Universidad de La Habana
Facultad de Matemática y Computación

Optimización Multiobjetivo para Autogoal

Autor:
Rodrigo Daniel Pino Trueba

Tutores:

Suilan Estévez Velarde
Daniel Valdés Pérez

Trabajo de Diploma
presentado en opción al título de
Licenciado en Ciencia de la Computación

Fecha

github.com/rodrigo-pino/Thesis

Dedicación

Agradecimientos

Agradecimientos

Opinión del tutor

Opiniones de los tutores

Resumen

Resumen en español

Abstract

Resumen en inglés

Índice general

Índice de figuras

Ejemplos de código

Introducción

Aprendizaje de Maquina (ML por sus siglas en inglés) es una rama de la Inteligencia Artificial enfocada en entender y construir programas que "aprendan" y logren buenos resultados en un conjunto de tareas definidas (Mitchell y col. 1990). Este campo que vió sus inicios sobre la década de los 50 no tuvo auge hasta la primera decada del nuevo milenio gracias al auge de las nuevas tecnologias en el campo de la computación. Se comenzó a utilizar ampliamente en diferentes aplicaciones donde definir algoritmos convencionales era muy díficil.

Construir un flujo válido de Aprendizaje de Máquina requiere tener al menos un experto en el tema pero estos son escasos lo que hace la herramienta parcialmente inaccesible. Se crea un nuevo campo que estudia el proceso de automzatizar la creación de estos flujos de ML (conocido por AutoML por sus siglas en inglés). Esto impulsa aún mas la adopción del aprendizaje de máquina ya que se vuelve accesible a todos.

El AutoML funciona dado un set de datos y una metrica busca el mejor algorimto de ML de acuerdo a la puntuación de dicha métrica. Usualmente lo que buscan es exactitud en sus datos.

Con el uso masivo que tiene los algorimtos de Aprendizaje de Máquina en la actualidad, y se forma parte de la vida diaria, ya obtener un sistema ML que tenga resultados relevanes no es el desafio. Se empiezaan a quere nuevas features que permitan mayor control sobre el sistema que se genera. Hay sistemas que están dispuestos a sacrificar un poco de relevancia si se puede obtener un flujo de ML menos complejo (y evitar el overfit en el proceso). Otro que quieren tener sistemas "interpretables", sistemas que permiten a usuarios no expertos entender por que el algoritmo de ML tomo cierta decisión.

La adopción masiva saca a relucir problemas desconocidos como sistemas de ML que producen resultados paracializados dependiendo a ciertas carecterísticas de estos. Esto tiene importancia especialmente cuando los datos representan seres humanos, y las decisiones del sistema tiene un impacto en el individuo o grupo. No se desea que juzgue por el color de piel o género. Nace la necesidad de producir sistemas justos que consideren ciertos atributos como protegidos y minimicen el sesgo que esto puedan provocar en sus decisiones.

Motivación

Un modelo de Aprendizaje de Máquina o *Machine Learning(ML)* tiene como objetivo maximizar la relevancia de sus resultados dado una métrica. Para medir dichas relevancia, se utilizan métricas como *accuracy* o *f-score*.

Cuando el proposito de un programa de AutoML es producir un algoritmo ML que minimice la relevancia a costa de cualquier otro tipo de criterio se producen buenas predicciones, pero no necesariemente convenientes. Existen casos donde el investigador esta dispuesto a sacrificar un poco de precisión en las predicciones en favor de otras métricas dependiendo del contexto donde se encuentre.

Existen situaciones donde la *interpretabilidad* que implica saber, sin ser un experto en Aprendizaje de Máquina, porque el sistema tomó ciertas decisión puede ser importante. También existen casos donde es necsario producir sistemas de ML justos, para evitar que el algoritmo de ML tenga algún tipo de sesgo por género, color de piel u otro. Estos problemas se resolverían muy bien con multiobjetivo; en caso de no optimizar la relevancia y optimizar únicamente para justeza o interpretabilidad respectivamente produciría un sistema inútil debido a sus malas predicciones.

Optimización multibojetivo en AutoML abré las puertas también a métricas que no tienen sentido usarlas individualemente como *precisión* y *recobrado*. Precisión mide la proproción entre los valores relevantes identificados y todos los valores identificados mientras que recobrado mide la proporción entre los valores relevantes identificados y todos los valores relevantes. Estas métircas usadas por separadas no son represantivas pues basta para tener un recobrado perfecto seleccionar todos los elementos del conjunto de datos, y una precisión casi perfecta seleccionando un pequeño conjunto de elementos. Para lograr un balance entre estas se utiliza *f-score*, que relaciona ambas medidas y permite darle más peso a una o a la otra según determine el investigador, no obstante con OM se pueden obtener un sistema ML que optimice para ambas.

Un framework de *Automated Machine Learning(AutoML)* que pueda optimizar smultáneamente para varios objetivos, se beneficia al ser capaz de ofrecer un conjunto de modelos igual de buenos, y dejar al investigador que seleccione el que más le convenga según su contexto.

Antecedentes

En el entorno del grupo de Inteligencia Artificial de la Universidad de La Habana se desarrolla AutoGOAL una herramienta de AutoML que permite obtener modelos optimizados con respecto a una sola métrica.

Problemática

AutoGOAL no presenta actualmente una herramienta para resolver problemas multiobjetivos. AutoGOAL utiliza Probabilistic Grammatic Evolution (PGE), un algorimtos genético. Implementar un algorimto de Optimización Multiobjetivo que sea capaz de utilizar la infraestructura ya creada.

Objetivo

Objetivo General

Añadir a AutoGOAL la habilidad de resolver problemas de optimización multiobjetivo.

Objetivos Específicos

- Estudiar el estado del arte relacionado con el problema
- Identificar un algoritmo multiobjetivo que aproveche la estructura de AutoGOAL y su implementación de Probabilistic Grammatic Evolution (PGE)
- Estudiar diferencias cuando se resuelve un mismo problema con optimización multibojetivo
- Analizar la efectividad de la solución

Estructura de la Tesis

Capítulo 1

Estado del Arte

1.1. Optimización Multiobjetivo

Optimización Multiobjetivo es la rama de la Ciencia y la Matemática que se dedica a optimizar para varias funciones objetivos simúltaneamente.

Definición 1.1 *Optimización Multibojetivo: Dado m funciones objetivos:* $f_1 : \mathcal{X} \to \mathbb{R}, ..., f_m : \mathcal{X} \to \mathbb{R}$ *que dado un vector en el espacio de decisión* $\mathcal{X}$ *es transformado en un valor de* $\mathbb{R}$*. Más formalmente:*

$$\textit{MOP: } \min f_1(x), ..., f_m(x), x \in \mathcal{X}$$

En presencia de varias funciones objetivos ya no es posible hablar de una solución única mejor que el resto. Como las solucions se comparan respecto a dos o más aspectos, se habla de dominación.

Definición 1.2 *Pareto Dominación: Dados dos vectores en el el espacio objetivo,* $x \in \mathbb{R}^m$ *y* $y \in \mathbb{R}^m$*, se dice que x Pareto domina a y (i.e.* $x \prec y$*), si y solo si:*

$$\forall i \in \{1, ..., m\} : x_i \leq y_i \textit{ and } \exists j \in \{1, ..., m\} : x_j < y_j$$

Existe el caso en el conjunto de soluciones que existan soluciones que no sean dominadas por ninguna otra solución. Estas soluciones vienen siendo el conjunto solucion del problema de optimziación multiobjetivo.

Definición 1.3 *Frente de Pareto: Todos los vectores x del espacio objetivo* $\mathcal{Y}$ *tal que no exista* $y \prec x$*.*

$$\mathcal{P} = \{x | x, y \in \mathcal{Y}, \neg \exists y \prec x\}$$

Encontrar puntos del frente de Pareto no es tarea díficil, puede ser tan trivial con encontrar el mínimo de cada función objetivo. La complejidad yace en encontrar un conjunto solución que sea lo más aproximado posible al frente de Pareto.

Se debe notar que el termino Optimización Multiobjetivo se utiliza cuando se optimza simultáneamente para dos o tres funciones objetivos. Para un cantidad de métricas mayor se le conoce coloquialmente en la literatura como Optimización para Muchos Objetivos o *many-objective optimization* en inglés propuesto inicialmente por Fleming y col. 2005. Se hace enfásis en esta diferenciación este pues al aumentar le número de métricas a optimizar:

1. ya no es posible visualizar el frente obtenido;
2. la computación de indicadores o de selección para muchos algoritmos se convierte en problemas NP-duros;
3. existe un rápido crecimiento de puntos no dominados, mientras mayor número de objetivos, la probabilidad de que un punto sea no dominado en un set con distribución normal tiende exponencialmente a 1.

1.1.1. Técnicas de Escalarización (Scalarization)

Los manera clásica de resolver MOP es utilizando técinas de Escalarización que es cuando de alguna manera u otra se combinan todas las funciones objetivos en una sola. Existen diferentes técinas de combinar estas funciones:

1. Linear Weighting: Todas las funciones objetivos se combinan en una sola, y a cada una se le asigna un peso. Modificando estos pesos se obtiene una solución distinta del frente de Pareto.

   $$\text{mín} \sum w_i f_i(x), x \in X$$

   Esta enfoque presenta el problema de que cuando el frente de Pareto no es convexo (tiene alguna porción cóncava) no es posible obtener soluciones en esta zona, no importa como se modifiquen los pesos w_i.

2. ϵ-constrain: Se selecciona una función objetivo como principal y las demás se establecen como restricciones de esta, y tienen que ser menor que sierto ϵ_i con $1 \leq n-1$ por cada función objetivo.

   $$\begin{aligned} &\text{mín}\, f_1(x), x \in X, \text{ sujeto a:} \\ &\quad g_i(x) \leq \epsilon_i \quad \forall i, 0 \leq i \leq n-1 \end{aligned}$$

Utilizar esta técninca tiene dos dificultades principales, primero la obtención de los ϵ_i, para una adecuada selección requiren conocimiento previo del frente de Pareto y al igual que *Linear Weighting* no es capaz de detectar soluciones en las partes cóncavas del frente.

3. Chebychev Distance (CSP): Se establece un punto de referencia z^* y se utiliza la distancia de Chebychev de los vectores objetivos hacia este como función objetivo utilizando un vector de pesos $\lambda \in \mathbb{R}^m_{\prec 0}$, donde $\mathbb{R}^m_{\prec 0} = \{x | x \in \mathbb{R}^m, x \prec 0\}$.

$$\text{mín} \quad \max_{i \in 1,\ldots,m} \lambda_i |f_i(x) - z_i^*|, x \in X$$

CSP dado los pesos indicados puede encontrar cualquier punto del frente de Pareto, no importa si es cóncavo o convexo.

1.1.2. Algoritmos Numéricos

En principio todos los algoritmos de escalarización se pueden resolver utilzando métodos númericos.

Además, existen métodos númericos que intentan resolver el problema haciendo cumplir las condiciones de Karush-Kuhn-Tucker (Kuhn y Tucker 2014).

La idea va de encontrar al menos una solución al sistema de ecuaciones creado por tratar de resolver KKT. Luego utilizando métodos de continuación y homotopía para añadir al conjunto solución soluciones cercanas a estas. Este algoritmo require que las soluciones satisfagan las condiciones de convexidad local y diferenciabilidad, se puede obtener mas infomración en Hillermeier y col. 2001 y Schütze y col. 2005.

Tambien existen otros metodos para la búsqueda de mínimos globales como *Técnicas de Subdivisión* por Dellnitz y col. 2005, *Optimización Global Bayesiana* por Emmerich, Yang y col. 2016 y *Optimización de Lipschitz* por Žilinskas 2013. Estas requiren de que el espacio de decisión tenga una baja dimensionalidad.

Aun más, también se investiga activamente métodos númericos que no dependan de derivadas para su resolución utilizando técnicas de búsquedas directas. Ejemplos de algoritmos en Custódio y col. 2011 y Audet y col. 2010.

1.1.3. Algoritmos Genéticos Multiobjetivos (MOEA)

Los algorimtos genéticos tuvieron sus inicios en la decada del 60 y fueron usados principalmente en resolución de problemas numéricos combinatorios y no convexos. Utilizan paradigmas extraídos de la naturaleza, tal como seleccion natural, mutación y recombinación para mover una población (o conjunto de vectores de decisión) hacian una solución óptima (Back 1996).

Los algoritmos genéticos multiobjetivos generalizan esta idea, y son diseñados para en cada iteración acercarse cada vez más al frente de Pareto. Como en este caso no existe solución única la manera de selccionar los individuos cambia fundamentalmente.

Dentro de los MOEA existen tres paradigmas principales:

1. **MOEA basados en el frente de Pareto**: Se identifican porque dividen el proceso de selección en dos etapas. Una primera donde seleccionan los invidiuos seguún su indíce de dominación, donde las soluciones que pertenecen al frente de Pareto tienen indíce cero. Luego se realiza una segunda selección entre los ya seleccionados utilizando una segunda estrategia de puntuación. NSGA-II (Deb, Pratap y col. 2002) y SPEA2 (Zitzler y Thiele 1999) son algoritmos de este tipo.

2. **Basados en indicador**: Estos utilizan un indicador para calcular cuan cercano es el conjunto actual al frente de Pareto (unario), o cuanto mejora el nuevo conjunto de soluciones respecto a la iteración anterior (binario). Ejemplo de este es SMS-EMOEA (Emmerich, Beume y col. 2005) que suele converger al frente de Pareto con soluciones igualmente distribuidas.

3. **Basados en descomposición**: La idea principal consiste en descomponer el problema en pequeños subproblemas cada una correspondiente a una sección del frente de Pareto. Por cada sub-problema se resuelve utilizando escalarización con dierente paramtrización. El método de escalarización más usado en estos casos suele ser CSP debido a ser capaz de obtener cualquier punto del frente de Pareto. Ejemplo de este paradigma son MOEA/D (Zhang y Li 2007) y NSGA-III (Deb y Jain 2013).

1.2. Aprendizaje de Máquina Automatizado

El Aprendizaje de Máquina Automatizado (Automated Machine Learning, AutoML) consiste en la generación automática de flujos o *pipelines* de Machine Learning que resuelven un problema determinado. El objetivo es transferir el problema de combinación, selcción y optimización de hiperparámetros(*Combined Algorithm Selection and Hyperparameter Optimization*, CASH por sus siglas en inglés) del investigador al sistema, permitiendole a este enfocarse en otras tareas como la validación de los datos o ingeniería de característcas. También democratiza el uso de Aprendizaje de Máquina pues usarios normales sin los recursos económicos suficiente pueden aplicar a sus problemas flujos de Machine Learning sin la necesidad de un Científico de Datos gracias a que las operaciones de combinación y selección de modelos y optimización de hiperparámetros esta automatizada.

Existen varias propuestas de Aprendizaje de Máquina Automatizado, utilizando técnias de variados dominios. Entre los principales relacionados con el problema CASH:

- **Optimización Bayesiana** (Hutter y col. 2019): Utilizan optimización bayesiana para encontrar el mejor flujo de ML que maximice cierta métrica. Introducido por Auto-Weka en 2013 (Thornton y col. 2013) con el fin de resolver problemas CASH. Auto-Skelearn Feurer y col. 2015 crece sobre este introduciendo mejoras com la inclusion de un paso de meta-aprendizaje que reduce el espacio de búsquda aprendiendo de modelos que funcionaron bien en conjuntos de datos similares y luego un paso de selección de ensemblers que le permite reusar los flujos de ML que tuvieron mejor rendimiento.

- **Programación Evolutiva** Chen y col. 2018: Basándos en algoritmos genéticos funcionan creando una población inicial de flujos válidos, se seleccionan los de mejor rendmiento respecto a una métrica y se utilzan para crear la población de la próxima iteración. TPOT es uno de los más reconocidos en esta área, permiten tener varias copias sobre el dataset y aplicar operadores sobre ellos, luego con un operador de cruce, permite crear flujos con los operadores con mejor puntuación. TPOT además realiza una búsqueda multiobjetivo sobre las soluciones encontradas utilizando NSGA-II (Deb, Pratap y col. 2002) optimzando maximizar exactitud (*accuracy* en la literatura) y minimizando la cantidad de operadores aplicados sobre los flujos de ML por un tema de simplicidad y evitar el sobreajuste a los datos.

  AutoGOAL (Estevez-Velarde y col. 2020) genera soluciones utilizando una Gramaática Probablístca Libre del Contexto y Probabilistic Grammatic Evolution para actualizar las producciones.

1.3. Optimización Multiobjetivo en AutoML

Optimización Multibojetivo esta bien establecido en ML como ROC analysis, o biologia computacional. Son modelos optimizados para tener un buen performance e interpretabilidad. En el campo de Configuracion de Algoritomos introduce multi objetvivo busqueda iterativa local para configurar SAT solvers, minetras que Zhang introduce basado en carrera?

Aunque es cada vez un feature de más demanda en la actualidad, no se conoce ningún sistema de AutoML que tenga implementado multiobjetivo, excepto por TPOT que optimiza sus pipelines mutuamente para ‘accuracy‘ y tiempo de entranamiento. No obstante no presenta flexibiliad sobre que métricas optimizar.

Capítulo 2

Propuesta

2.1. Probabilistic Grammatic Evolution

Algoritmos evolucionarios estan sutilmente inspirados por las ideas de Evlocion Natural, donde una selecciion de invdividuos evoluciona a traves de una aplicacion de operadores como seleccion, mutacion y recombinacion. La calidad de estos individuos se evaluan en contra de una métrica de evaluacion. Tras aplicar los operadores e ir reteniendo los individuos con mayor puntuacion se espera que la poblacion mejore con el paso de las iteraciones.

2.1.1. GE

Con esta idea en mente nace Grammatical Evolution, que utiliza una gramatica para establecer restricciones sintácticas sobre las soluciones individuales. Exisite una distincion entre el genotipo y el fenotipo.

Para crear una solcuión se tienen el genotipo (Una lista de eneteros), que se mapea el fenotipo siguiendo las reglas de producción en una Gramática Libre del Contexto (*Context-Free Grammar*, CFG). Donde una gramatica es una tupla $G = (NT, T, S, P)$ donde NT y T representan los conjuntos disjuntos no vacío de los símbolos no terminales y terminales respectivamente. S es un elemento de NT llamado el axioma que representa el no-terminal principal que expandiendo este se puede llegar a todas las posibles formas de la gramatica. P es el conjunto de reglas de producción que rigen a la grmaática. Las reglas en P tienen la forma de $A ::= \alpha$, donde $A \in NT$ y $\alpha \in (NT \cup F)^*$

La relacion genotipo-fenotipo es la cosa principal de GE, y se realiza en pasos sucesivos. Con el fin de seleccionar la produccion que se debe escoger para sustituir un elemento de NT se utiliza el operador módulo.

Insertar ejemplo de como funciona esta talla

El rendimiento de GE ha sido criticado en la literatura por tener alta redundancia y poca localidad. Una representación tiene alta redundancia cuando varios genotipos corresponden al mismo fenotipo y localidad se refiere a como los cambios en el genotipo se refeljean en el fenotipo.

2.1.2. PGE

Con el objetivo de mejora GE han exisitido varias propuestas. Una de esta es Evolución de Gramática Probabilistica (*Probabilistic Grammatic Evolution*). En PGE el genotipo ya no es una lista de enteros, sino una lista con la misma longitud que se adapta segun los fentoipos escogidos o algo asi

Utiliza Algirmtos de Estimacion de Distribucion (*Estimation of Distribution Algorithms, EDA*) una técninca probabiística que remplaza los operadores de mutacion y cruce sampleando sobre la probabiliad de distribución del mejor individuo, para generar una nueva población en cada generación. Las probabilidades comienzan todas inicializadas en igual proporción y se actualizan basado en la frecuencia de las reglas de produccion escogidas para obtener el individuo con la mayor métrica.

Probablisitic Grammatical Evolution (PGE) se apoya en una Gramatica Probabilistica Libre del Contexto (*Probilistic Context-Free Grammatic Evolution* PCFGE) para realizar los mapeos de los fenotipos a los genotipos. PCFGE se establece como una tupla $PG = (NT, T, S, P, Prob)$ donde $Prob$ es un conjunto de probabilidades asocaido con cada regla de la gramática. El genotipo en PGE es un vector de numeros fraccionarios, donde cada uno corresponer con la probabiliad de seleccion cierta regla de derivación.

insetar ejemplo de PGE.

En PGE las probabilidades se actualizan despues de cada generación despues de evaluar la población generada, basasdo en cuantas veces cada regla de derivacion fue seleccionada por el el individuo de mejor rendimiento. Si la regla fue seleccionada su probablidad incrementa, en cambio si no, su probabilidad se reduce. Alternando entre estas dos variantes se ayuda a evitar usar el mismo individuo en iteraciones consecutivas, balanceando exploracion global con expotacion local.

2.2. Nodominated Sorting Genetic Algorithm

Como dicho en (insertar estado del arte), NSGA-II es (eso que dice). Se guia por dos ranking fundamentalmente

El primer ranking ordena las soluciones de acuerdo a su índice de dominación

Despues de obtenidas las N mejores respuestas, se quiere que el conjunto este bien esparcido sobre el frente de Pareto.

Se establece una métrica de densidad (Crowding distance): Para obtener un estimado de l a densidad de soluciones aledañas a cierata solucion se calcula la distancia promedio de dos puntos en ambos de cada objetivo. Esta cantidad sirve como un estimado del permiteero del cuboide formado usando los vecinos mas cercanos como vertices.

El calculo de crowding distanc ereuier la ordenacion de la poblacion de acuerdo a acada funcion objetivo. Las soluciones fronterizas (los valores minimos y maximos de cada funcion objetivo) son asignados distancia infinita. El resto de las soluciones intermedias son asignados son asignados a la distancia igual a la diferencia normalizada absoluta en los valores de las funciones objetivos de las soluciones adyacentes.

Es importante nomralizar la función objetivo antes de calcualr el crowding distance.

Poner pseudo codigo de Crowding distance

Luego el algorimto escoge de las poblacion general formada: Si x *prec* y, entonces se va x Si rank x == rank y entonces se escoge a x si su crowding distancia es mayor que y

2.3. Autogoal con NSGA-II y PGE

Como lo que existe ya de AutoGOAL interactua con la nueva implementación de PGA

Capítulo 3

Detalles de Implementación y Experimentos

Conclusiones

Conclusiones

Recomendaciones

Recomendaciones

Bibliografía

Audet, C., Savard, G. & Zghal, W. (2010). A mesh adaptive direct search algorithm for multiobjective optimization. *European Journal of Operational Research*, *204*(3), 545-556 (vid. pág. 6).

Back, T. (1996). *Evolutionary algorithms in theory and practice: evolution strategies, evolutionary programming, genetic algorithms*. Oxford university press. (Vid. pág. 6).

Chen, B., Wu, H., Mo, W., Chattopadhyay, I. & Lipson, H. (2018). Autostacker: A compositional evolutionary learning system. *Proceedings of the genetic and evolutionary computation conference*, 402-409 (vid. pág. 8).

Custódio, A. L., Madeira, J. A., Vaz, A. I. F. & Vicente, L. N. (2011). Direct multisearch for multiobjective optimization. *SIAM Journal on Optimization*, *21*(3), 1109-1140 (vid. pág. 6).

Deb, K. & Jain, H. (2013). 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 (vid. pág. 7).

Deb, K., Pratap, A., Agarwal, S. & Meyarivan, T. (2002). A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE transactions on evolutionary computation*, *6*(2), 182-197 (vid. págs. 7, 8).

Dellnitz, M., Schütze, O. & Hestermeyer, T. (2005). Covering Pareto sets by multilevel subdivision techniques. *Journal of optimization theory and applications*, *124*(1), 113-136 (vid. pág. 6).

Emmerich, M., Beume, N. & Naujoks, B. (2005). An EMO algorithm using the hypervolume measure as selection criterion. *International Conference on Evolutionary Multi-Criterion Optimization*, 62-76 (vid. pág. 7).

Emmerich, M., Yang, K., Deutz, A., Wang, H. & Fonseca, C. M. (2016). A multicriteria generalization of bayesian global optimization. *Advances in Stochastic and Deterministic Global Optimization* (pp. 229-242). Springer. (Vid. pág. 6).

Estevez-Velarde, S., Piad-Morffis, A., Gutiérrez, Y., Montoyo, A., Munoz, R. & Almeida-Cruz, Y. (2020). Solving heterogeneous automl problems with AutoGOAL.

ICML Workshop on Automated Machine Learning (AutoML@ ICML) (vid. pág. 8).

Feurer, M., Klein, A., Eggensperger, K., Springenberg, J., Blum, M. & Hutter, F. (2015). Efficient and robust automated machine learning. *Advances in neural information processing systems*, *28* (vid. pág. 8).

Fleming, P. J., Purshouse, R. C. & Lygoe, R. J. (2005). Many-Objective Optimization: An Engineering Design Perspective. En C. A. Coello Coello, A. Hernández Aguirre & E. Zitzler (Eds.), *Evolutionary Multi-Criterion Optimization* (pp. 14-32). Springer Berlin Heidelberg. (Vid. pág. 5).

Hillermeier, C. y col. (2001). *Nonlinear multiobjective optimization: a generalized homotopy approach* (Vol. 135). Springer Science & Business Media. (Vid. pág. 6).

Hutter, F., Kotthoff, L. & Vanschoren, J. (2019). *Automated machine learning: methods, systems, challenges*. Springer Nature. (Vid. pág. 8).

Kuhn, H. W. & Tucker, A. W. (2014). Nonlinear programming. *Traces and emergence of nonlinear programming* (pp. 247-258). Springer. (Vid. pág. 6).

Mitchell, T., Buchanan, B., DeJong, G., Dietterich, T., Rosenbloom, P. & Waibel, A. (1990). Machine learning. *Annual review of computer science*, *4*(1), 417-433 (vid. pág. 1).

Schütze, O., Dell'Aere, A. & Dellnitz, M. (2005). On Continuation Methods for the Numerical Treatment of Multi-Objective Optimization Problems. En J. Branke, K. Deb, K. Miettinen & R. E. Steuer (Eds.), *Practical Approaches to Multi-Objective Optimization* (pp. 1-15). Schloss Dagstuhl – Leibniz-Zentrum für Informatik. https://doi.org/10.4230/DagSemProc.04461.16. (Vid. pág. 6)
Keywords: multi-objective optimization, continuation, k-manifolds

Thornton, C., Hutter, F., Hoos, H. H. & Leyton-Brown, K. (2013). Auto-WEKA: Combined selection and hyperparameter optimization of classification algorithms. *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, 847-855 (vid. pág. 8).

Zhang, Q. & Li, H. (2007). MOEA/D: A multiobjective evolutionary algorithm based on decomposition. *IEEE Transactions on evolutionary computation*, *11*(6), 712-731 (vid. pág. 7).

Žilinskas, A. (2013). On the worst-case optimal multi-objective global optimization. *Optimization Letters*, *7*(8), 1921-1928 (vid. pág. 6).

Zitzler, E. & Thiele, L. (1999). Multiobjective evolutionary algorithms: a comparative case study and the strength Pareto approach. *IEEE transactions on Evolutionary Computation*, *3*(4), 257-271 (vid. pág. 7).