenderse como asignar un orden o ranking a los requisitos considerando múltiples criterios.

Algunos enfoques multiobjetivo tratan la priorización como un problema de ordenamiento óptimo: por ejemplo, algoritmos evolutivos de ranking que buscan permutaciones de requisitos que optimicen simultáneamente criterios de valor de negocio, riesgo y dependencia técnica. Una solución de Pareto en este contexto sería un ordenamiento tal que no exista otro orden que mejore un criterio sin empeorar otro. Govil & Sharma (2021) aplicaron un algoritmo basado en colonias de hormigas multiobjetivo para priorizar requisitos, definiendo feromonas asociadas a la importancia y al costo, produciendo varios rankings candidatos balanceados. Sin embargo, esta área es menos frecuente que la selección directa (NRP), dado que en la práctica la priorización suele resolverse mediante métodos más simples (como MoSCoW, dot voting, etc.), mientras que la selección optimizada se utiliza cuando hay datos cuantitativos suficientes.

2.3 Calendarización de Proyectos y Asignación de Tareas/Recursos

La calendarización en proyectos de software implica decidir quién realiza qué tarea y en qué momento, dentro de las restricciones de capacidad del equipo y las dependencias entre tareas. En metodologías ágiles, típicamente se realiza planificación de cada sprint (iteración) asignando historias de usuario al equipo, pero también puede considerarse la planificación a nivel de *release* completo. Los objetivos multiobjetivo más comunes en scheduling de proyectos de software son: (a) minimizar la duración total del proyecto o cumplir con la fecha objetivo lo antes posible, (b) minimizar el costo asociado (considerando costo-hora de desarrolladores, horas extra, etc.), y (c) maximizar la calidad o minimizar retrasos (por ejemplo, evitar asignar tareas a desarrolladores no idóneos que generen errores o retrabajo).

En contextos ágiles se añaden otros criterios relevantes: maximizar la satisfacción del equipo (evitando sobrecarga de ciertos miembros, distribuyendo equitativamente el trabajo) y adaptabilidad a cambios (por ejemplo, favorecer planes que dejen holgura para incorporar nuevos requisitos imprevistos).

En general, la optimización multiobjetivo en calendarización de proyectos de software busca proveer al Scrum Master o Project Manager un conjunto de planes viables que ponderan de forma distinta rapidez vs. costo vs. carga de trabajo. Un plan podría terminar antes, pero requerir más personas o más costo (horas extra, contrataciones temporales), mientras otro plan podría ser más barato, pero extenderse más en el tiempo.

La utilidad práctica es presentar estas alternativas para decisión informada. Los MOEAs han sido efectivos en encontrar dichas alternativas, y los modelos matemáticos (como programación lineal entera) también se han usado cuando el problema se simplifica (por ejemplo, asignación estática de personas a roles sin considerar cronograma detallado, lo cual se puede formular linealmente).

2.4 Otras aplicaciones

Además de los anteriores, existen aplicaciones menos frecuentes pero interesantes de optimización multiobjetivo en el ciclo de vida ágil/DevOps. Por ejemplo, la optimización de pruebas (testing) en entornos ágiles continuos: identificar subconjuntos de casos de prueba para ejecutar en cada iteración que maximizan cobertura de código y minimizan tiempo de ejecución (un problema multiobjetivo analizado por Tağtekin, Öztürk, & Sezer (2021)). Otro caso es la optimización de arquitectura evolutiva: decidir refactorizaciones o re-arquitecturas durante el desarrollo ágil considerando costo inmediato vs. facilidad de mantenimiento futura vs. performance (varios objetivos), como indican (Vemuri, Tatikonda, & Thaneeru, 2022).

Aunque estos temas se adentran en ingeniería de software más que en la gestión de proyectos puramente, muestran el potencial amplio de los métodos multiobjetivo. Esta revisión se centra en los ámbitos de planificación de alcance (requisitos) y planificación temporal/recursos por ser los más directamente ligados a la gestión de proyectos ágiles.

Tabla 2:
Aplicaciones típicas de optimización multiobjetivo en gestión de proyectos ágiles, con objetivos y enfoques.

Problema / Decisión	Objetivos comunes (a maximizar ⬆ / minimizar ⬇)	Técnicas/Enfoques aplicados	Referencias clave
Planificación de versión (NRP) – Selección de requisitos para próximo release.	⬆ Valor de negocio / Satisfacción de clientes; ⬇ Costo o Esfuerzo de desarrollo; ⬇ Riesgo técnico; ⬆ Equilibrio satisfacción entre stakeholders (equidad)	MOEAs (NSGA-II, SPEA2, MOEA/D, etc.); Algoritmos híbridos (enfriamiento sim., colinas) en estudios clásicos; Programación entera multiobjetivo (en instancias pequeñas); Heurísticas de búsqueda multiarranque.	(Zhang, Harman, & Mansouri, 2007); (Amaral & Elias, 2018); (Rahimi, Gandomi, Nikoo, & Chen, 2023); (Dong, Xue, Brinkkemper, & Li, 2022); (Dukhan, Mohamed, Amer, Zanaty, & Reyad, 2022)

Priorización de requisitos – Ordenar backlog considerando criterios múltiples.	⬆ Satisfacción ponderada (varios interesados); ⬇ Complejidad o Riesgo acumulado temprano; ⬆ Valor entregado lo antes posible.	Algoritmos evolutivos adaptados a permutaciones; Optimización basada en hormigas; Enfoques de decisión multicriterio (AHP/ ANP combinados con búsqueda).	(Kumari, Singh, & Ranjan, 2019); (Govil & Sharma, 2021)
Asignación de tareas / calendarización interna – Asignar desarrolladores a tareas (por sprint o proyecto).	⬇ Duración total (makespan) del proyecto/release; ⬇ Costo (horas-persona, horas extra); ⬆ Utilización equilibrada del equipo (evitar sobrecargas); ⬇ Retraso respecto a plan; ⬆ Calidad (p.ej., minimizar cambios de contexto o errores).	Algoritmos genéticos multiobjetivo (representación de asignación y calendario); Algoritmos meméticos con búsqueda local (para ajuste fino de cronograma); Programación por metas (para integrar prioridades de plazo vs costo).	(Shen, Minku, Marturi, Guo, & Han, 2018); (Kaur, Singh, Anand, & Agarwal, 2023)
Replanificación ágil – Ajuste del plan ante cambios (nuevos requerimientos, cambios en equipo).	⬇ Impacto en plazos (retraso adicional); ⬇ Trabajo extra (horas adicionales requeridas); ⬆ Valor aún entregado vs. plan original; ⬇ Disrupción al equipo (cambios en asignaciones).	MOEAs (comparativa de NSGA-II, IBEA, MOEA/D para distintos criterios de calidad); Enfoques dinámicos iterativos (re-ejecutar algoritmo con estado previo como semilla); Modelos de optimización robusta (optimizar peor caso de escenarios).	(García-Nájera, Zapotecas-Martínez, Falcón-Cardona, & Cervantes, 2021); (Zapotecas-Martínez & Ponsich, 2020)
Planificación multi-proyecto – Asignar equipos a múltiples proyectos ágiles en paralelo.	⬆ Throughput (nº proyectos finalizados en plazo); ⬇ Tiempo ocioso de equipos; ⬆ Sincronización (que todos terminen en tiempos cercanos); ⬇ Costos de retraso/ penalización.	Algoritmos genéticos multiobjetivo (codificación multi-proyecto); Algoritmos de enjambre (p. ej. PSO multiobjetivo ajustando asignaciones de equipo); Programación matemática multiobjetivo (en casos reducidos).	(Sayyad, Mohammed, Shaga, Kumar, & Vengatesan, 2019); (Aldhubaiban & AlMatouq, 2025)

Optimización de pruebas (DevOps) – Selección de casos de prueba regresiva por iteración.	⬆ Cobertura de funcionalidades/código; ⬇ Tiempo total de ejecución de pruebas; ⬇ Redundancia (casos similares); ⬆ Detección temprana de fallos críticos.	Algoritmos genéticos multiobjetivo (representación binaria de selección de tests); Algoritmos de Pareto Local Search; Heurísticas voraces multiobjetivo (greedy add/drop).	(Asha, Rajesh, Verma, & Poonguzhali, 2023) (Tağtekin, Öztürk, & Sezer, 2021)

Nota: La lista no es exhaustiva; refleja las aplicaciones más documentadas en la literatura revisada. Algunas aplicaciones, como la optimización de procesos DevOps o la planificación de capacidad a largo plazo, también pueden abordarse con enfoques multiobjetivo, pero no fueron el foco principal de la revisión.

Se evidencia que la optimización multiobjetivo en proyectos ágiles de software ya no se queda solo en el Next Release Problem, sino que también se aplica a la priorización, la calendarización y otras decisiones del ciclo ágil. Aun así, la mayoría de las propuestas se han probado en entornos controlados y con datos pequeños, por lo que su escalabilidad y uso real siguen siendo un reto. Lo revisado confirma que estas técnicas pueden ayudar a los equipos ágiles a decidir mejor (como cuando se generan planes de versión o asignaciones más equilibradas y transparentes), pero también evidencia la necesidad de integrarlas de forma sencilla con las herramientas que ya usa la industria, estandarizar métricas y validar resultados con casos reales para que dejen de ser solo experimentos académicos y pasen a la práctica cotidiana.

3. Herramientas y Plataformas Utilizadas

El desarrollo e implementación de algoritmos de optimización multiobjetivo en contextos de proyectos de software suele apoyarse en diversas herramientas de software especializadas, tanto para la experimentación investigativa como, en menor medida, para la adopción práctica. De esta revisión se identifican principalmente dos categorías: frameworks de optimización metaheurística y solvers/entornos de programación matemática, además de utilidades para generar instancias de prueba representativas de proyectos ágiles.

Frameworks de optimización multiobjetivo (metaheurísticas)

Muchos estudios que emplean algoritmos evolutivos o de enjambre utilizan frameworks de código abierto para facilitar la implementación de variantes. Uno de los más citados es ***jMetal*** (Durillo & Nebro, 2011), un framework en Java diseñado específicamente para algoritmos multiobjetivo. Varios trabajos reportan haber implementado NSGA-II, SPEA2, MOEA/D, etc., usando jMetal y, a veces, extendiéndolo con operadores personalizados para el problema en cuestión (por ejemplo, un operador de cruce específico para representaciones de asignación de tareas).

De forma similar, el framework **MOEA Framework** (Hadka, 2025) y librerías en Python como DEAP o Platypus se mencionan en algunos estudios para configurar algoritmos evolutivos estándar. Estas herramientas proveen implementaciones probadas de los algoritmos, así como medidas de desempeño que facilitan la comparación. Por ejemplo, Rahimi et al. (2023) mencionan haber utilizado MOEA Framework para ejecutar comparativamente NSGA-II, NSGA-III, SPEA2, MOEA/D y PAES sobre instancias de NRP, garantizando condiciones uniformes de comparación.

Solvers matemáticos y lenguajes de modelado

En los trabajos que abordan formulaciones exactas (programación lineal entera multiobjetivo, programación por metas, etc.), es común el uso de solvers y lenguajes de modelado matemático. **IBM ILOG CPLEX** y **Gurobi** son solvers de optimización ampliamente usados; algunos estudios (Dong, Xue, Brinkkemper, & Li, 2022) formulan el NRP multiobjetivo y luego utilizan CPLEX para resolver iterativamente variaciones con restricciones epsilon (obteniendo soluciones Pareto óptimas).

En programación por metas, herramientas como **LINGO/LINDO** o incluso hojas de cálculo avanzadas se han empleado para resolver modelos de tamaño moderado. Kaur, Singh, Anand, & Agarwal (2023) indican que resolvieron sus modelos de asignación de recursos con un enfoque de programación por metas usando implementaciones propias en MATLAB, valiéndose de su solver de programación lineal subyacente para cada variante de la función objetivo-agregada.

Simuladores y generación de instancias de proyecto

Un desafío particular en esta área es disponer de datos de entrada realistas para experimentar: por ejemplo, listas de requisitos con valores y costos, o planes de proyecto con tareas y personal. Algunos trabajos usan datos de proyectos de la industria anonimizados, pero la mayoría recurre a instancias sintéticas. Herramientas como IBM Rational Team Concert (RTC) (arcadsoftware, 2025) o Jira (Atlassian, 2025) pueden exportar datos de proyectos ágiles, pero no se encontraron referencias explícitas a su uso directo en optimización.

Herramientas de apoyo a la decisión

Finalmente, cabe mencionar que más allá de la experimentación académica, existen intentos de llevar estas optimizaciones a herramientas de apoyo a la decisión para gestores de proyectos. Por ejemplo, se han desarrollado dashboards prototipo donde el gerente ingresa sus datos de backlog o sprint y el sistema (usando un motor de optimización detrás, típicamente un GA multiobjetivo) genera varias opciones de planning, visualizando el trade-off entre criterios. En la literatura revisada, Ferrucci, Harman, Ren, & Sarro (2013) describen la visión de tales herramientas como parte del paradigma de *Analytics for Software Project Management*.

En resumen, las implementaciones de optimización multiobjetivo en proyectos ágiles revisadas típicamente combinan frameworks de metaheurísticas para experimentación algorítmica, solvers de OR para enfoques exactos, y a menudo requieren diseñar o simular datos de entrada representativos. Esta disponibilidad ha permitido avanzar rápidamente en variantes y pruebas controladas, pero también ha generado una cierta distancia con la práctica real: los gestores de proyectos rara vez cuentan con estas herramientas ni con datos listos para optimizar. Un reto pendiente es convertir estos marcos en soluciones accesibles e integradas dentro de las plataformas ágiles habituales, de modo que la optimización multiobjetivo deje de ser un ejercicio académico y se convierta en un recurso cotidiano para la toma de decisiones.

4. Desafíos Actuales y Tendencias Emergentes

A pesar de los avances notables en la última década, la aplicación de optimización multiobjetivo a proyectos ágiles de software enfrenta diversos desafíos abiertos. A continuación, se discuten los más relevantes identificados en la literatura reciente, así como las tendencias que están marcando la agenda de investigación.

4.1 Manejo de la Dinamicidad e Incertidumbre

Los entornos ágiles se caracterizan por el cambio constante – nuevos requisitos pueden surgir, prioridades pueden cambiar, la productividad real puede diferir de la estimada, etc. Un plan multiobjetivo óptimo puede volverse obsoleto rápidamente ante dichos cambios. Esto plantea el desafío de desarrollar algoritmos adaptativos o robustos. Algunos trabajos, como el de Shen, Minku, Marturi, Guo, & Han (2018) y García-Nájera, Zapotecas-Martínez, Falcón-Cardona, & Cervantes (2021), han abordado problemas dinámicos explícitamente, ya sea incorporando mecanismos de reacción (como reiniciar la optimización parcialmente cada vez que ocurre un cambio, o mantener una población evolutiva que continuamente se ajuste) o mediante optimización robusta (optimizar asumiendo el peor caso de ciertas perturbaciones). Sin embargo, no existe aún un consenso sobre la mejor forma de manejar la incertidumbre en estos problemas.

4.2 Escalabilidad y Complejidad Computacional

Si bien los metaheurísticos pueden manejar problemas más grandes que los métodos exactos, proyectos de software reales pueden involucrar cientos de requisitos o tareas, combinados con múltiples objetivos, lo cual resulta en espacios de búsqueda de enorme dimensión. Un desafío es cómo escalar las técnicas para que sigan siendo útiles con tiempos de cómputo razonables. En contextos ágiles, la planificación suele hacerse en iteraciones cortas (por ejemplo, planificación de sprint cada dos semanas), por lo que una herramienta de optimización debería entregar resultados en minutos u horas a lo sumo, no días.

Algunos enfoques que se vislumbran en la literatura para mejorar la escalabilidad incluyen:

(a) Algoritmos paralelos o distribuidos – por ejemplo, ejecutar una población evolutiva distribuida en varios hilos o máquinas (ya hay versiones paralelas de NSGA-II y MOEA/D que podrían aplicarse); (b) Uso de metamodelos o surrogate models – entrenar modelos predictivos (p.ej. redes neuronales) para estimar la calidad de soluciones sin evaluarlas completamente, ahorrando tiempo en la evaluación de objetivos complejos (esto se ha visto en optimización de diseño, pero podría aplicarse a estimar métricas de proyecto); (c) Dividir el problema en subproblemas – por ejemplo, primero optimizar selección de requisitos y luego optimizar asignación de personal, en lugar de todo en una sola formulación, aunque esto requiere cuidado para no perder la óptima global; **(d) Podas y heurísticas iniciales** – usar soluciones factibles obtenidas con métodos tradicionales (planificaciones manuales o greedy) como parte de la población inicial de algoritmos evolutivos, de modo que la búsqueda parta de regiones prometedoras.

Hasta ahora, estos estudios trabajaron con casos de prueba relativamente pequeños (por practicidad de experimentación), por lo que la escalabilidad a casos mayores sigue siendo en parte teórica. Desde esta perspectiva, esta brecha refleja un problema de transferencia a la práctica, ya que muchas empresas no disponen de la infraestructura ni del tiempo

para ejecutar algoritmos complejos en cada iteración.

En este sentido, es necesario avanzar hacia modelos híbridos y herramientas más ligeras que combinen técnicas de optimización con heurísticas propias del dominio, incorporen aprendizaje incremental y aprovechen recursos paralelos o en la nube. Solo así la optimización multiobjetivo podrá integrarse de forma rutinaria en la planificación ágil y no quedar restringida a estudios de laboratorio.

4.3 Multiplicidad de Objetivos y Preferencias del Decisor

Si bien agregar objetivos aporta riqueza al modelo, demasiados objetivos pueden dificultar tanto al algoritmo (problema de muchos-objetivos) como al usuario final, que debe elegir de un frente de Pareto potencialmente muy diverso. En optimización multiobjetivo clásica, se considera que más de 3 objetivos se vuelve problemático para métodos como NSGA-II (de ahí el desarrollo de NSGA-III y otros). En problemas de proyectos de software, fácilmente se enumeran 4 o 5 criterios relevantes (costo, tiempo, calidad, riesgo, satisfacción...).

Una tendencia es la adopción de algoritmos multiobjetivo (como NSGA-III, MOEA/D con técnicas especiales, o algoritmos basados en indicadores de alta dimensión) para asegurarse de obtener frentes diversos en casos de más de cuatro objetivos. Aunque no se reportaron implementaciones concretas de esto en los estudios recientes revisados, es una dirección mencionada conceptualmente en artículos sobre frameworks de apoyo a decisión (Ferrucci, Harman, Ren, & Sarro, 2013). La dificultad radica en lograr una interacción que no sea demasiado compleja o que los decisores (generalmente no expertos en optimización) entiendan.

4.4 Métricas de Desempeño y Comparabilidad

Un aspecto más académico pero importante es cómo se evalúa la efectividad de los algoritmos. Deb & Jain (2014) en su revisión sistemática enfatizan la variedad de métricas usadas en SBSE multiobjetivo. Hipervolumen, spread, generational distance, son algunas de las métricas de calidad del frente de Pareto. La diversidad de métricas dificulta comparar resultados entre estudios, ya que cada autor puede reportar métricas distintas. Por ejemplo, un estudio puede decir “nuestro algoritmo obtuvo un hipervolumen de 0.8, superando a NSGA-II con 0.75”, mientras otro quizá use inverted generational distance. Esta falta de estandarización es un desafío para construir conocimiento acumulativo.
La tendencia es moverse hacia reportes más completos: en varios trabajos recientes ((Rahimi, Gandomi, Nikoo, & Chen, 2023); (García-Nájera, Zapotecas-Martínez, Falcón-Cardona, & Cervantes, 2021)) se ve que los autores reportan múltiples métricas y hasta aplican pruebas estadísticas (como Mann-Whitney, y Test de Friedman) para afirmar con significancia qué algoritmo es mejor. Esta rigurosidad estadística se está convirtiendo en norma en conferencias de optimización. Aun así, desde el punto de vista práctico, sería ideal definir un conjunto mínimo de métricas relevantes para estos problemas (quizá combinando una métrica de proximidad al frente ideal, otra de diversidad y otra de equidad de soluciones) y que los futuros estudios las adopten para facilitar comparaciones directas.

4.5 Adopción Industrial y Usabilidad

Finalmente, un desafío transversal es que muchos de estos avances permanecen en el entorno académico o de simulación, con poca adopción en entornos ágiles industriales reales. Las causas pueden incluir: carencia de confianza (los gestores podrían desconfiar

de una “caja negra” que les sugiere planes), falta de integración con herramientas de gestión (por ejemplo, Jira no ofrece de fábrica optimizaciones multiobjetivo, y pocas empresas desarrollarían sus propias), y también la relativa novedad – muchos gerentes no conocen que estas técnicas existen.

En síntesis, los desafíos actuales de la optimización multiobjetivo en proyectos ágiles no son solo técnicos, sino también de transferencia a la práctica y giran en torno a cómo hacer las técnicas más aplicables en escenarios reales: manejando cambios, escalando a problemas grandes, ayudando al decisor a lidiar con muchos objetivos, y eventualmente cerrando la brecha con la práctica industrial.

La revisión muestra que hay avances conceptuales y prototipos prometedores (algoritmos dinámicos, integración con IA, métricas más completas y atención a aspectos humanos), pero también evidencia que faltan validaciones a gran escala, estándares comunes y herramientas integradas en las plataformas ágiles existentes. Abordar estos vacíos será clave para que, en los próximos años, la optimización multiobjetivo pase de ser una práctica experimental a convertirse en parte habitual del toolkit de gestión de proyectos de software

CONCLUSIONES

La optimización multiobjetivo se ha consolidado en los últimos años como una herramienta valiosa para abordar decisiones complejas en proyectos ágiles de desarrollo de software. Se han identificado las principales estrategias algorítmicas empleadas (con predominio de metaheurísticas evolutivas como NSGA-II, NSGA-III, SPEA2, entre otras, complementadas por enfoques exactos como programación entera multiobjetivo y métodos híbridos), así como las aplicaciones frecuentes en las que dichas técnicas han demostrado su potencial (planificación de *releases*, priorización de requisitos, asignación de tareas y replanificación adaptativa, principalmente).

La literatura revisada confirma que problemas de gestión ágil tradicionalmente resueltos de forma intuitiva pueden modelarse formalmente como problemas multiobjetivo, generando soluciones de alta calidad que exploran compensaciones entre criterios y ofrecen al equipo opciones más informadas.

No existe un algoritmo “dominante” único para todos los casos, la eficacia depende del contexto. Los algoritmos evolutivos genéricos funcionan bien en la mayoría de los escenarios, pero en situaciones con alta incertidumbre pueden requerirse extensiones dinámicas o enfoques robustos; los métodos exactos garantizan óptimos en problemas pequeños y sirven como referencia para validar heurísticas en problemas mayores.

Frameworks como jMetal y solvers como CPLEX han sido clave para avanzar la investigación, pero trasladar esos avances a herramientas prácticas de planificación ágil es aún una tarea pendiente. Una conclusión importante es que la adopción industrial de estas técnicas requerirá esfuerzos multidisciplinarios: no solo mejorar algoritmos, sino también trabajar en usabilidad, integración con herramientas de gestión de proyectos, y formación de profesionales en su uso.

El camino hacia proyectos ágiles optimizados pasa por abordar su naturaleza dinámica y humana. Los algoritmos deberán incorporar aprendizaje continuo, ser transparentes y configurables para incluir las preferencias del equipo, enriqueciendo la experiencia

humana sin reemplazarla. La investigación futura apunta a hibridar la optimización con técnicas de inteligencia artificial como aprendizaje por refuerzo profundo y sistemas de recomendación, así como a explorar objetivos alineados con metodologías modernas, por ejemplo, sostenibilidad del proceso o innovación en la planificación.

En síntesis, la optimización multiobjetivo ofrece un marco riguroso y potente para mejorar la toma de decisiones en proyectos ágiles. Los avances recientes han ampliado algoritmos y aplicaciones, pero queda trabajo para superar limitaciones y ganar aceptación industrial. Modelar cuidadosamente los problemas, elegir algoritmos apropiados y comunicar resultados a los decisores son pasos clave para que, al integrarse en el ciclo de vida ágil, los equipos tomen decisiones más informadas y logren proyectos con mayor probabilidad de éxito en resultados y satisfacción.

Una recomendación emergente es emplear múltiples métodos de forma complementaria. En la práctica, un gestor podría usar una herramienta heurística rápida para obtener recomendaciones y, cuando sea posible, validar alguna de ellas con un modelo exacto simplificado (por ejemplo, verificando si, fijando algunas decisiones, el resto es óptimo bajo ciertas condiciones).

REFERENCIAS BIBLIOGRÁFICAS

Aldhubaiban, A., & AlMatouq, A. (Junio de 2025). Efficient scheduling of multiple software projects for work continuity and identical completion time. *MethodsX*, *14*, 103215. doi:https://doi.org/10.1016/j.mex.2025.103215

Amaral, A., & Elias, G. (2018). A Multi-Objective, Risk-based Approach for Selecting Software Requirements. *Proceedings of the 10th International Conference on Agents and Artificial Intelligence (ICAART)*. *2*, págs. 338-346. Madeira: SciTePress. doi:10.5220/0006555503380346

arcadsoftware. (Marzo de 2025). *IBM Rational Team Concert*. Obtenido de Agile Application Lifecycle Management: https://www.arcadsoftware.com/arcad/products/rtc-rational-team-concert/#:~:text=Rational%20Team%20Concert%20(RTC)%20is,together%20and%20complement%20each%20other.

Asha, A., Rajesh, A., Verma, N., & Poonguzhali, I. (Abril de 2023). Multi-objective-derived energy efficient routing in wireless sensor networks using hybrid African vultures-cuckoo search optimization. *International Journal Communication System*, e5438. doi:https://doi.org/10.1002/dac.5438

Atlassian. (Junio de 2025). *Great outcomes start with Jira*. Obtenido de atlassian.com: https://www.atlassian.com/software/jira/guides/getting-started/introduction

Deb, K., & Jain, H. (Abril de 2014). An Evolutionary Many-Objective Optimization Algorithm Using Reference-Point-Based Nondominated Sorting Approach, Part I: Solving Problems With Box Constraints. *IEEE. Transactions on Evolutionary Computation*, *18*(4), 577-601. doi:10.1109/TEVC.2013.2281535

Deb, K., Pratap, A., Agarwal, S., & Meyarivan, T. (2002). A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, *6*, 182-197. doi:10.1109/4235.996017

Dong, S., Xue, Y., Brinkkemper, S., & Li, Y.-F. (Julio de 2022). Multi-objective integer programming approaches to Next Release Problem — Enhancing exact methods for finding whole pareto front. *Information and Software Technology*, *147*(C), 106825. doi:https://doi.org/10.1016/j.infsof.2022.106825

Dukhan, W. H., Mohamed, M. H., Amer, A. A., Zanaty, E. A., & Reyad, O. (Diciembre de 2022). Software requirement selection using a combined multi-objectiveoptimisation

technique. (J. W. Ltd, Ed.) *IET Software*, *16*(6), 558-575. doi:https://doi.org/10.1049/sfw2.12070

Durillo, J. J., & Nebro, A. J. (2011). jMetal: A Java framework for multi-objective optimization. *Advances in Engineering Software*, *42*(10), 760-771. doi:https://doi.org/10.1016/j.advengsoft.2011.05.014

Ferrucci, F., Harman, M., Ren, J., & Sarro, F. (2013). Not Going to Take this Anymore: Multi-Objective Overtime Planning for Software Engineering Projects. *Proceedings - International Conference on Software Engineering* (págs. 462-471). San Francisco: IEEE. doi:http://dx.doi.org/10.1109/ICSE.2013.6606592

García-Nájera, A., Zapotecas-Martínez, S., Falcón-Cardona, J. G., & Cervantes, H. (2021). Multi-objective Release Plan Rescheduling in Agile Software Development. En C. d. Instituto Politécnico Nacional (Ed.), *Advances in Computational Intelligence: 20th Mexican International Conference on Artificial Intelligence, MICAI 2021* (págs. 403-414). México: MICAI 2021. doi:https://doi.org/10.1007/978-3-030-89817-5_30

Govil, N., & Sharma, A. (2021). Information Extraction on Requirement Prioritization Approaches in Agile Software Development Processes. *5th International Conference on Computing Methodologies and Communication (ICCMC)* (págs. 1097-1100). Erode, India,: IEEE. doi:10.1109/ICCMC51019.2021.9418285

Hadka, D. (2025). *MOEA framework-a free and open source Java framework for multiobjective optimization*. Recuperado el 6 de Junio de 2025, de moeaframework.org: https://github.com/MOEAFramework/MOEAFramework.

Kaur, J., Singh, O., Anand, A., & Agarwal, M. (Marzo de 2023). A goal programming approach for agile-based software development resource allocation. *Decision Analytics Journal*, *6*, 100146. doi:https://doi.org/10.1016/j.dajour.2022.100146

Kitchenham, B., & Charters, S. M. (2007). *Guidelines for performing Systematic Literature Reviews in Software Engineering*. Keely, Durham: Keele University y Durham University. Obtenido de BSE Technical Report EBSE-2007-01: https://www.elsevier.com/__data/promis_misc/525444systematicreviewsguide.pdf

Kumari, M., Singh, V. R., & Ranjan, R. (2019). Multi-objective planning of rural distribution system using PSO. *International Conference on Next Generation Computing Technologies*. *922*, págs. 116-127. Singapur: Springer Singapore. doi:https://doi.org/10.1007/978-981-15-1718-1_10

Liu, B. Z., Zhou, Y., Luo, Q., & Wei, Y. (22 de Mayo de 2024). Quantum-inspired multi-objective African vultures optimization algorithm with hierarchical structure for software requirement. *Cluster Computing*, *27*, 11317–11345. doi:https://doi.org/10.1007/s10586-024-04503-6

Liu, J., & Dai, Q. (29 de Enero de 2020). Portfolio optimization of photovoltaic/battery energy storage/electric vehicle charging stations with sustainability perspective based on cumulative prospect theory and MOPSO. *Sustainability*, *12*(3), 985. doi:https://doi.org/10.3390/su12030985

Rahimi, I., Gandomi, A. H., Nikoo, M. R., & Chen, F. (2023). A comparative study on evolutionary multi-objective algorithms for next release problem. *Applied Soft Computing*, *144*, 110472. doi:https://doi.org/10.1016/j.asoc.2023.110472.

Sayyad, S., Mohammed, A., Shaga, V., Kumar, A., & Vengatesan, K. (2019). Digital Marketing Framework Strategies Through Big Data. *Proceeding of the International Conference on Computer Networks, Big Data and IoT (ICCBI - 2018)* (págs. 1065–1073). Switzerland: Springer, Cham. doi:http://dx.doi.org/10.1007/978-3-030-24643-3_127

Schwaber, K., & Sutherland, J. (Noviembre de 2020). *The Scrum Guide: The definitive guide to Scrum: The rules of the game*. Obtenido de The 2020 Scrum GuideTM: https://scrumguides.org/docs/scrumguide/v2020/2020-Scrum-Guide-US.pdf#zoom=100

SCRUM.ORG. (6 de marzo de 2025). *What is Scrum*? Recuperado el 30 de marzo de 2025, de www.scrum.org: https://www.scrum.org/resources/what-scrum-module

Sharma, D., Gaur, P., & Mittal, A. P. (20 de Noviembre de 2014). Comparative Analysis of Hybrid GAPSO Optimization Technique With GA and PSO Methods for Cost Optimization of an Off-Grid Hybrid Energy System. *Energy Technology & Policy, 1*(1), 106-114. doi:https://doi.org/10.1080/23317000.2014.969450

Shen, X.-N., Minku, L. L., Marturi, N., Guo, Y.-N., & Han, Y. (Febrero de 2018). A Q-learning-based memetic algorithm for multi-objective dynamic software project scheduling. *Information Sciences, 428*, 1-29. doi:https://doi.org/10.1016/j.ins.2017.10.041

Tağtekin, B., Öztürk, M. U., & Sezer, M. K. (9 de Junio de 2021). *A Case Study: Using Genetic Algorithm for Job Scheduling Problem*. Obtenido de arXiv.org: https://arxiv.org/pdf/2106.04854

Tong, H., Minku, L. L., Menzel, S., Sendhoff, B., & Yao, X. (2021). A hybrid local search framework for the dynamic capacitated arc routing problem. *GECCO '21: Proceedings of the Genetic and Evolutionary Computation Conference Companion* (págs. 139-140). Nueva York: Association for Computing Machinery. doi:https://doi.org/10.1145/3449726.3459450

Vemuri, N., Tatikonda, V. M., & Thaneeru, N. (2022). Integrating Deep Learning with DevOps for Enhanced Predictive Maintenance in the Manufacturing Industry. *Tuijin Jishu/ Journal of Propulsion Technology, 43*(4), 315-322. doi:http://dx.doi.org/10.52783/tjjpt.v43.i4.5041

Zapotecas-Martínez, S., & Ponsich, A. (26 de Junio de 2020). Constraint handling within MOEA/D through an additional scalarizing function. *GECCO '20: Proceedings of the 2020 Genetic and Evolutionary Computation Conference*, 595 - 602. doi:https://doi.org/10.1145/3377930.3390240

Zhang, Y., Harman, M., & Mansouri, S. A. (2007). The Multi-Objective Next Release Problem. *Annual Conference of Genetic and Evolutionary Computation Conference* (págs. 1129 - 1136). Londres: ASSOC Computing Machinery. doi:https://doi.org/10.1145/1276958.1277179

Zitzler, E., & Künzli, S. (2004). Indicator-Based Selection in Multiobjective Search. *Parallel Problem Solving from Nature - PPSN VIII. 3242*, págs. 832–842. Berlin: Springer. doi:https://doi.org/10.1007/978-3-540-30217-9_84

Zitzler, E., Laumanns, M., & Thiele, L. (2001). *SPEA2: Improving the strength pareto evolutionary algorithm*. Swiss Federal Institute of Technology (ETH). Zurich, Switzerland: ETH Zurich, Computer Engineering and Networks Laboratory. doi:https://doi.org/10.3929/ethz-a-004284029