Búsqueda de vecindad variable multi-objetivo para la optimización de la mantenibilidad software

Javier Yuste[0000−0002−5956−9977] and Eduardo G. Pardo[0000−0002−6247−5269]

GRAFO Research, Departamento de Informática y Estadística,
Universidad Rey Juan Carlos, C/Tulipán, s/n, 28933, Spain
{javier.yuste,eduardo.pardo}@urjc.es

Resumen Este trabajo se centra en dos problemas de optimización multi-objetivo de ingeniería del *software* estrechamente relacionados: el *Maximizing Cluster Approach* y el *Equal-size Cluster Approach*. Cada uno de estos problemas busca la optimización de cinco métricas diferentes relacionadas con la modularidad del *software*. Se propone un método basado en la metodología de Búsqueda de Vecindad Variable Multi-objetivo en combinación con un procedimiento constructivo basado en Path-Relinking. Además, se presentan tres estrategias para mejorar la eficiencia del método: una evaluación incremental de las funciones objetivo, una exploración eficiente de áreas prometedoras y una reducción de las funciones guía utilizadas durante la búsqueda. La propuesta ha sido validada experimentalmente mediante comparación con el mejor método previo del estado del arte para el problema y tres algoritmos de referencia para optimización multi-objetivo, utilizando un conjunto de 124 instancias reales de *software* previamente reportadas en la literatura. Los resultados muestran que el enfoque propuesto obtiene mejores resultados que el estado del arte.

Keywords: Search-Based Software Engineering · Modularización · Heurísticas · Optimización multi-objetivo · Búsqueda de Vecindad Variable

1. Introducción

La mantenibilidad desempeña un papel central en el ciclo de vida de los sistemas *software*, donde las actividades de mantenimiento pueden representar más del 80 % de los costes totales [7]. Una práctica común para mejorar la mantenibilidad es agrupar archivos o clases en módulos con alta cohesión interna y bajo acoplamiento externo. Este escenario se puede modelar y abordar como un problema de optimización [6]. Desgraciadamente, se trata de un problema NP-difícil [2]. Además, existen varias propiedades deseables que entran en conflicto entre sí, convirtiéndolo en un problema de optimización multi-objetivo. Entre las formulaciones existentes, el *Maximizing Cluster Approach* (MCA) y el *Equal-size Cluster Approach* (ECA) son las variantes más estudiadas [8].

Este artículo describe brevemente un trabajo publicado recientemente en la revista *Engineering Applications of Artificial Intelligence* [9], donde se propone un método basado en el esquema *Multi-Objective General Variable Neighborhood Search* (MO-GVNS) con un procedimiento constructivo basado en *Path-Relinking*. Además, se incluyen tres estrategias en la propuesta: una evaluación incremental de las funciones objetivo, una reducción del tamaño de los vecindarios y una reducción del conjunto de funciones guía. El método propuesto ofrece mejores resultados que el algoritmo previo del estado del arte, un procedimiento *Two-Archive Artificial Bee Colony* (TA-ABC) [1], y que tres métodos de referencia bien conocidos: NSGA-III [4], PESA2 [3] y MOEA/D [10].

2. Definición del problema

Un sistema *software* se modela como un grafo no dirigido y ponderado donde los vértices corresponden a componentes *software* y las aristas representan dependencias. Los pesos de las aristas reflejan la intensidad de dichas dependencias. Una solución es un agrupamiento de los vértices en módulos disjuntos. Los problemas MCA y ECA consideran cinco de los siguientes seis objetivos [8]: (1) **Cohesión** (a maximizar), que es igual a la suma de los pesos de las aristas entre vértices que pertenecen al mismo módulo; (2) **Acoplamiento** (a minimizar), que equivale a la suma de los pesos de las aristas entre vértices que pertenecen a distintos módulos; (3) **Número de módulos** (a maximizar), que corresponde al número de módulos disjuntos; (4) **Calidad de modularización** (a maximizar), calculada mediante la formulación TurboMQ, que agrega la calidad de cada módulo como un compromiso entre cohesión y acoplamiento [6]; (5) **Número de módulos aislados** (a minimizar), que equivale al número de módulos que contienen un único vértice; (6) **Desbalance en el tamaño de los módulos** (a minimizar), que corresponde a la diferencia entre el máximo y el mínimo número de vértices en los módulos de la solución. Mientras que el problema MCA optimiza los objetivos del 1 al 5, la formulación ECA considera los objetivos 1, 2, 3, 4 y 6.

3. Propuesta algorítmica

El método propuesto se basa en el esquema MO-GVNS [5]. El conjunto inicial de soluciones se genera utilizando un procedimiento inspirado en *Path-Relinking*. Primero se construyen dos soluciones triviales: una en la que cada componente se encuentra en un módulo separado y otra en la que todos los componentes se agrupan en un mismo módulo. A continuación, se genera un conjunto de soluciones intermedias no dominadas fusionando progresivamente módulos de manera voraz.

Para escapar de óptimos locales, un procedimiento de perturbación realiza operaciones de intercambio. Se estudian varias variantes que combinan estrategias aleatorias y voraces. Para las búsqueda locales, se proponen cuatro vecindarios diferentes basadas en operadores de intercambio, inserción, destrucción

y extracción. El operador de intercambio intercambia el módulo de dos vértices distintos. El operador de inserción reubica un vértice en un módulo diferente. El operador de destrucción reubica todos los vértices de un módulo en otros módulos. El operador de extracción selecciona algunos vértices de distintos módulos para formar uno nuevo. Mientras que los dos primeros operadores no modifican el número de módulos, el tercero y el cuarto están diseñados para reducirlo y aumentarlo, respectivamente. Esto permite al método impulsar las soluciones en diferentes direcciones del espacio de objetivos.

Finalmente, se presentan tres estrategias principales: una evaluación incremental para recalcular parcialmente las funciones objetivo tras cada movimiento; una identificación de movimientos prometedores basada en la adyacencia de vértices y en criterios específicos de cada objetivo para reducir el tamaño de los vecindarios; y una reducción del número de funciones guía para disminuir el tiempo de cómputo con una pérdida mínima de calidad. Estas estrategias reducen el tiempo de ejecución hasta en un 70 % manteniendo la calidad de las soluciones.

4. Resultados experimentales

La propuesta se ha evaluado utilizando 124 instancias reales propuestas en la literatura [7]. La Tabla 1 presenta los resultados obtenidos para el problema MCA considerando indicadores de calidad estándar en optimización multiobjetivo: el número de soluciones no dominadas (PFS), el hipervolumen (HV), la cobertura (C), IGD+ y la diversidad generalizada (GS). Los mejores resultados se presentan en negrita. Como se puede observar, el método propuesto (MO-GVNS) obtiene resultados con mejor convergencia, mayor diversidad y mayor número de soluciones en menor tiempo que los algoritmos del estado del arte. Los resultados para el problema ECA son similares.

Tabla 1. Comparación del MO-GVNS propuesto con los métodos del estado del arte para el problema MCA.

Método	$\overline{\text{CPUt (s)}}$	$\overline{\text{PFS}}$	$\overline{\text{HV}}$	$\overline{\text{C}}$	$\overline{\text{IGD+}}$	$\overline{\text{GS}}$
MO-GVNS	**311.18**	**507.60**	**0.2213**	**0.0264**	**0.0443**	**0.5175**
MOEA/D [10]	336.43	300.00	0.0982	0.5254	0.2184	0.6844
NSGA-III [4]	596.41	479.56	0.0937	0.2719	0.2587	0.5891
PESA2 [3]	643.16	99.63	0.0604	0.7335	0.3209	0.6994
TA-ABC [1]	1037.31	97.70	0.0277	0.2145	0.3991	0.8026

5. Conclusiones

Este trabajo presenta un método basado en el esquema MO-GVNS para el problema de la optimización de la modularidad en sistemas *software*. El enfoque

propuesto, junto con vecindades y estrategias para el problema, logra mejores resultados que los enfoques existentes en el estado del arte. Como trabajo futuro, se plantea la incorporación de métricas de calidad adicionales en el problema, así como la exploración de métodos híbridos que combinen MO-GVNS con técnicas evolutivas. Además, se discuten posibles mejoras generales del esquema MO-GVNS.

Agradecimientos Esta investigación ha sido parcialmente financiada por los proyectos: PID2024-160226OB-C22 y PID2021-125709OA-C22, de MCIN/AEI y ERDF; CIRMA-CM TEC-2024/COM-404, de la Comunidad Autónoma de Madrid; TSI-100930-2023-3; "Cátedra AI4DDS: Artificial Intelligence for Data Drive Solutions" (Ref. ID MCA07); "Cátedra de Innovación y Digitalización Empresarial entre Universidad Rey Juan Carlos y Second Episode"; "Cátedra Institucional de Transición Energética" (Ref. ID MCA09); PLEC2023-010266; PID2022-142964OA-I00; y RED2022-134647-T.

Referencias

1. Amarjeet, Chhabra, J.K.: TA-ABC: Two-Archive Artificial Bee Colony for Multi-objective Software Module Clustering Problem. Journal of Intelligent Systems **27**(4), 619–641 (2018)
2. Brandes, U., Delling, D., Gaertler, M., Gorke, R., Hoefer, M., Nikoloski, Z., Wagner, D.: On modularity clustering. IEEE Transactions on Knowledge and Data Engineering **20**(2), 172–188 (2007)
3. Corne, D.W., Jerram, N.R., Knowles, J.D., Oates, M.J.: Pesa-ii: Region-based selection in evolutionary multiobjective optimization. In: Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2001). pp. 283–290 (2001)
4. Deb, K., Jain, H.: 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 (2013)
5. Duarte, A., Pantrigo, J.J., Pardo, E.G., Mladenovic, N.: Multi-objective variable neighborhood search: an application to combinatorial optimization problems. Journal of Global Optimization **63**(3), 515–536 (2015)
6. Mancoridis, S., Mitchell, B.S., Rorres, C., Chen, Y.F., Gansner, E.R.: Using automatic clustering to produce high-level system organizations of source code. In: 6th International Workshop on Program Comprehension (IWPC'98). pp. 45–52. IEEE (1998)
7. Monçores, M.C., Alvim, A.C.F., Barros, M.O.: Large neighborhood search applied to the software module clustering problem. Computers & Operations Research **91**, 92–111 (2018)
8. Praditwong, K., Harman, M., Yao, X.: Software module clustering as a multi-objective search problem. IEEE Transactions on Software Engineering **37**(2), 264–282 (2011)
9. Yuste, J., Pardo, E.G., Duarte, A., Hao, J.K.: Multi-objective general variable neighborhood search for software maintainability optimization. Engineering Applications of Artificial Intelligence **133**, 108593 (2024)
10. Zhang, Q., Li, H.: MOEA/D: A multiobjective evolutionary algorithm based on decomposition. IEEE Transactions on Evolutionary Computation **11**(6), 712–731 (2007)