Evaluación Comparativa de Algoritmos Evolutivos Multi-Objetivo (MOEA) para el Problema del Viajante de Comercio Multi-Objetivo

Autores:

- Christian Daniel Fleitas Velaztiqui
- Satoshi Andres Nakamori Terabayashi

Resumen: Este trabajo presenta un estudio comparativo de dos algoritmos evolutivos multi-objetivo (MOEA), el Strength Pareto Evolutionary Algorithm (SPEA) y el Non-dominated Sorting Genetic Algorithm II (NSGA-II), aplicados al Problema del Viajante de Comercio (TSP) con dos objetivos en conflicto. Se describen la formulación del problema y los algoritmos implementados, incluyendo pseudocódigo. Los resultados experimentales, obtenidos a partir de múltiples ejecuciones en instancias estándar del TSP, son analizados utilizando métricas de rendimiento multi-objetivo como Spacing (M1), Diversity (M2), Coverage of Front (M3) y Error. Además, se propone un conjunto de configuraciones de parámetros para futuras experimentaciones, buscando investigar su impacto en la calidad del frente de Pareto aproximado.

Palabras Clave: Optimización Multi-Objetivo, Algoritmos Evolutivos, TSP, SPEA, NSGA-II, Metaheurísticas.

1. Introducción

La optimización multi-objetivo (MOO) representa un paradigma fundamental en la resolución de problemas complejos, donde múltiples criterios, a menudo en conflicto, deben ser considerados simultáneamente para identificar un conjunto de soluciones óptimas. A diferencia de la optimización de un solo objetivo, que busca un único óptimo global, la MOO se centra en la identificación de un **frente de Pareto** o **conjunto de soluciones Pareto-óptimas**, donde cada solución es no dominada por ninguna otra en todos los objetivos. Este enfoque es crucial en escenarios del mundo real, desde el diseño de ingeniería hasta la logística, donde las decisiones deben equilibrar factores como el costo, el tiempo, la calidad y la eficiencia.

Este trabajo se enfoca en la aplicación de algoritmos de optimización multi-objetivo basados en metaheurísticas para resolver una clase de problemas de optimización combinatoria: el Problema del Viajante de Comercio (TSP). Este problema presenta características combinatorias y complejidades computacionales que lo hacen ideal para la aplicación de enfoques metaheurísticos debido a la imposibilidad práctica de encontrar soluciones óptimas exactas en tiempos razonables para instancias de gran tamaño.

Se implementará y evaluará la capacidad de los **Algoritmos Evolutivos Multi-Objetivo (MOEA)**, específicamente el **Strength Pareto Evolutionary Algorithm (SPEA)** y el **Non-dominated Sorting Genetic Algorithm II (NSGA-II)**. Estos algoritmos serán aplicados a instancias estándar del problema TSP, formulado con dos funciones objetivo en conflicto. El objetivo principal es comparar la eficacia de estos enfoques metaheurísticos en

la aproximación del frente de Pareto para cada problema, utilizando métricas de rendimiento multi-objetivo estandarizadas.

La contribución de este estudio radica en la implementación práctica de estas metaheurísticas avanzadas y en un análisis comparativo de su desempeño en problemas combinatorios con múltiples objetivos, proporcionando una visión sobre su aplicabilidad y eficiencia en la generación de conjuntos de soluciones diversas y de alta calidad. Se detallará la formulación del problema TSP, la descripción de los algoritmos implementados y se presentarán los resultados experimentales obtenidos, analizados a través de métricas como M1, M2, M3 y Error, para ofrecer una evaluación exhaustiva del rendimiento de cada algoritmo.

2. Formulación del Problema: Problema del Viajante de Comercio Multi-Objetivo (MOTSP)

El Problema del Viajante de Comercio (TSP) es un problema clásico de optimización combinatoria donde un vendedor debe visitar un conjunto de N ciudades exactamente una vez, comenzando y terminando en la ciudad de origen, minimizando el costo total del viaje. En su versión multi-objetivo (MOTSP), se consideran dos o más criterios de optimización simultáneamente.

En este estudio, el MOTSP se formula con dos funciones objetivo a minimizar:

- **Objetivo 1 (f1):** El costo total del viaje basado en una matriz de adyacencia de costos.
- **Objetivo 2 (f2):** El tiempo total del viaje basado en una segunda matriz de adyacencia de tiempos.

La representación de una solución es una permutación de las ciudades, lo que define el orden de visita. Las instancias de problema se leen desde archivos de texto con el siguiente formato:

- Primera línea: Cantidad de ciudades (N).
- Segunda línea: Cantidad de objetivos (siempre 2 para este estudio).
- Matriz de adyacencia para el objetivo 1.
- Una línea en blanco.
- Matriz de adyacencia para el objetivo 2.

Para este trabajo, se utilizan las instancias `tsp_KROAB100.TSP.TXT` y `tsp_kroac100.tsp.txt`, que consisten en N=100 ciudades y dos matrices de costos/tiempos, respectivamente. Un verificador de restricciones simple asegura que la solución es una permutación válida y un ciclo cerrado.

3. Algoritmos MOEA Implementados

Se han implementado dos algoritmos evolutivos multi-objetivo (MOEA) para abordar el MOTSP: el Strength Pareto Evolutionary Algorithm (SPEA) y el Non-dominated Sorting Genetic Algorithm II (NSGA-II). Ambos algoritmos se basan en principios de la computación evolutiva, utilizando operadores de selección, cruce y mutación para explorar el espacio de soluciones y converger hacia el frente de Pareto.

3.1 Strength Pareto Evolutionary Algorithm (SPEA)

SPEA es un algoritmo evolutivo multi-objetivo que utiliza un archivo externo para almacenar las soluciones no dominadas encontradas durante la evolución. Su mecanismo de asignación de fitness se basa en la "fuerza" de las soluciones dominantes y un factor de densidad, lo que ayuda a preservar la diversidad.

Pseudocódigo de SPEA:

```
Algoritmo SPEA

Entrada:
    - Problema multiobjetivo
    - Tamaño de población P
    - Tamaño de archivo externo A_max
    - Número de generaciones G
    - Parámetros de crossover y mutación

1. Inicializar población P con soluciones aleatorias
2. Inicializar archivo externo A como vacío

Para generación = 1 hasta G hacer:
    3. Evaluar objetivos de todos los individuos en P y A
    4. Calcular Strength, Raw Fitness y Densidad para todos los
individuos en P ∪ A
    5. Construir nuevo archivo A_next:
        a. Copiar todos los individuos no dominados de P ∪ A a A_next
        b. Si |A_next| > A_max, truncar usando densidad (eliminar los más
densos)
        c. Si |A_next| < A_max, rellenar con dominados menos dominados
    6. Si A_next está vacío, terminar (no hay soluciones válidas)
    7. Seleccionar padres desde A_next (por torneo usando fitness)
    8. Generar nueva población P aplicando crossover y mutación a los
padres

Fin Para

9. Retornar el archivo externo A como aproximación del frente de Pareto

Fin Algoritmo

Notas:
Strength: Cuántos individuos domina cada solución.
Raw Fitness: Suma de las strengths de los individuos dominantes.
Densidad: Calculada usando la distancia de k-ésimo vecino más cercano,
para mantener diversidad.
```

3.2 Non-dominated Sorting Genetic Algorithm II (NSGA-II)

NSGA-II es otro MOEA ampliamente utilizado, conocido por su eficiencia y capacidad para mantener la diversidad en el frente de Pareto. Utiliza un algoritmo de clasificación no dominada rápida y un operador de distancia de aglomeración (crowding distance) para clasificar y seleccionar individuos.

Pseudocódigo de NSGA-II:

```
Algoritmo NSGA-II

Entrada:
    - Problema multiobjetivo
    - Tamaño de población N
    - Número de generaciones G
    - Parámetros de crossover y mutación

1. Inicializar una población P de N soluciones aleatorias
2. Evaluar los objetivos de cada individuo en P

Para generación = 1 hasta G hacer:
    3. Realizar non-dominated sorting sobre P (clasificar en frentes de
Pareto)
    4. Calcular crowding distance para cada individuo en cada frente
    5. Seleccionar padres de P usando selección por torneo basada en
rango de no-dominancia y crowding distance
    6. Generar una población Q de hijos aplicando operadores de crossover
y mutación a los padres
    7. Combinar P y Q en una población R = P ∪ Q
    8. Realizar non-dominated sorting sobre R para identificar nuevos
frentes de Pareto
    9. Construir la nueva población P_next de tamaño N:
        a. Añadir frentes completos desde el mejor rango hasta que la
población alcance N o el último frente exceda N.
        b. Si el último frente excede N, seleccionar los individuos con
mayor crowding distance de ese frente para completar P_next.
    10. P = P_next

Fin Para

11. Retornar el frente de Pareto de la última población P (o el archivo
externo, si se usa una estrategia de archivo)

Fin Algoritmo

Notas:
Non-dominated sorting: Clasifica las soluciones en frentes de Pareto
basados en dominancia.
Crowding distance: Mide la densidad de soluciones alrededor de un punto,
una mayor crowding distance favorece la diversidad de soluciones.
El proceso de selección, crossover y mutación es estándar de algoritmos
evolutivos.
```

5. Resultados Experimentales

Esta sección presenta los resultados obtenidos de la ejecución de los algoritmos de optimización multi-objetivo implementados (SPEA y NSGA-II) sobre las instancias seleccionadas del Problema del Viajante de Comercio (TSP). Los experimentos se realizaron con el objetivo de evaluar y comparar el desempeño de estas metaheurísticas en la aproximación de los frentes de Pareto multi-objetivo.

5.1. Configuración Experimental

Todas las ejecuciones se llevaron a cabo en el siguiente entorno de hardware:

- **Procesador:** Intel Core i5-7200u
- **Memoria RAM:** 8 GB
- **Sistema Operativo:** Windows 10
- **Entorno de Desarrollo:** Python 3.11 con las librerías NumPy 2.2.6 y matplotlib 3.10.3

5.2. Métricas de Comparación

El rendimiento de los algoritmos se evaluó utilizando un conjunto de métricas comunes en la optimización multi-objetivo, cada una proporcionando una perspectiva diferente sobre la calidad del frente de Pareto aproximado:

- **M1 (Spacing):** Cuantifica la uniformidad de la distribución de las soluciones en el frente de Pareto aproximado. Un valor bajo (idealmente 0) indica que las soluciones están bien espaciadas.
- **M2 (Diversity / Spread):** Mide la extensión o amplitud del frente de Pareto aproximado en el espacio objetivo. Un valor más alto es deseable, indicando que el algoritmo ha explorado una porción más amplia del frente de Pareto.
- **M3 (Coverage of Front):** Evalúa la fracción del `Ytrue` que es cubierto o dominado por el frente de Pareto aproximado. Valores más altos indican una mejor cobertura del frente de referencia.
- **Error:** Representa el porcentaje de soluciones en el frente de Pareto aproximado que son dominadas por soluciones en el `Ytrue`. Un valor bajo (idealmente 0) es deseable, indicando una alta precisión de las soluciones encontradas.

5.3. Resultados Obtenidos

A continuación, se presentan las tablas con los resultados promedio para cada métrica, para el problema TSP y las instancias utilizadas.

Tabla 1: Resultados Promedio para el Problema del Viajante de Comercio (TSP)

Problema	Instancia	Algoritmo	M1 (Spacing) (avg)	M2 (Generational Distance) (avg)	M3 (Coverage of Front) (avg)	Error (avg)	Tiempo Promedio (s)
TSP	tsp_KROAB100.TSP.TXT	NSGA	767.089	414.493	2912.8	23.2061	12.5415
TSP	tsp_KROAB100.TSP.TXT	SPEA	709.981	30035.2	167567	0	29.3949
TSP	tsp_kroac100.tsp.txt	NSGA	723.586	643.845	4792.68	0.27027	13.6964
TSP	tsp_kroac100.tsp.txt	SPEA	0	42033.3	173033	0	32.5306

Conclusión de los Resultados

Esta sección discute los hallazgos basados en la tabla de resultados promedio para el Problema del Viajante de Comercio (TSP) utilizando las instancias `tsp_KROAB100.TSP.TXT` y `tsp_kroac100.tsp.txt` con los algoritmos SPEA y NSGA.

Análisis de Métricas:

- **M1 (Spacing):** Esta métrica busca cuantificar la uniformidad de la distribución de las soluciones. Idealmente, un valor bajo indica una mejor uniformidad. Para `tsp_KROAB100.TSP.TXT`, SPEA (709.981) muestra un `Spacing_avg` ligeramente inferior a NSGA (767.089), lo que podría sugerir una distribución un poco más uniforme. Sin embargo, para `tsp_kroac100.tsp.txt`, SPEA presenta un `Spacing_avg` de 0, lo cual es inusual y probablemente indica un problema en la implementación o una situación extrema donde el algoritmo converge a muy pocas soluciones idénticas, lo que requeriría una revisión. NSGA para esta instancia tiene un `Spacing_avg` de 723.586.
- **M2 (Generational Distance):** Esta métrica mide la cercanía de las soluciones del frente Pareto aproximado al frente Pareto verdadero (o de referencia). Un valor más bajo es preferible, indicando una mejor convergencia. SPEA exhibe valores de `Generational_Distance_avg` significativamente más altos que NSGA para ambas instancias (`30035.2` y `42033.3` para SPEA vs. `414.493` y `643.845` para NSGA). Esto sugiere que NSGA, en promedio, produce frentes de Pareto aproximados más cercanos al frente de referencia en términos de Generational Distance.
- **M3 (Coverage of Front):** Esta métrica evalúa qué tan bien el frente aproximado cubre o domina el frente de referencia. Un valor más alto es deseable. Aquí, SPEA supera drásticamente a NSGA, con valores de `M3_Convergencia_Promedio_avg` de `167567` y `173033` en comparación con `2912.8` y `4792.68` para NSGA. Esto indica que SPEA logra una cobertura mucho mayor del frente de referencia.

- **Error (Error Ratio):** Esta métrica representa el porcentaje de soluciones en el frente de Pareto aproximado que son dominadas por soluciones en el frente de referencia (`Ytrue`). Un valor idealmente 0 es deseable. **SPEA destaca notablemente en esta métrica, logrando un `Error_Ratio_avg` de 0.0 para ambas instancias.** Esto significa que todas las soluciones generadas por SPEA son no-dominadas respecto al frente de referencia, indicando una excelente precisión en la convergencia. En contraste, NSGA muestra un `Error_Ratio_avg` considerablemente alto (`23.2061` para `KROAB100` y `0.27027` para `KROAC100`), aunque el valor para `KROAC100` es muy bajo, indicando una mejor precisión en esa instancia en comparación con `KROAB100`.
- **Tiempo Promedio (s):** Esta métrica evalúa la eficiencia computacional. Un tiempo menor es preferible. NSGA es consistentemente más rápido que SPEA en ambas instancias, con tiempos promedio de aproximadamente 12-13 segundos, mientras que SPEA tarda alrededor de 29-32 segundos. Esto sugiere que NSGA es más eficiente en términos de tiempo de ejecución.

Conclusiones Específicas:

- **Precisión y Convergencia:** SPEA demuestra una capacidad excepcional para encontrar soluciones verdaderamente no-dominadas, como lo indica su `Error_Ratio_avg` de 0.0. Esto lo convierte en el algoritmo preferido cuando la precisión y la garantía de no-dominancia son los objetivos principales.
- **Diversidad y Cercanía:** NSGA parece tener una mejor capacidad para generar frentes más cercanos al frente de referencia (menor `Generational_Distance_avg`), lo que podría indicar una mejor convergencia en términos de distancia. Sin embargo, su mayor `Error_Ratio_avg` en una de las instancias sugiere que, aunque las soluciones pueden estar cerca del frente de referencia, un porcentaje significativo de ellas pueden ser dominadas.
- **Cobertura:** SPEA sobresale en la `M3_Convergencia_Promedio_avg`, lo que implica que sus frentes cubren una porción más amplia del frente de referencia en el espacio objetivo.
- **Eficiencia Computacional:** NSGA es claramente más rápido que SPEA, lo que lo hace más adecuado para escenarios donde el tiempo de ejecución es una restricción crítica.

Consideraciones y Trabajos Futuros

La anomalía del `Spacing_avg` en 0.0 para SPEA en la instancia `tsp_kroac100.tsp.txt` debe ser investigada a fondo, ya que podría indicar un problema en la métrica o una falta de diversidad en las soluciones generadas. La interpretación de la `M2 (Generational Distance)` debe hacerse con cautela en conjunto con la `Error Ratio`, ya que un frente cercano no siempre significa un frente de Pareto verdaderamente óptimo si contiene muchas soluciones dominadas.

Para futuras investigaciones, se reafirma la necesidad de:

- **Validación y Refinamiento de Métricas:** Es crucial verificar la correcta implementación de todas las métricas, especialmente `M1` y `M2`, para asegurar que reflejen con precisión las características de los frentes de Pareto aproximados. La construcción del `Ytrue` también debe ser robusta.
- **Experimentación con Parámetros:** Un estudio de sensibilidad a los parámetros de cada algoritmo es esencial para optimizar su rendimiento y entender su impacto en las diferentes métricas.
- **Visualización de Frentes de Pareto:** La representación gráfica de los frentes de Pareto aproximados y del `Ytrue` sería de gran utilidad para una comprensión visual y cualitativa de la convergencia y diversidad de los algoritmos.
- **Análisis Multicriterio Complementario:** Considerar otras métricas o métodos de análisis multicriterio que puedan ofrecer una visión más completa del desempeño de los algoritmos.