Combinación de computación evolutiva y aprendizaje automatizado en la resolución de problemas de optimización

Doctorando: Alejandro Marrero*,
Directores: Eduardo Segredo†, Coromoto León‡
Departamento de Ingeniería Informática y de Sistemas,
Facultad de Ciencias, Sección de Matemáticas,
Universidad de La Laguna (ULL).
Apartado 456 CP 38200
San Cristóbal de La Laguna. S/C de Tenerife.
Email: *alu0100825008@ull.edu.es, †esegredo@ull.edu.es, ‡cleon@ull.edu.es

***Resumen*—En este documento se detalla el proyecto de investigación de una tesis doctoral relacionada con la computación evolutiva, el aprendizaje automatizado y su aplicación a la resolución de un subconjunto de problemas de optimización. Principalmente, se persigue el desarrollo de un modelo hiperheurístico que sea capaz de evaluar el desempeño de diversos algoritmos evolutivos multiobjetivo sobre un conjunto de instancias de un problema determinado. De este modo, cuando se presente una nueva instancia de dicho problema se pueda inferir, basándose en las características propias de dicha instancia y los resultados obtenidos previamente con instancias de similares características, qué algoritmo de los disponibles obtendría el mejor desempeño sobre esa nueva instancia dada. Asimismo, con este documento se pretende describir el trabajo que se llevará a cabo a partir de la hipótesis inicial y además, de qué manera se realizará está investigación y cómo los resultados obtenidos pueden ser transferidos a la sociedad.**

***Index Terms*—computación evolutiva, aprendizaje automatizado, optimización combiatoria.**

I. INTRODUCCIÓN

Los problemas de *optimización* pueden clasificarse en optimización discreta u optimización combinatoria y optimización continua dependiendo si las variables del problema tratado son discretas o continuas [1]. Debido a que esta investigación se centrará, principalmente, en la optimización de problemas combinatorios, serán este tipo de problemas a los que se haga referencia en el presente documento. Sin embargo, no se descarta aplicar las técnicas desarrolladas durante esta investigación a la resolución de posibles problemas de optimización continua que puedan surgir.

Cualquier problema que tenga un conjunto de soluciones discretas y una función de coste para puntuar estas soluciones respecto a las demás es un *Problema de Optimización Combinatoria (POC)*. Del mismo modo que otros problemas de optimización, el objetivo para un POC es encontrar una solución óptima al problema. Adicionalmente, un POC incluye un conjunto de restricciones que limitan el número de posibles soluciones. Las soluciones que satisfacen el conjunto de restricciones para un problema de optimización se llaman *soluciones factibles*.

El documento continua detallando los antecedentes sobre los que se sustenta este proyecto, los objetivos fijados de la investigación, la metodología y plan de trabajo que se llevará a cabo para alcanzar dichos objetivos y, por último, la relevancia científica y la transferencia a la sociedad de este proyecto de tesis doctoral.

II. ANTECEDENTES

En un POC, el número de posibles soluciones crece exponencialmente con el tamaño del problema *n* a $\mathcal{O}(n!)$ o $\mathcal{O}(e^n)$ por lo que no existe ningún algoritmo que pueda encontrar la solución óptima en un tiempo de computación polinómico [1]. Formalmente, un problema de optimización combinatoria se denota como (χ, f, Ω) y se formula de la siguiente forma [1]:

$$min\ f(x), x \in \chi, \quad sujeto\ a\ \Omega$$

donde $\chi \subset \mathbb{Z}^n$ es el espacio de búsqueda definido sobre un conjunto de *n* variables de decisión discretas $x = (x_1, x_2, ..., x_n)$, $f : \chi \to \mathbb{R}$ es una función que convierte una solución dentro del espacio de búsqueda χ en un valor entero, *n* es el número de variables de decisión y Ω es el conjunto de restricciones en x. En esta definición se ha empleado una formulación de un problema de minimización, sin embargo, la formulación sería equivalente en el caso de un problema de maximización excepto por el cambio: $min\ f(x)$ por $max\ f(x)$.

La definición proporcionada anteriormente describe la formulación de un POC mono-objetivo, es decir, únicamente existe una función objetivo a optimizar. Sin embargo, también podemos encontrar POCs multiobjetivo. En ese caso, en lugar de tener una única función objetivo a optimizar, existe en cambio un vector de funciones objetivos a optimizar. La investigación de este proyecto de tesis doctoral se centrará en la resolución de problemas de optimización combinatoria multiobjetivo, pero no exclusivamente.

Un *Problema de Optimización Combinatoria Multiobjetivo* (POM) se describe como encontrar un vector x dentro del dominio χ de forma que optimiza el vector de funciones objetivo $f(x)$. Además, los distintos objetivos en un POM suelen estar en conflicto y pueden tomar direcciones distintas. Entendiendo el término dirección como *maximizar* o *minimizar*. Los objetivos en conflicto causarán que el incremento de la calidad de un objetivo tienda, simultáneamente, a decrecer la calidad de otros objetivos. Por ello, la solución a un POM no es una única solución, sino un conjunto de soluciones que representan los mejores compromisos entre los objetivos. Formalmente, la formulación de un POM es [1]:

$$\begin{aligned} min\ f(x) = (f_1(x), f_2(x), ..., f_k(x)),\ x\ \in \chi \\ g_i(x) \leq 0,\ i = 1, 2, ..., q. \\ h_i(x) \leq 0,\ i = 1, 2, ..., p. \end{aligned}$$

donde $x = (x_1, x_2, ..., x_n) \in \mathbb{Z}^n$, las funciones objetivo a optimizar $f_i : \mathbb{Z}^n \to \mathbb{R},\ i = 1, ..., k$ siendo n el número de variables de decisión y las funciones $g_i : \mathbb{Z}^n \to \mathbb{R},\ i = 1, ..., q$ y $h_i : \mathbb{Z}^n \to \mathbb{R},\ i = 1, ..., p$ son las funciones de restricción del problema. Aunque en esta definición solamente se presentan dos funciones de restricción g y h, el número de funciones de restricción del problema puede ser $l \geq 1$.

El método estándar para trabajar con un POM es el método de Pareto. Este método se basa en el principio de no-dominancia [1], [2]. En primer lugar, hay que definir el concepto de *dominancia*. El concepto de dominancia es simple. Dadas dos soluciones, ambas con puntuaciones de acuerdo a un conjunto de funciones objetivos definidas, se dice que una solución domina a otra si su puntuación es al menos de la misma calidad para todos los objetivos y, estrictamente de mejor calidad en al menos uno [2]. Formalmente, la dominancia se puede expresar como: dadas dos soluciones A y B, diremos que la solución A domina a la solución B si:

$$\begin{aligned} A \succeq B \Leftrightarrow \forall i \in \{1, 2, ..., n\}\ a_i \leq b_i, \\ y\ \exists i \in \{1, 2, ..., n\},\ a_i < b_i \end{aligned}$$

Cuando existen objetivos en conflicto, no existe una única solución que domine al resto de soluciones. Entonces, es cuando aparece el concepto de *no-dominancia*. La no-dominancia hace referencia a la situación en la que una solución no es dominada por ninguna otra, es decir, no se puede encontrar ninguna otra solución que mejore la calidad de algún valor objetivo sin que se empeore la calidad de otro valor objetivo de dicha solución. Las soluciones no dominadas suelen encontrarse en el límite de la región factible del espacio de búsqueda definido por el conjunto de funciones de restricción del problema tratado. Este conjunto de soluciones no dominadas se suele llamar generalmente conjunto de Pareto [1].

Hoy en día, existe un gran abanico de posibilidades en cuanto a técnicas de optimización. Como se puede apreciar en la Figura 1, los algoritmos de optimización se pueden dividir en *métodos exactos* y *métodos aproximados*.

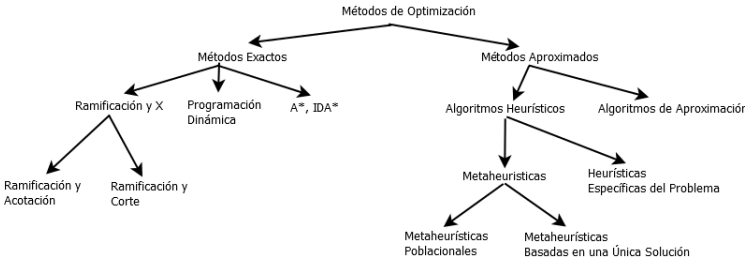

Figura 1. Métodos de optimización.

Por un lado, los algoritmos exactos son aquellos que se caracterizan, principalmente, porque obtienen la solución óptima, si existe, al problema para el que se emplean. Sin embargo, este no será el campo de estudio de este proyecto de tesis doctoral. Por otro lado, están los algoritmos aproximados que a diferencia de los métodos de optimización exactos, permiten obtener soluciones factibles a problemas complejos en un tiempo aceptable, aunque no garantizan obtener la solución óptima global [3]. A pesar de esto, en los últimos años estas técnicas algorítmicas han ganado una gran popularidad dado que han demostrado su efectividad y eficiencia en muchos campos y con una gran cantidad de problemas distintos [4]–[6]. Los algoritmos aproximados se pueden subdividir principalmente en heurísticas, metaheurísticas e hiperheurísticas. Las palabras *meta* y *heurística* tienen su origen en la antigua Grecia: *meta* significa 'nivel superior' y *heurística* denota el arte de descubrir nuevas estrategias [1]. Las heurísticas son técnicas basadas en la experiencia para la resolución de problemas y el aprendizaje [1], [2]. El término metaheurísticas fue acuñado en 1986 por Glover para hacer referencia a un conjunto de técnicas conceptualmente posicionadas encima de las heurísticas en el sentido que estas guiaban el diseño de las heurísticas [1], [2]. En esencia, una metaheurística es un procedimiento de alto nivel o heurística diseñada para encontrar, generar o seleccionar un procedimiento de bajo nivel o heurística que pueda proporcionar una solución suficientemente buena a un problema de optimización. Además, dentro del conjunto de metaheurísticas existentes, podemos encontrar metaheurísticas poblacionales o metaheurísticas basadas en una única solución. Esta nueva clasificación discierne entre algoritmos que emplean una única solución que será modificada con cada iteración del algoritmo hasta el final de su ejecución de los algoritmos que emplean un conjunto de $m \geq 2$ soluciones factibles que son modificadas hasta que la condición de parada sea alcanzada.

Por último, la idea de hiperheurísticas apareció a principios de los años 60. Las hiperheurísticas se pueden ver como técnicas heurísticas que permiten seleccionar o generar (meta) heurísticas. Una hiperheurística es un método de búsqueda heurístico que persigue automatizar el proceso de seleccionar, combinar, generar o adaptar varias heurísticas o metaheurísti-

cas para resolver problemas complejos eficientemente. Al contrario que las metaheurísticas, las hiperheurísticas no trabajan sobre el espacio de búsqueda del problema tratado sino que siempre lo hacen sobre el espacio de búsqueda de heurísticas o meta-heurísticas del nivel inferior. Asimismo, las hiperheurísticas de generación se pueden basar en la programación genética o evolución gramatical [1], [2].

Adicionalmente, existe un conjunto de técnicas de optimización dentro de un marco denominado *Computación Evolutiva (CE)* que trata de sintetizar el comportamiento de la naturaleza o la interacción entre ciertas especies de animales y aplicar estos comportamientos a la resolución de problemas de optimización [2]. La metáfora fundamental sobre la que se sustenta el campo de la CE es el poder de la evolución natural y su particular estilo para solventar problemas, el método de prueba y error.

Si consideramos la evolución natural como la existencia de un entorno determinado en el que conviven individuos de una determinada población que buscan su supervivencia, podríamos denominar las aptitudes de cada uno de los individuos de la población como su calidad o *fitness* para determinar la posibilidad de supervivencia y reproducción de cada uno de ellos. Así pues, en el marco de la resolución de problemas, el entorno sería el problema tratado y la población de individuos estaría representada por un conjunto de soluciones factibles al problema dónde cada una de ellas posee un valor de calidad que determinará su "supervivencia" y la posibilidad de que sean empleadas como semillas para obtener nuevas soluciones candidatas.

Actualmente existen diversas variantes de algoritmos evolutivos aunque se puede destacar principalmente las siguientes categorías:

- Algoritmos Genéticos [7]–[9].
- Estrategias Evolutivas [10], [11].
- Evolución Diferencial [12]–[15].

Dado que la investigación de este proyecto de tesis doctoral se centrará en la resolución de problemas multiobjetivo, cabe hacer especial mención al estado del arte en cuanto a algoritmos evolutivos multiobjetivo, centrándose en aquellos aplicados a la resolución de problemas de optimización combinatoria multiobjetivo.

A la hora de resolver un problema de optimización combinatoria multiobjetivo, se puede hacer uso de diversos algoritmos evolutivos multiobjetivo (Multiobjective Evolutionary Algorithms - MOEAs) como pueden ser entre otros:

- Multiobjective Genetic Algorithm (MOGA) [2], [16].
- Non-dominated Sorting Genetic Algorithm (NSGA) [2], [16].
- Non-dominated Sorting Genetic Algorithm II (NSGA-II) [2], [16].
- Niched Pareto Genetic Algorithm (NPGA) [2], [16].
- Strength Pareto Evolutionary Algorithm (SPEA-2) [2], [16].
- Pareto Archived Evolutionary Strategy (PAES) [2], [16].
- Indicator-Based Evolutionary Algorithm (IBEA) [17].

Sin embargo, hay que tener en cuenta que probar el rendimiento de estos algoritmos requiere realizar experimentos computacionales con varias instancias distintas del mismo problema tratado. Por lo tanto, el desempeño de estos algoritmos se puede ver afectado por la idiosincrasia de la instancia del problema tratado, y elegir un único algoritmo para optimizar todas las instancias de un determinado problema puede no ser eficiente. Es entonces, cuando la aplicación del aprendizaje automatizado o *Machine Learning (ML)* aparece.

El término aprendizaje automatizado implica la construcción de modelos matemáticos para ayudar a entender los datos [18]. El aprendizaje entra en escena cuando se proporciona a estos modelos parámetros modificables que pueden ser adaptados a partir de los datos observados [18]. Dentro del término aprendizaje automatizado se pueden encontrar también varias categorías en función del tipo de aprendizaje empleado para el modelo desarrollado:

- Aprendizaje Supervisado *(Supervised Learning)*: un modelo con aprendizaje supervisado es entrenado con un conjunto de datos de entrenamiento hasta alcanzar un cierto umbral de acierto y posteriormente pasa a un estado de validación con un conjunto de datos distinto. En ese momento, el modelo puede ser empleado con un nuevo conjunto de datos distinto al utlizado en las fases de entrenamiento y validación [18].
- Aprendizaje no Supervisado *(Unsupervised Learning)*: el aprendizaje no supervisado implica obtener las características del conjunto de datos sin tener referencias previas [18].
- Aprendizaje Semi-supervisado *(Semi-supervised Learning)*: esta aproximación se encuentra a medio camino entre el aprendizaje supervisado y el aprendizaje no supervisado y suele ser útil cuando sólo se tienen unas pocas referencias previas [18].
- Aprendizaje por Refuerzo *(Reinforcement Learning)*: el aprendizaje por refuerzo se basa en aprender continuamente de las tareas realizadas para posteriormente poder determinar el comportamiento idóneo en un contexto específico para maximizar su rendimiento [19].

Además de lo explicado anteriormente, el aprendizaje automatizado se puede clasificar en diversos paradigmas. Este proyecto de tesis doctoral se orientará al paradigma denominado *Lifelong Machine Learning o Lifelong Learning* (o Aprendizaje Permanente). El aprendizaje permanente es una técnica avanzada de aprendizaje automatizado que se caracteriza porque los modelos construidos bajo esta filosofía aprenden continuamente, acumulan conocimiento adquirido de las tareas previas y lo usan para las futuras tareas a realizar [20], [21].

Las aplicaciones generales del aprendizaje automatizado junto con la computación evolutiva suele tener como objetivo aumentar el desempeño de algoritmos evolutivos para aprovechar el conocimiento adquirido a lo largo de la ejecución de un experimento computacional y así incrementar la eficiencia de los resultados finales [22]–[24].

'Combinación de computación evolutiva y aprendizaje automatizado en la resolución de problemas de optimización'

es un proyecto de tesis doctoral que propone la unión de algoritmos metaheurísticos basados en la CE y el ML [25] para la resolución de problemas de optimización combinatoria multiobjetivo. No obstante, este proyecto de tesis doctoral no pretende seguir las aproximaciones convencionales que desarrollan el resto de estudios del campo sino que, persigue el desarrollo de un modelo hiperheurístico que sea capaz de evaluar el desempeño de diversos algoritmos evolutivos multiobjetivo sobre un conjunto de instancias de un problema determinado. De este modo, cuando se presente una nueva instancia de dicho problema se pueda inferir, basándose en las características propias de dicha instancia y los resultados obtenidos previamente con instancias de similares características, qué algoritmo de los disponibles obtendría el mejor desempeño sobre esa nueva instancia dada.

Por último, el desarrollo del modelo hiperheurístico se realizará en lenguaje de programación Python. Si bien existen numerosos lenguajes de programación que proporcionan herramientas de desarrollo de aprendizaje automatizado y módulos matemáticos, Python se ha convertido en el lenguaje más empleado en la actualidad debido a la gran cantidad de módulos que proporciona, su sencillez y fiabilidad. La combinación de estas características han convertido a Python en el nuevo lenguaje de la *computación científica* para la resolución de problemas como: simulaciones numéricas, modelos apropiados y análisis de datos y optimización.

III. Objetivos

Concretamente, el objetivo principal y subobjetivos de este proyecto de tesis doctoral son:

- Desarrollar una herramienta que incluya un modelo hiperheurístico de aprendizaje permanente haciendo uso de las herramientas de aprendizaje automatizado proporcionadas en el lenguaje Python [18], [20], [26].
  1. Implementar el modelo hiperheurístico con diversos tipos de aprendizaje:
     a) Aprendizaje Supervisado.
     b) Aprendizaje por Refuerzo.
  2. Generar gráficos con los resultados que apoyen los resultados obtenidos.
  3. Generar informes del proceso realizado por el modelo.
  4. Integrar el modelo con una herramienta que proporcione algoritmos evolutivos multiobjetivo y problemas de optimización combinatoria para poder realizar experimentos computacionales.

Para la realización de experimentos computacionales y testeo del modelo hiperheurístico desarrollado se trabajará sobre problemas académicos y problemas reales. Con respecto al marco académico, se trabajará con problemas ampliamente estudiados como el problema del Viajante de Comercio (*Travelling Salesman Problem - TSP)*, el probblema de la Mochila Multidimensional Multiobjetivo *(Multidimensional Multiobjective Knapsack Problem - MMKP)* [27]–[29], entre otros. En cambio, en el marco de la aplicación a problemas reales, se estudiarán problemas relacionados con el área de la nutrición dentro del proyecto *Meta-heuristics for the Optimisation of Resources and Problems in Health* en el que trabaja el grupo de investigación de Algoritmos y Lenguajes Paralelos de la Universidad de La Laguna.

IV. Metodología y Plan de Trabajo

Durante el desarrollo de este proyecto de tesis doctoral se diferenciará entre la metodología general de investigación seguida y la metodología empleada para el desarrollo del software del modelo hiperheurístico.

La metodología de investigación que se seguirá para el desarrollo de esta tesis doctoral se basa en las referencias tomadas de [30]–[32]. La metodología seguida será la de una investigación cuantitativa con experimentos computacionales marcada por las siguientes fases:

1. Fase conceptual: esta fase inicial se subdivide en las siguientes subfases:
   a) Formulación y delimitación del problema a tratar.
   b) Revisión de la literatura asociada.
   c) Construcción del marco teórico.
   d) Formulación de Hipótesis.
2. Fase de formulación y diseño: fase en la que se definirán los experimentos a realizar:
   a) Selección de un conjunto de instancias del problema tratado para estudiar.
   b) División del conjunto de instancias en diferentes subconjuntos en función del tipo de aprendizaje empleado por el modelo.
   c) Diseño de los experimentos.
3. Fase empírica: incluye los siguientes procesos:
   a) Realización de los experimentos diseñados.
   b) Recolección de datos (resultados de los experimentos realizados).
   c) Preparación de los datos para realizar un análisis.
4. Fase analítica: comprende los procesos de análisis de los resultados obtenidos:
   a) Análisis de datos.
   b) Interpretación de los resultados del análisis.
5. Fase de difusión: esta fase esta destinada a la redacción y publicación de la investigación realizada.

Por otro lado, metodología para el desarrollo del modelo hiperheurístico, se basará en las directrices de metodologías de desarrollo de software ágiles, la metodología de desarrollo software guiado por pruebas (Test-Driven Development-TDD) [26], [33], [34], los principios SOLID [35] de la programación orientada a objetos y la filosofía de código abierto. Además, el seguimiento del correcto funcionamiento de la herramienta desarrollada se corroborará con la utilización de la plataforma Travis-CI[1] para integración continua.

Si bien poner práctica estas metodologías y directrices pueden ralentizar ligeramente el proceso de desarrollo de software, su aplicación ayuda enormemente al mantenimiento

[1] https://travis-ci.org/

del software desarrollado así como a reducir el número de posibles errores de codificación dentro del programa [26].

Hoy por hoy, existen diversos entornos y herramientas que implementan multitud de algoritmos heurísticos y metaheurísticos para la resolución de problemas mono-objetivos y multiobjetivos en diversos lenguajes de programación como pueden ser: JMetal [36], [37], HeuristicLab [38], ParadisEO [39], etc. No obstante, la herramienta seleccionada para la realización de experimentos computacionales en esta investigación será la denominada Metaheuristic-based Extensible Tool for Cooperative Optimisation (METCO) [40].

En lo que respecta al plan de trabajo para este proyecto, se diseñará considerando los objetivos *SMART* [41], [42], siglas en inglés para específico, medible, alcanzable, relevante y con tiempo limitado *(Specific-Measurable-Attainable-Realistic-Timely)*. Por ello, se definirán las tareas a realizar dentro de un espacio de cuatro años con una dedicación a tiempo completo. Así, el plan de trabajo propuesto se compone de las siguientes tareas y subtareas:

1. Estudiar el estado del arte de los tópicos tratados en este proyecto:
   a) Aprendizaje permanente.
   b) Aprendizaje Supervisado y Aprendizaje por Refuerzo.
   c) Algoritmos Evolutivos Multiobjetivo.
   d) Problemas académicos y reales enumerados en la sección III.
2. Implementar uno de los problemas académicos en la herramienta METCO.
3. Implementar uno de los problemas reales en la herramienta METCO.
4. Recabar instancias del problema desarrollado, preferiblemente ya usadas en algún estudio previo para contrastar los resultados obtenidos.
5. Experimentación computacional inicial con el objetivo de obtener los resultados iniciales para el modelo.
   a) Diseño del experimento: instancias a emplear, algoritmos que se probarán y parametrización de los algoritmos.
   b) División del conjunto de instancias en dos conjuntos: entrenamiento y validación.
   c) Ejecución del experimento para obtener los resultados iniciales.
6. Análisis de los resultados experimentales iniciales.
   a) Recolección de los resultados experimentales.
   b) Análisis de los resultados obtenidos mediante un test estadístico.
   c) Interpretación de los resultados obtenidos.
7. Desarrollo del modelo hiperheurístico en Python:
   a) Definición del formato que deben tener los ficheros de las instancias y resultados que se van a emplear.
   b) Desarrollo del modelo hiperheurístico con aprendizaje supervisado.
   c) Desarrollo del modelo hiperheurístico con aprendizaje por refuerzo.
   d) Desarrollo de un módulo que permita obtener gráficos que soporten las conclusiones obtenidas por el modelo.
   e) Desarrollo de informes generados por el proceso realizado por el modelo.
8. Experimentación computacional con el modelo hiperheurístico y METCO.
   a) Diseño del experimento: nuevas instancias a emplear.
   b) Ejecución del experimento.
9. Análisis de los resultados experimentales.
   a) Recolección de los resultados experimentales.
   b) Análisis de los resultados obtenidos mediante un test estadístico.
   c) Interpretación de los resultados obtenidos
10. Desarrollo de la documentación relativa al software desarrollado:
    a) Generar la documentación del software en formato PDF, HTML y Markdown.
    b) Crear un tutorial de uso de la herramienta en formato PDF, HTML y Markdown.
11. Redacción de la memoria de la tesis doctoral. Este proceso, aunque se incluye en último lugar, será realizado de manera incremental tras finalizar cada una de las tareas precedentes.

Especialmente, la consecución de las tareas 2, 3, 7, 8 y 9 representan los hitos que van a marcar el cumplimiento de los objetivos propuestos en la sección III.

V. RELEVANCIA

Desde el punto de vista de la relevancia que puede tener este proyecto de tesis doctoral, podríamos destacar por un lado que, a nivel científico, supondría un cambio en la tendencia en el campo de investigación con respecto a la aplicación de técnicas de aprendizaje automatizado junto con algoritmos de Computación Evolutiva para la resolución de problemas de optimización.

Además, los problemas académicos con los que se trabajará en este proyecto de tesis doctoral no han sido resueltos aplicando este enfoque por lo que se aportaría nuevos resultados de gran interés como base para nuevas investigaciones.

Por otro lado, los problemas académicos tratados tienen una aplicación casi directa al mundo real. Por ejemplo, el problema MMKP tiene muchas aplicaciones en el ámbito industrial, social y logístico. Asimismo, el modelo matemático que define el problema MMKP puede ser extraído a otros problemas existentes como por ejemplo el problema de la planificación de menús [43]. De igual manera, el empleo de las técnicas desarrolladas al proyecto *Meta-heuristics for the Optimisation of Resources and Problems in Health* proporcionará una aplicación directa a la sociedad.

Es por ello, que la obtención de resultados relevantes acerca de los problemas académicos tratados, y el proyecto *Meta-heuristics for the Optimisation of Resources and Problems in Health* así como la finalización exitosa de este proyecto de

tesis doctoral, puede conducir a nuevos y mejores resultados no sólo de estos problemas, sino de otros problemas con similar formulación matemática, y por consiguiente a la mejora de procesos industriales y factores sociales.

Agradecimientos

Este trabajo ha sido financiado por el Ministerio de Economía, Industria y Competitividad dentro del programa 'I+D+i Orientada a los Retos de la Sociedad' con referencia de proyecto TIN2016-78410-R.

Referencias

[1] K.-L. Du and M. N. S. Swamy, *Search and Optimization by Metaheuristics: Techniques and Algorithms Inspired by Nature*, 1st ed. Birkhäuser Basel, 2016.

[2] A. E. Eiben and J. E. Smith, *Introduction to Evolutionary Computing*. SpringerVerlag, 2003.

[3] E.-G. Talbi, *Metaheuristics*, 1st ed. Wiley, 2009.

[4] L. Antonio, J. Berenguer, and C. Coello, "Evolutionary many-objective optimization based on linear assignment problem transformations," *Soft Computing*, pp. 1–22, 2018, cited By 0; Article in Press.

[5] M. Köksalan and D. Tezcaner Öztürk, "An evolutionary approach to generalized biobjective traveling salesperson problem," *Computers and Operations Research*, vol. 79, pp. 304–313, 2017, cited By 0.

[6] W. Mashwani and A. Salhi, "Multiobjective evolutionary algorithm based on multimethod with dynamic resources allocation," *Applied Soft Computing Journal*, vol. 39, pp. 292–309, 2016, cited By 2.

[7] D. Whitley, "A genetic algorithm tutorial," *Statistics and Computing*, 1994.

[8] P. G. Algorithms, S. Edition, R. Haupt, and S. Haupt, "The continuous genetic algorithm," *Practical Genetic Algorithms, Second . . .* , 2004.

[9] S. Sivanandam and S. Deepa, "Genetic Algorithm Implementation Using Matlab," *Introduction to Genetic Algorithms*, 2008.

[10] H.-g. Beyer and H.-p. Schwefel, "Evolution strategies," *Evolutionary Computation*, 2002.

[11] N. Hansen, "CMA-ES Python package," 2017.

[12] T. Algorithm, "Differential Evolution," *Evolution*, 2006.

[13] L. Zheng, S. Zhang, K. Tang, and S. Zheng, "Differential evolution powered by collective information," *Information Sciences*, vol. 399, pp. 13–29, 2017, cited By 0.

[14] C. Fu, C. Jiang, G. Chen, and Q. Liu, "An adaptive differential evolution algorithm with an aging leader and challengers mechanism," *Applied Soft Computing Journal*, vol. 57, pp. 60–73, 2017, cited By 0.

[15] M. Tian, X. Gao, and C. Dai, "Differential evolution with improved individual-based parameter setting and selection strategy," *Applied Soft Computing Journal*, vol. 56, pp. 286–297, 2017, cited By 0.

[16] A. Zhou, B.-Y. Qu, H. Li, S.-Z. Zhao, P. N. Suganthan, and Q. Zhang, "Multiobjective evolutionary algorithms: A survey of the state of the art," *Swarm and Evolutionary Computation*, vol. 1, no. 1, pp. 32 – 49, 2011.

[17] E. Zitzler and S. Künzli, "Indicator-based selection in multiobjective search," in *Parallel Problem Solving from Nature - PPSN VIII*, X. Yao, E. K. Burke, J. A. Lozano, J. Smith, J. J. Merelo-Guervós, J. A. Bullinaria, J. E. Rowe, P. Tiňo, A. Kabán, and H.-P. Schwefel, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, pp. 832–842.

[18] J. VanderPlas, *Python Data Science Handbook: Essential Tools for Working with Data*, 1st ed. O'Reilly Media, Inc., 2016.

[19] L. P. Kaelbling, M. L. Littman, and A. W. Moore, "Reinforcement learning: A survey," *CoRR*, vol. cs.AI/9605103, 1996.

[20] Z. Chen and B. Liu, *Lifelong Machine Learning*. Morgan & Claypool Publishers, 2016.

[21] D. Silver, Q. Yang, and L. Li, "Lifelong machine learning systems: Beyond learning algorithms," 03 2013.

[22] J. Zhang, Z.-h. Zhang, Y. Lin, N. Chen, Y.-j. Gong, J.-h. Zhong, H. Chung, Y. Li, and Y.-h. Shi, "Evolutionary computation meets machine learning: A survey," *Comp. Intell. Mag.*, vol. 6, no. 4, pp. 68–75, Nov. 2011.

[23] M. Yoshida, T. Hinkley, S. Tsuda, Y. M. Abul-Haija, R. T. McBurney, V. Kulikov, J. S. Mathieson, S. Galinanes Reyes, M. D. Castro, and L. Cronin, "Using evolutionary algorithms and machine learning to explore sequence space for the discovery of antimicrobial peptides," *Chem*, vol. 4, no. 3, pp. 533–543, mar 2018.

[24] B. Moradi, "An intelligent evolutionary computation approach for solving the shortest path problem," *Journal of Multiple-Valued Logic and Soft Computing*, vol. 30, no. 4-6, pp. 335–357, 2018, cited By 0.

[25] M. Gimenez Fayos, "Una aproximacion basada en aprendizaje automatico para diversos problemas de procesamiento de lenguaje natural en redes sociales," 2016.

[26] M. Kirk, *Thoughtful Machine Learning with Python: A Test-Driven Approach*, 2nd ed. O'Reilly Media, Inc., 2017.

[27] K. Florios, G. Mavrotas, and D. Diakoulaki, "Solving multiobjective, multiconstraint knapsack problems using mathematical programming and evolutionary algorithms," *European Journal of Operational Research*, vol. 203, no. 1, pp. 14–21, 2010.

[28] T. Lust and J. Teghem, "The multiobjective multidimensional knapsack problem: A survey and a new approach," *International Transactions in Operational Research*, vol. 19, no. 4, pp. 495–520, 2012.

[29] N. Abboud, M. Sakawa, and M. Inuiguchi, "A fuzzy programming approach to multiobjective multidimensional 0-1 knapsack problems," *Fuzzy Sets and Systems*, vol. 86, no. 1, pp. 1–14, 1997.

[30] S. Rajasekar, P. Philominathan, and V. Chinnathambi, "Research Methodology," pp. 1–53, 2006.

[31] H. Hassani, "Research methods in computer science: The challenges and issues," *CoRR*, vol. abs/1703.04080, 2017.

[32] C. A. Monje Álvarez, "Metodología de la investigación cuantitativa y cualitativa. Guía didáctica." *Universidad Surcolombiana*, pp. 1–216, 2011.

[33] C. Desai, D. Janzen, and K. Savage, "A survey of evidence for test-driven development in academia," *SIGCSE Bull.*, vol. 40, no. 2, pp. 97–101, Jun. 2008.

[34] W. Bissi, A. Neto, and M. Claudia Figueiredo Pereira Emer, "The effects of test driven development on internal quality, external quality and productivity: A systematic review," vol. 74, 02 2016.

[35] H. Signh and S. I. Hassan, "Effect of solid design principles on quality of software: An empirical assessment," vol. 6, 04 2015.

[36] J. J. Durillo and A. J. Nebro, "jmetal: A java framework for multi-objective optimization," *Advances in Engineering Software*, vol. 42, pp. 760–771, 2011.

[37] J. Durillo, A. Nebro, and E. Alba, "The jmetal framework for multi-objective optimization: Design and architecture," in *CEC 2010*, Barcelona, Spain, July 2010, pp. 4138–4325.

[38] S. Wagner, G. Kronberger, A. Beham, M. Kommenda, A. Scheibenpflug, E. Pitzer, S. Vonolfen, M. Kofler, S. Winkler, V. Dorfer, and M. Affenzeller, *Advanced Methods and Applications in Computational Intelligence*, ser. Topics in Intelligent Engineering and Informatics. Springer, 2014, vol. 6, ch. Architecture and Design of the HeuristicLab Optimization Environment, pp. 197–261.

[39] S. Cahon, N. Melab, and E.-G. Talbi, "Paradiseo: A framework for the reusable design of parallel and distributed metaheuristics," *Journal of Heuristics*, vol. 10, no. 3, pp. 357–380, May 2004.

[40] C. León, G. Miranda, and C. Segura, "Metco: A parallel plugin-based framework for multi-objective optimization," *International Journal on Artificial Intelligence Tools*, vol. 18, no. 04, pp. 569–588, 2009.

[41] G. T. Doran, "There's a S.M.A.R.T. way to write managements's goals and objectives." *Management Review*, vol. 70, no. 11, pp. 35–36, 1981.

[42] C. Fuhrmann, J. Hobin, P. S. Clifford, and B. Lindstaedt, "Goal-setting strategies for scientific and career success," 12 2013.

[43] B. K. Seljak, "Computer-based dietary menu planning," *Journal of Food Composition and Analysis*, vol. 22, no. 5, pp. 414–420, 2009.