Evolver: Un Framework de Meta-Optimización para la Configuración Automática de Metaheurísticas Multiobjetivo

Antonio J. Nebro[1] and Nicolás R. Uribe[2]

[1] Dept. de Lenguajes y Ciencias de la Computación, ITIS Software, Universidad de Málaga,
ETSI Informática, Campus de Teatinos, 29071 Málaga, España
ajnebro@uma.es
[2] Dept. de Ciencias de la Computación, Universidad Rey Juan Carlos, 28933 Móstoles, España
nicolas.rodriguez@urjc.es

Abstract. Evolver es un framework en Java para la configuración automática de metaheurísticas multiobjetivo. A diferencia de herramientas como irace o paramILS, Evolver formula el ajuste de parámetros como un problema de optimización multiobjetivo que se resuelve con otra metaheurística. Cada solución del meta-optimizador representa una configuración del algoritmo base, cuya evaluación consiste en ejecutarlo sobre un conjunto de problemas de entrenamiento usando indicadores de calidad como funciones objetivo. Construido sobre el framework jMetal, admite múltiples metaheurísticas configurables tanto en el nivel base como en el meta-optimizador. En este artículo describimos sus componentes principales e ilustramos su uso en un caso de estudio donde EvMOEA/D, una variante de MOEA/D configurada automáticamente con Evolver, supera significativamente a MOEA/D y MOEA/D-DE en todos los problemas del benchmark DTLZ. Se discuten también las capacidades de la herramienta, sus aplicaciones y líneas de investigación que se abren.

Keywords: Optimización Multiobjetivo · Metaheurísticas · Herramientas Software · Configuración Automática de Metaheurísticas

1 Introducción

La aplicación de metaheurísticas a la optimización multiobjetivo ha sido un área de investigación activa durante los últimos 25 años, especialmente tras la introducción de NSGA-II [4], SPEA2 [19] y PAES [9] en torno al año 2000. Desde entonces, los avances continuos han dado lugar al desarrollo de nuevos algoritmos diseñados para abordar dominios de problemas cada vez más complejos, como la optimización de problemas con muchos objetivos, dinámicos o con un elevado número de variables. Un desafío fundamental en este campo es que el rendimiento de las metaheurísticas es altamente sensible a la configuración de

sus parámetros de control, y no existe una configuración universalmente óptima para todos los problemas [17]. Tradicionalmente, este proceso de ajuste se ha apoyado en métodos manuales de prueba y error, que carecen de rigor científico y resultan especialmente difíciles para los expertos del dominio que no están familiarizados con los algoritmos de optimización.

Para abordar esta limitación, se han desarrollado herramientas de configuración automática de algoritmos como irace [12] y paramILS [8]. Estas herramientas optimizan la configuración de parámetros generando y evaluando configuraciones de forma iterativa sobre un conjunto de entrenamiento de problemas seleccionados y aplicando estrategias de aprendizaje para refinar las configuraciones, identificando finalmente configuraciones de alto rendimiento. Una vez finalizada la fase de entrenamiento, la configuración seleccionada se valida en un conjunto de test independiente para evaluar su capacidad de generalización. Sin embargo, estas herramientas no están orientadas a ajustar los parámetros de algoritmos multiobjetivo.

En este contexto se encuadra Evolver [1], un paquete en Java para la configuración automática de metaheurísticas multiobjetivo[3]. Evolver formula el propio proceso de ajuste como un problema de optimización multiobjetivo que se resuelve usando otra metaheurística, lo que lo define como un sistema de meta-optimización. Está construido sobre el framework jMetal [6, 14], que proporciona una variada colección de metaheurísticas, problemas de benchmark e indicadores de calidad.

Las contribuciones de este artículo son: (i) una descripción detallada de los componentes de Evolver y su espacio de configuración; (ii) un caso de estudio en el que la configuración automática de MOEA/D produce una variante (EvMOEA/D) que supera significativamente a MOEA/D y MOEA/D-DE en el benchmark DTLZ; y (iii) una discusión de aplicaciones y líneas de investigación abiertas por esta herramienta.

El resto del artículo se organiza como sigue: la Sección 2 describe el enfoque de meta-optimización de Evolver; la Sección 3 presenta el caso de estudio; la Sección 4 discute las capacidades, aplicaciones y líneas de investigación abiertas; y la Sección 5 recoge las conclusiones y el trabajo futuro.

2 Enfoque de Meta-Optimización

El enfoque de meta-optimización adoptado por Evolver guarda similitudes con la propuesta presentada en [16], que sigue el esquema mostrado en la Figura 1. El objetivo es encontrar la mejor configuración de una metaheurística multiobjetivo base para resolver eficientemente un conjunto de problemas, que denominamos conjunto de entrenamiento. Esta idea se formula como un problema de meta-optimización, donde el espacio de soluciones está compuesto por configuraciones del algoritmo para el optimizador de nivel base y el espacio objetivo está definido por un conjunto de indicadores de calidad que deben minimizarse.

[3] Repositorio de Evolver en GitHub: `https://github.com/jMetal/Evolver`

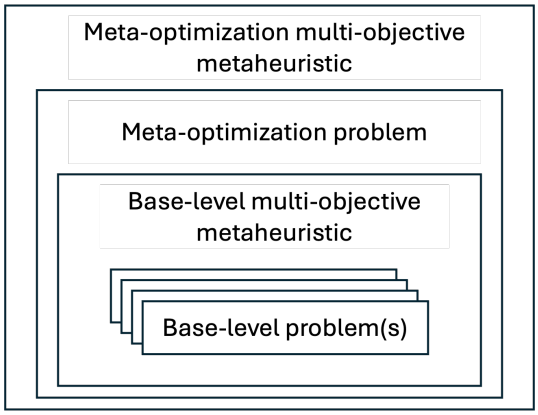


Fig. 1. Enfoque de meta-optimización utilizado en Evolver.

2.1 Metaheurísticas Multiobjetivo de Nivel Base

Aunque las metaheurísticas multiobjetivo clásicas como NSGA-II o MOEA/D [18] tienen pocos parámetros canónicos más allá de sus mecanismos principales, existen múltiples opciones configurables adicionales. Por ejemplo, el espacio completo de parámetros del NSGA-II de nivel base para problemas continuos en Evolver comprende hasta 18 parámetros configurables. Evolver también incluye implementaciones altamente configurables de MOEA/D, SMS-EMOA [2] y un optimizador multiobjetivo de enjambre de partículas (MOPSO) [15]. Con la excepción de MOPSO, estos algoritmos también pueden aplicarse a problemas binarios y de permutaciones.

2.2 Espacios de Parámetros

Los espacios de parámetros en Evolver definen el conjunto completo de parámetros que pueden configurarse de forma automática para las metaheurísticas de nivel base. Los parámetros pueden ser categóricos, como el operador de cruce en los algoritmos evolutivos, o numéricos, como la probabilidad de mutación (parámetro real) o el tamaño de la población de descendientes (parámetro entero). Además, cada parámetro tiene un nombre, una definición de valores válidos y relaciones opcionales con otros parámetros: parámetros condicionales, que solo se aplican cuando un parámetro categórico padre toma un valor específico, y subparámetros globales, que siempre se aplican independientemente del valor del parámetro padre.

El espacio de parámetros en Evolver se describe mediante ficheros YAML. El Listado 1.1 muestra un extracto que ilustra el parámetro `selection`, típico de un algoritmo evolutivo, que tiene dos opciones (`tournament` y `random`) y que incluye un parámetro condicional (`selectionTournamentSize`) que solo se aplica cuando se elige la selección por torneo.

Listing 1.1. Extracto del espacio de parámetros de NSGA-II en YAML (parámetro de selección).

```yaml
# ... (otros parametros omitidos por brevedad)
selection:
  type: categorical
  values:
    tournament:
      conditionalParameters:
        selectionTournamentSize:
          type: integer
          range: [2, 10]
    random: {}
# ...
```

2.3 Codificación de Soluciones

A diferencia del esquema utilizado en [16], donde se emplea una codificación en árbol para representar las soluciones del problema de meta-optimización, en Evolver todos los parámetros se codifican en un vector de valores reales en el rango $[0.0, 1.0]$. Esto implica que, en caso de usar el NSGA-II como algoritmo base, cada solución es un vector de 18 números reales. Evaluar una solución requiere decodificar los valores reales en los parámetros correspondientes para configurar el NSGA-II base y ejecutarlo sobre los problemas de nivel base.

El esquema de codificación adoptado es simple y tiene la ventaja de que cualquier algoritmo multiobjetivo de jMetal capaz de resolver problemas continuos puede usarse como meta-optimizador. Sin embargo, existen dos aspectos a tener en cuenta. En primer lugar, todos los parámetros se aplanan sin considerar restricciones, por lo que valores de parámetros inactivos también son decodificados. En segundo lugar, codificar parámetros booleanos y categóricos en $[0.0, 1.0]$ puede provocar que una mutación no altere el valor decodificado. Ambos efectos pueden mitigarse aumentando la probabilidad del operador de mutación del meta-optimizador.

2.4 Evaluación de Soluciones y Funciones Objetivo

Dada una solución generada por los operadores de variación del algoritmo meta-optimizador, evaluarla implica ejecutar la metaheurística base sobre los P problemas del conjunto de entrenamiento. Dado que las metaheurísticas son técnicas estocásticas, se deben realizar N ejecuciones independientes por cada combinación $\langle$configuración, problema$\rangle$. A continuación, se aplica cada indicador de calidad elegido a los frentes resultantes de las N ejecuciones independientes. Los objetivos se calculan en dos pasos:

1. Para cada problema, se calcula la mediana de los indicadores de calidad para las N ejecuciones.
2. Para cada indicador de calidad, la media de las medianas de cada problema es el valor objetivo resultante.

N es un parámetro del problema de meta-optimización y debe fijarse con cuidado, ya que puede tener un alto impacto en el tiempo total de ejecución del proceso de meta-optimización. Por defecto, N es igual a 1.

jMetal proporciona una amplia gama de indicadores de calidad que miden el grado de convergencia y/o diversidad de una aproximación al frente de Pareto obtenida por una metaheurística multiobjetivo, como el épsilon aditivo (EP), la distancia generacional invertida (IGD) o el hipervolumen (HV) [20]. Las funciones objetivo del problema de meta-optimización son indicadores de calidad elegidos entre los disponibles en jMetal.

Todos los indicadores de calidad utilizados como funciones objetivo están pensados para minimizarse en Evolver. Por lo tanto, se requiere especial cuidado al seleccionar el HV como objetivo, ya que representa un volumen a maximizar. En lugar del HV, puede sustituirse por el hipervolumen normalizado (NHV), definido como $1-HV_f/HV_{rf}$, donde HV_f es el HV de una aproximación al frente de Pareto y HV_{rf} es el HV del frente de referencia utilizado para calcularlo. Los valores de NHV oscilan entre 0.0 y 1.0, siendo los valores más bajos los que indican un mejor rendimiento.

En Evolver, los usuarios pueden seleccionar cualquier combinación de indicadores de calidad. Sin embargo, para que un problema de optimización se considere multiobjetivo, sus objetivos deben ser conflictivos. Esta condición no siempre se cumple al usar diferentes indicadores de calidad. Por ejemplo, seleccionar EP e IGD no garantiza que mejorar EP empeore IGD. No obstante, los indicadores de calidad pueden ser parcialmente conflictivos, permitiendo que emerjan algunas soluciones no dominadas (véase, por ejemplo, la Figura 2 en la sección siguiente). El punto clave es que el objetivo del meta-optimizador no es generar un conjunto diverso de soluciones, sino converger rápida y precisamente hacia configuraciones óptimas de la metaheurística base.

2.5 Metaheurísticas Multiobjetivo del Meta-Optimizador

Como se ha mencionado anteriormente, elegir una codificación real para el meta-optimizador permite utilizar la mayoría de las metaheurísticas multiobjetivo disponibles en jMetal, incluyendo algoritmos evolutivos (NSGA-II, MOEA/D, SMS-EMOA, SPEA2, etc.), evolución diferencial (GDE3 [10], MOEA/D-DE [11]) y algoritmos de optimización por enjambre de partículas como SMPSO [13].

Algunos de estos algoritmos pueden evaluar la población en paralelo mediante un esquema paralelo síncrono para acelerar la ejecución. Para NSGA-II, también está disponible una versión paralela asíncrona que permite un aprovechamiento más eficiente de los núcleos disponibles [7].

3 Caso de Estudio

En esta sección ilustramos el uso de Evolver en un estudio en el que la metaheurística base es MOEA/D, los problemas de entrenamiento son el benchmark DTLZ [5] y el meta-optimizador es la versión paralela asíncrona de NSGA-II.

Table 1. Espacio de parámetros de MOEA/D en Evolver. Los parámetros en cursiva son condicionales. Acrónimos: WS: weightedSum; PBI: penaltyBoundaryIntersection; MT: modifiedTschebyscheff; LHS: latinHypercubeSampling; CM: crossoverAndMutation; LP: linkedPolynomial; NU: nonUniform; WA: wholeArithmetic; DE: differentialEvolution.

Componente	**Valor / Dominio**	**Parámetros Condicionales**
General	neighborhoodSize $\in [10, 200]$	–
	maximumNumberOfReplacedSolutions $\in [1, 5]$	–
	neighborhoodSelectionProbability $\in [0.0, 1.0]$	–
	aggregationFunction $\in$ {tschebyscheff, WS, PBI, MT}	–
	normalizeObjectives $\in$ {TRUE, FALSE}	–
	epsilonForNormalizing $\in$ [1e-8, 25.0]	normalizeObjectives = TRUE
	pbiTheta $\in [1.0, 200.0]$	aggregationFunction = PBI
Archivo	algorithmResult $\in$ {population, externalArchive}	–
	externalArchive $\in$ {crowdingDistance, unbounded}	algorithmResult = externalArchive
Inicialización	createInitialSolutions $\in$ {random, LHS, scatterSearch}	–
Variación	variation $\in$ {CM, DE}	–
	crossover $\in$ {SBX, BLX_ALPHA, WA}	variation = CM
	crossoverProbability $\in [0.0, 1.0]$	variation = CM
	crossoverRepairStrategy $\in$ {random, round, bounds}	variation = CM
	sbxDistributionIndex $\in [5.0, 400.0]$	crossover = SBX
	blxAlphaCrossoverAlphaValue $\in [0.0, 1.0]$	crossover = BLX_ALPHA
	differentialEvolutionScheme $\in$ {RAND-1-BIN, RAND-1-EXP, RAND-2-BIN}	variation = DE
	CR, *F* $\in [0.0, 1.0]$	variation = DE
Mutación	mutation $\in$ {uniform, polynomial, LP, NU}	–
	mutationProbabilityFactor $\in [0.0, 2.0]$	–
	mutationRepairStrategy $\in$ {random, round, bounds}	–
	polynomialMutationDistributionIndex $\in [5.0, 400.0]$	mutation $\in$ {polynomial, LP}
	uniformMutationPerturbation $\in [0.0, 1.0]$	mutation = uniform
	nonUniformMutationPerturbation $\in [0.0, 1.0]$	mutation = NU

El objetivo es comprobar si Evolver es capaz de encontrar una configuración para los problemas DTLZ que mejore los resultados del algoritmo MOEA/D original [18] y la variante MOEA/D-DE [11]. El espacio de parámetros de la implementación de MOEA/D de nivel base en Evolver (denominada EvMOEA/D a partir de ahora) se detalla en la Tabla 1.

En este estudio nos fijamos en dos aspectos. Por un lado, nos interesa obtener configuraciones de EvMOEA/D orientadas a producir frentes con un alto grado de convergencia y diversidad, por lo que hemos elegido el NHV como objetivo primario. Complementamos el NHV con el indicador EP para guiar también la búsqueda hacia configuraciones que promuevan la convergencia. En segundo lugar, queremos determinar si, partiendo de un bajo número de evaluciones al entrenar (en concreto, se van a usar 16.000 evaluaciones en el algoritmo base), se consiguen buenos resultados al validar con los mismos problemas pero con 40.000 evaluaciones, que es un valor habitual al resolver los DTLZ.

Los parámetros fijos del EvMOEA/D de nivel base son tamaño de población de 100 individuos y 1 ejecución independiente por configuración. Como los problemas DTLZ tienen por defecto tres objetivos, MOEA/D generalmente utiliza 91 vectores de pesos, pero hemos usado el generador propuesto en [3] para producir 100 vectores de pesos uniformemente espaciados. El meta-optimizador NSGA-II asíncrono se configuró con tamaño de población de 50 individuos, condición de parada de 2.000 evaluaciones de la función objetivo, cruce SBX (probabili-

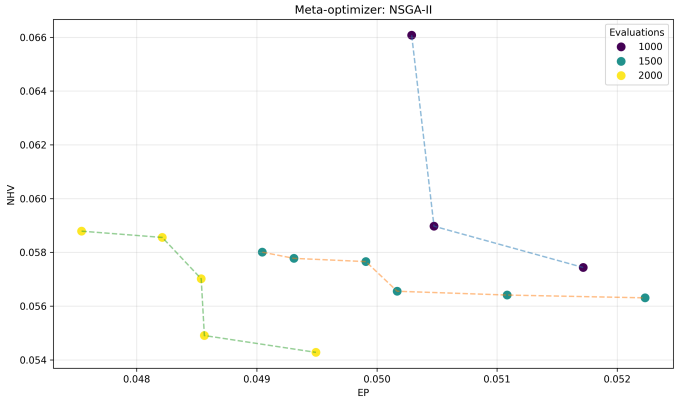


Fig. 2. Evolución del frente de soluciones (configuraciones de EvMOEA/D) producido por el meta-optimizador NSGA-II asíncrono en la búsqueda de la mejor configuración de EvMOEA/D sobre el benchmark DTLZ.

dad: 0.9, índice de distribución: 20.0) y mutación polinomial (probabilidad: $1/n$, índice de distribución: 20.0), donde n es el número de variables de decisión del problema de meta-optimización. Hemos ejecutado Evolver con estos ajustes en un ordenador portátil MacBook Pro con procesador Apple M1 Max, 32 GB de RAM y macOS Sequoia 15.3.2. Configurando el número de núcleos del NSGA-II asíncrono en 8, el tiempo total de cómputo fue de aproximadamente tres horas.

La Figura 2 muestra cómo evoluciona el frente de soluciones obtenido por NSGA-II hasta alcanzar la condición de parada. Se puede observar que durante la búsqueda se encuentran conjuntos de entre tres y seis soluciones no dominadas. Del conjunto final de soluciones, seleccionamos la solución con el valor NHV más bajo, al ser el NHV el objetivo principal. La configuración correspondiente a esta solución se incluye en la Tabla 2, junto con la configuración por defecto de MOEA/D y MOEA/D-DE.

En la Tabla 2 se observa que EvMOEA/D, como MOEA/D-DE, emplea evolución diferencial y utiliza la función de escalarización de Tschebyscheff. Sin embargo, a diferencia de MOEA/D-DE, incorpora un archivo externo no acotado, normaliza los objetivos, aplica mutación uniforme y utiliza valores de CR y F de 0.13 y 0.98, respectivamente, en lugar de 1.0 y 0.5.

Una vez finalizada la meta-optimización de EvMOEA/D, el siguiente paso es validar la configuración encontrada. Para ello, realizamos un estudio comparativo de las tres variantes de MOEA/D sobre el benchmark DTLZ, estableciendo la condición de parada en 40.000 evaluaciones (en lugar de las 16.000 usadas en la meta-optimización), que es un valor típico. Utilizamos el indicador NHV para evaluar el rendimiento de los algoritmos.

La Tabla 3 muestra que EvMOEA/D obtiene el mejor rendimiento en todos los problemas (fondo gris oscuro). El test de Wilcoxon confirma que las diferencias son estadísticamente significativas en los siete casos.

Table 2. Configuración de las variantes de MOEA/D. CM: crossoverAndMutation; DE: differentialEvolution.

Parámetro	**MOEA/D**	**MOEA/D-DE**	**EvMOEA/D**
neighbourhoodSize	20	20	34
maximumNumberOfReplacedSolutions	2	2	3
neighborhoodSelectionProbability	0.9	0.9	0.0
aggregationFunction	PBI	Tschebyscheff	Tschebyscheff
normalizeObjectives	False	False	True
epsilonForNormalizing	N/A	N/A	12.99
pbiTheta	5.0	N/A	N/A
algorithmResult	Population	Population	ExternalArchive
externalArchive	N/A	N/A	Unbounded
variation	CM	DE	DE
crossover	SBX	N/A	N/A
crossoverProbability	0.9	N/A	N/A
crossoverRepairStrategy	Bounds	N/A	N/A
sbxDistributionIndex	20.0	N/A	N/A
differentialEvolutionScheme	N/A	RAND-1-BIN	RAND-1-BIN
CR	N/A	1.0	0.13
F	N/A	0.5	0.98
mutation	Polynomial	Polynomial	Uniform
mutationProbabilityFactor	1.0	1.0	0.58
mutationRepairStrategy	Bounds	Bounds	Random
polynomialMutationDistributionIndex	20.0	20.0	N/A
uniformMutationPerturbation	N/A	N/A	0.27

Table 3. Mediana y Rango Intercuartílico de los valores del hipervolumen normalizado (NHV) obtenidos por los algoritmos. Los símbolos + y − en las celdas indican que el algoritmo de referencia (última columna) es estadísticamente mejor y peor, respectivamente, según el test de Wilcoxon de suma de rangos; el símbolo = indica que las diferencias no son significativas.

	MOEA/D	MOEA/D-DE	EvMOEA/D
DTLZ1	$4.78e{-2}_{2.39e-3}+$	$1.01e{-1}_{2.90e-3}+$	$4.58e{-2}_{7.46e-4}$
DTLZ2	$1.11e{-1}_{8.96e-5}+$	$1.92e{-1}_{4.28e-3}+$	$1.10e{-1}_{3.54e-3}$
DTLZ3	$2.14e{-1}_{6.98e-2}+$	$2.21e{-1}_{8.76e-2}+$	$1.05e{-1}_{1.44e-2}$
DTLZ4	$2.90e{-2}_{4.88e-1}+$	$1.06e{-1}_{1.49e-2}+$	$2.84e{-2}_{3.22e-3}$
DTLZ5	$1.14e{-1}_{9.56e-4}+$	$4.61e{-2}_{9.05e-4}+$	$2.14e{-2}_{8.19e-4}$
DTLZ6	$9.56e{-1}_{3.20e-2}+$	$2.97e{-2}_{8.52e-5}+$	$9.53e{-3}_{7.07e-4}$
DTLZ7	$2.40e{-1}_{7.94e-3}+$	$4.27e{-1}_{3.78e-2}+$	$5.51e{-2}_{3.29e-3}$
+ / - / =	**7 / 0 / 0**	**7 / 0 / 0**	

Para concluir el estudio, la Figura 3 presenta las aproximaciones al frente de Pareto correspondientes a los valores medianos de NHV de las tres variantes de MOEA/D para los problemas DTLZ3, DTLZ5 y DTLZ7. Los frentes muestran que EvMOEA/D encuentra con éxito puntos uniformemente distribuidos en los tres problemas, mientras que MOEA/D tiene dificultades con las instancias DTLZ5 y DTLZ7, y MOEA/D-DE enfrenta problemas con DTLZ1 y DTLZ7.

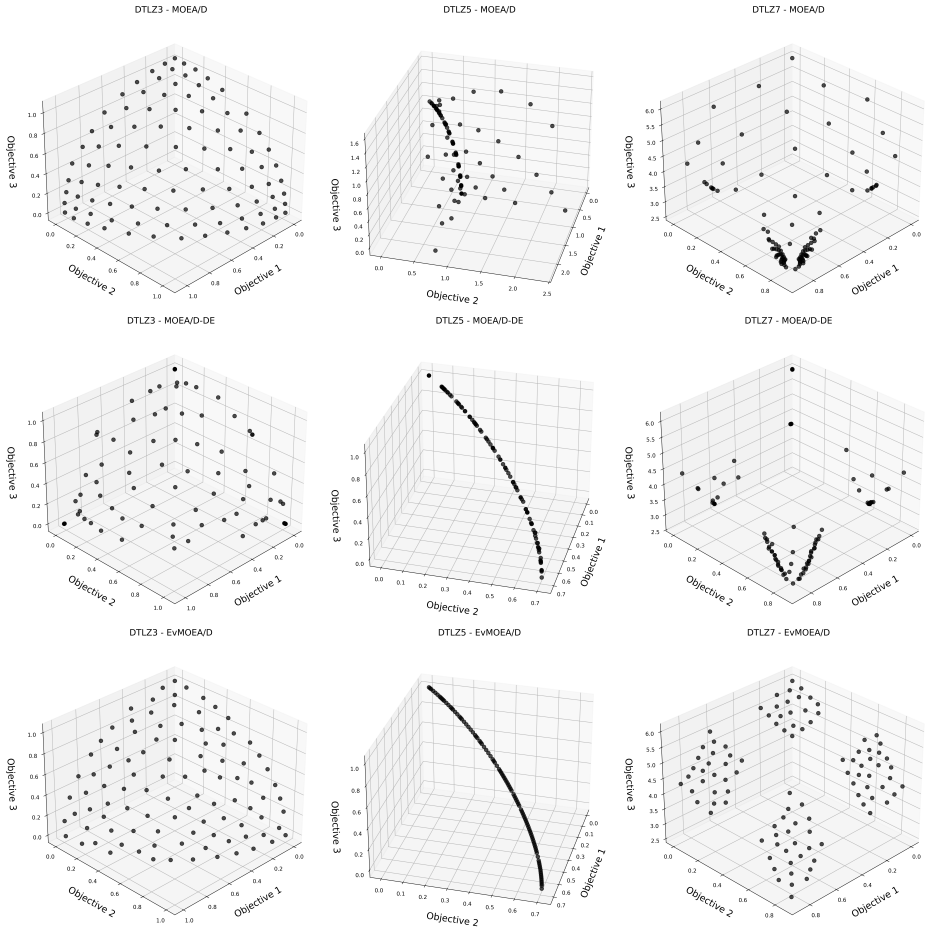


Fig. 3. Aproximaciones al frente de Pareto, correspondientes al valor mediano del NHV, encontradas por MOEA/D (fila 1), MOEA/D-DE (fila 2) y EvMOEA/D (fila 3) para los problemas DTLZ3, DTLZ5 y DTLZ7.

4 Discusión

4.1 Capacidades

Evolver es una herramienta que automatiza el ajuste de parámetros, eliminando el proceso manual de prueba y error mediante la optimización de los parámetros de algoritmos de metaheurísticas multiobjetivo.

Construido sobre el framework jMetal, Evolver proporciona acceso a una amplia gama de metaheurísticas multiobjetivo, problemas de benchmark e indicadores de calidad. Al ser un software autocontenido basado en Java, no requiere herramientas externas. Los usuarios de jMetal encontrarán Evolver fácil de utilizar, mientras que los investigadores con conocimientos básicos de Java pueden

implementar sus problemas multiobjetivo en jMetal y aprovechar Evolver para encontrar configuraciones casi óptimas sin necesidad de un conocimiento profundo de la metaheurística base.

4.2 Aplicaciones

A continuación se describen las principales aplicaciones de Evolver:

- **Optimización de Problemas del Mundo Real:** Resolver problemas de optimización del mundo real de distintos dominios (ingeniería, logística, finanzas, etc.) es un campo donde el uso de Evolver para ajustar finamente los algoritmos puede resultar especialmente útil. De hecho, el objetivo último de las metaheurísticas es resolver problemas reales en lugar de benchmarks sintéticos.
- **Benchmarking y Análisis de Algoritmos:** Evolver puede utilizarse para evaluar y comparar sistemáticamente distintos algoritmos multiobjetivo bajo configuraciones de parámetros optimizadas.

El enfoque de meta-optimización de Evolver puede resultar inicialmente difícil de aplicar en entornos reales, ya que los frentes de Pareto generalmente se desconocen, lo que hace impracticable usar indicadores de calidad como objetivos de optimización, pues la mayoría requieren un frente de referencia. La excepción es el hipervolumen (HV), que solo necesita un punto de referencia. En consecuencia, el HV puede servir como objetivo principal en el problema de meta-optimización. Sin embargo, dado que el HV representa un volumen cuyos valores más altos indican mejor rendimiento, y jMetal está diseñado para minimización, nuestro enfoque consiste en utilizar HV^- (es decir, HV multiplicado por -1). Utilizar únicamente HV^- puede provocar estancamiento, especialmente en las primeras iteraciones, cuando las configuraciones producen un HV^- de cero si ninguna solución en la población actual domina el punto de referencia. Para superar esta limitación, el indicador EP puede servir como objetivo secundario. Cuando todas las soluciones tienen un HV^- de cero, el meta-optimizador puede apoyarse en EP para refinar las configuraciones, mejorar la convergencia y alcanzar eventualmente regiones donde las soluciones estén por debajo del punto de referencia.

4.3 Líneas de Investigación

A continuación enumeramos varias líneas de investigación prometedoras que podrían surgir del uso de Evolver:

- **Ajuste Multi-Fidelidad:** Usar aproximaciones de baja fidelidad (instancias de problemas reducidas o condiciones de parada bajas) para acelerar el ajuste sin sacrificar precisión, como sugieren nuestros resultados experimentales. Esta línea es especialmente relevante para problemas a gran escala y con muchos objetivos, donde puede aplicarse sobre versiones reducidas del problema.

– **Explicabilidad e Interpretabilidad en el Ajuste:** Desarrollar técnicas para analizar por qué ciertas configuraciones superan a otras, haciendo el proceso de ajuste más transparente e informativo.
– **Selección y Configuración Automática de Algoritmos:** Ejecutar varios algoritmos de nivel base simultáneamente para seleccionar conjuntamente la mejor metaheurística y su configuración optimizada para un dominio de problema dado.
– **Aprendizaje por Transferencia en Configuración de Algoritmos:** Transferir configuraciones de parámetros entre dominios de problemas relacionados para reducir el coste computacional de la meta-optimización.
– **Desarrollo de Benchmarks:** Crear escenarios de ajuste estandarizados (combinaciones de metaheurísticas de nivel base, problemas e indicadores de calidad) como suites de benchmark para comparaciones de meta-optimizadores.

5 Conclusiones y Trabajo Futuro

En este artículo hemos presentado Evolver, un framework en Java que formula el ajuste de parámetros de metaheurísticas multiobjetivo como un problema de optimización multiobjetivo resuelto con otra metaheurística sobre jMetal. Como primera contribución, se han descrito en detalle sus componentes principales y el espacio de configuración de las metaheurísticas de nivel base. Como segunda contribución, se ha ilustrado su uso en un caso de estudio donde EvMOEA/D, una variante de MOEA/D configurada automáticamente con Evolver, supera significativamente a MOEA/D y MOEA/D-DE en los siete problemas del benchmark DTLZ. Como tercera contribución, se han discutido las capacidades de la herramienta, sus aplicaciones en problemas reales y las líneas de investigación que abre.

El trabajo futuro sobre Evolver se centrará en mejorar la usabilidad simplificando la integración de nuevos problemas y algoritmos. Además, se están realizando esfuerzos para integrar más metaheurísticas multiobjetivo de nivel base y ampliar la configurabilidad de los algoritmos ya existentes.

Agradecimientos Este trabajo ha sido parcialmente financiado por los proyectos PID2024-155363OB-C41 (KOSMOS) – MCIN/AEI/10.13039/501100011033 – y DGP_PIDI_2024_01174 – Junta de Andalucia.

References

1. Aldana-Martín, J.F., Durillo, J.J., Nebro, A.J.: Evolver: Meta-optimizing multiobjective metaheuristics. SoftwareX **24**, 101551 (2023)
2. Beume, N., Naujoks, B., Emmerich, M.: SMS-EMOA: Multiobjective selection based on dominated hypervolume. European Journal of Operational Research **181**(3), 1653–1669 (2007)
3. Blank, J., Deb, K., Dhebar, Y., Bandaru, S., Seada, H.: Generating well-spaced points on a unit simplex for evolutionary many-objective optimization. IEEE Transactions on Evolutionary Computation **25**(1), 48–60 (2021)

4. Deb, K., Pratap, A., Agarwal, S., Meyarivan, T.: A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE Transactions on Evolutionary Computation **6**(2), 182–197 (2002)
5. Deb, K., Thiele, L., Laumanns, M., Zitzler, E.: Scalable multi-objective optimization test problems. In: Proceedings of the 2002 Congress on Evolutionary Computation. CEC'02 (Cat. No.02TH8600). vol. 1, pp. 825–830 vol.1 (2002)
6. Durillo, J., Nebro, A.: jMetal: A Java framework for multi-objective optimization. Advances in Engineering Software **42**(10), 760 – 771 (2011)
7. Durillo, J.J., Nebro, A.J., Luna, F., Alba, E.: A study of master-slave approaches to parallelize NSGA-II. In: 2008 IEEE International Symposium on Parallel and Distributed Processing. pp. 1–8 (2008)
8. Hutter, F., Hoos, H.H., Leyton-Brown, K., Stützle, T.: ParamILS: An automatic algorithm configuration framework. J. Artif. Int. Res. **36**(1), 267–306 (9 2009)
9. Knowles, J., Corne, D.: The pareto archived evolution strategy: a new baseline algorithm for pareto multiobjective optimisation. In: Proceedings of the 1999 Congress on Evolutionary Computation-CEC99 (Cat. No. 99TH8406). vol. 1, pp. 98–105 Vol. 1 (1999)
10. Kukkonen, S., Lampinen, J.: GDE3: the third evolution step of generalized differential evolution. In: 2005 IEEE Congress on Evolutionary Computation. vol. 1, pp. 443–450 Vol.1 (2005)
11. Li, H., Zhang, Q.: Multiobjective optimization problems with complicated pareto sets, MOEA/D and NSGA-II. IEEE Transactions on Evolutionary Computation **13**(2), 284–302 (2009)
12. López-Ibáñez, M., Dubois-Lacoste, J., Pérez Cáceres, L., uro Birattari, M., Stützle, T.: The irace package: Iterated racing for automatic algorithm configuration. Operations Research Perspectives **3**, 43–58 (2016)
13. Nebro, A., Durillo, J., García-Nieto, J., Coello Coello, C., Luna, F., Alba, E.: SMPSO: A new PSO-based metaheuristic for multi-objective optimization. In: IEEE Symposium on Computational Intelligence in Multicriteria Decision-Making (MCDM 2009). pp. 66–73. IEEE Press (2009)
14. Nebro, A.J., Durillo, J.J., Vergne, M.: Redesigning the jMetal multi-objective optimization framework. GECCO 2015 - Companion Publication of the 2015 Genetic and Evolutionary Computation Conference pp. 1093–1100 (7 2015)
15. Nebro, A.J., López-Ibáñez, M., García-Nieto, J., Coello, C.A.C.: On the automatic design of multi-objective particle swarm optimizers: experimentation and analysis. Swarm Intelligence **18**(2), 105–139 (2024)
16. Neumüller, C., Wagner, S., Kronberger, G., Affenzeller, M.: Parameter meta-optimization of metaheuristic optimization algorithms. In: Moreno-Díaz, R., Pichler, F., Quesada-Arencibia, A. (eds.) Computer Aided Systems Theory – EUROCAST 2011. pp. 367–374. Springer Berlin Heidelberg, Berlin, Heidelberg (2012)
17. Wolpert, D., Macready, W.: No free lunch theorems for optimization. IEEE Transactions on Evolutionary Computation **1**(1), 67–82 (1997)
18. Zhang, Q., Li, H.: MOEA/D: A multiobjective evolutionary algorithm based on decomposition. IEEE Transactions on Evolutionary Computation **11**(6), 712–731 (2007)
19. Zitzler, E., Thiele, L.: Multiobjective evolutionary algorithms: a comparative case study and the strength pareto approach. IEEE Trans. on Evol. Comp. **3**(4), 257–271 (1999)
20. Zitzler, E., Thiele, L., Laumanns, M., Fonseca, C.M., da Fonseca, V.G.: Performance assessment of multiobjective optimizers: an analysis and review. IEEE Trans. Evol. Comput. **7**(2), 117–132 (2003)