Materia:

DISEÑO ELECTRÓNICO BASADO EN SISTEMAS EMBEBIDOS

Alumno:

Posadas Pérez Isaac Sayeg

Paniagua Rico Juan Julian

García Azzúa Jorge Roberto

Grado y grupo:

8°G

Profesor:

Garcia Ruiz Alejandro Humberto

Unidad 3 - Tarea 7:

Clasificación de Metaheurísticas

Clasificación de las metaheurísticas

Introducción

Las metaheurísticas constituyen un conjunto amplio de estrategias de optimización que buscan resolver problemas complejos con eficiencia aceptable, incluso cuando no es posible encontrar la solución óptima exacta. Dada su versatilidad, las metaheurísticas pueden organizarse en distintas categorías según sus mecanismos de búsqueda, inspiración natural o estructura de exploración.

Comprender la clasificación de las metaheurísticas permite seleccionar la técnica más adecuada según las características del problema y las restricciones del sistema embebido. Esta clasificación también ayuda a identificar oportunidades de mejora, combinación o adaptación de algoritmos a entornos con recursos limitados.

Desarrollo

1. Clasificación según la estructura de búsqueda

Esta clasificación se basa en la forma en que la metaheurística explora el espacio de soluciones:

a) Métodos basados en trayectorias (Trajectory-based)

Exploran una sola solución a lo largo del tiempo. La búsqueda se mueve de una solución a otra cercana mediante reglas específicas.

Ejemplos:

- **Recocido Simulado (Simulated Annealing - SA)** Se inspira en el proceso físico de recocido de metales, donde la temperatura se reduce gradualmente para alcanzar una estructura estable. En la optimización, SA permite explorar soluciones menos óptimas al inicio, pero reduce gradualmente la "temperatura" para converger hacia la mejor solución sin quedar atrapado en óptimos locales.
- **Búsqueda Tabú (Tabu Search - TS)** Usa una lista de prohibiciones o restricciones ("tabú") para evitar la exploración repetida de soluciones ya visitadas y fomentar la búsqueda en nuevas regiones del espacio de soluciones. Es útil para mejorar la exploración global y evitar ciclos innecesarios en la búsqueda.

-

Características:

- Útil para problemas donde se puede definir un vecindario claro.
- Fáciles de implementar en sistemas embebidos por su bajo requerimiento de memoria.

b) Métodos basados en población (Population-based)

Trabajan con múltiples soluciones simultáneamente. La evolución de la población guía la búsqueda hacia mejores soluciones.

Ejemplos:

- **Algoritmos Genéticos (GA)** Se inspiran en la evolución biológica mediante mecanismos como selección, cruce y mutación. Operan sobre una población de soluciones, evolucionando a través de generaciones hasta encontrar una solución óptima.
- **Programación Evolutiva (EP)** Similar a los GA, pero se enfoca más en la adaptación de individuos mediante mutaciones en lugar de cruces. Se utiliza en problemas de optimización continua y no necesariamente en estructuras codificadas como cromosomas.
- **Estrategias Evolutivas (ES)** Enfatizan la mutación y selección en poblaciones pequeñas. Se usan en optimización de parámetros y diseño de ingeniería, generalmente con modelos matemáticos de evolución sin necesidad de operadores genéticos como el cruce.
-

Características:

- Mejor exploración global del espacio de búsqueda.
- Mayor carga computacional, aunque se pueden paralelizar.

2. Clasificación según la inspiración natural

Las metaheurísticas muchas veces se inspiran en fenómenos observados en la naturaleza:

a) Inspiradas en la biología evolutiva

Simulan mecanismos de evolución natural, como selección, reproducción y mutación.

- **Algoritmos Genéticos (GA)**
- **Programación Evolutiva**
- **Estrategias Evolutivas**

Aplicación en sistemas embebidos:

- Optimización de arquitecturas hardware/software adaptativas.

b) Inspiradas en el comportamiento colectivo (inteligencia de enjambre)

Se basan en el comportamiento emergente de agentes sociales simples (hormigas, aves, peces).

ACO (Ant Colony Optimization - Optimización de Colonias de Hormigas) - Se basa en el comportamiento de las hormigas al buscar rutas óptimas hacia fuentes de alimento. Utiliza feromonas virtuales para guiar la búsqueda de soluciones en problemas de optimización combinatoria, como el viaje del vendedor o el enrutamiento de redes.

PSO (Particle Swarm Optimization - Optimización por Enjambre de Partículas) Inspirado en el movimiento de bandadas de aves y cardúmenes de peces, este algoritmo optimiza soluciones mediante la colaboración de partículas que ajustan su posición en función de su experiencia y la de sus vecinos. Se usa en problemas de optimización continua y aprendizaje automático.

BFO (Bacterial Foraging Optimization - Optimización por Forrajeo Bacteriano) Modela el comportamiento de las bacterias en la búsqueda de nutrientes. Simula procesos como quimiotaxis, reproducción y eliminación, permitiendo encontrar soluciones óptimas en problemas de optimización global.

Aplicación en sistemas embebidos:

- Enrutamiento dinámico y tolerancia a fallos en redes de sensores.

c) Inspiradas en fenómenos físicos o químicos

Se inspiran en leyes de la física o procesos de la materia.

- **Recocido Simulado (enfriamiento de metales)**
- **Electromagnetismo**
- **Algoritmos basados en fluidos o gravedad**

Aplicación en sistemas embebidos:

- Optimización de recursos y planificación térmica.

3. Clasificación según la estrategia de búsqueda

Aquí se agrupan las metaheurísticas en función de cómo equilibran exploración y explotación:

a) Explotación intensiva (búsqueda local refinada)

Se enfocan en mejorar soluciones existentes sin explorar mucho.

- SA, TS

Ventaja:

- Buen desempeño en problemas donde la calidad local importa más.

Desventaja:

- Riesgo de quedar atrapado en óptimos locales.

b) Exploración extensiva (búsqueda global)

Recorren muchas partes del espacio de búsqueda.

- PSO, GA, ACO

Ventaja:

- Mejor para problemas con múltiples óptimos o alta dimensionalidad.

Desventaja:

- Requiere mayor poder de cómputo, que puede ser un desafío en sistemas embebidos.

c) Algoritmos híbridos

Combinan técnicas de exploración y explotación. Ejemplo: GA + búsqueda local.

Aplicación:

- Sistemas adaptativos embebidos donde se requiere precisión y eficiencia.

4. Clasificación según adaptabilidad y control

En esta categoría se considera la capacidad de los algoritmos para adaptarse durante la ejecución:

a) Metaheurísticas estáticas

Parámetros fijos durante toda la ejecución (ej. temperatura en SA, probabilidad de mutación en GA).

- Más fáciles de implementar.
- Menor capacidad de adaptación al problema.

b) Metaheurísticas dinámicas o auto-adaptativas

Los parámetros cambian durante la ejecución en respuesta al progreso.

- GA con tasa de mutación variable.
- PSO con coeficientes adaptativos.

Ventajas:

- Mejor adaptación a diferentes fases del problema.
- Pueden mejorar eficiencia y convergencia en tiempo real.

Ejemplos

- **GA con búsqueda local en FPGA:** Se clasifica como híbrido y basado en población, útil para asignación eficiente de recursos lógicos.
- **ACO para redes WSN:** Inspirado en comportamiento social, basado en población, útil en sistemas embebidos para enrutamiento tolerante a fallos.
- **SA para control PID embebido:** Basado en trayectorias, físico, buena solución con poco cómputo.
- **PSO auto-adaptativo en drones autónomos:** Alta exploración, comportamiento colectivo, adaptabilidad dinámica.

Conclusión

Clasificar las metaheurísticas permite comprender mejor sus fortalezas, limitaciones y áreas de aplicación. Esta organización es clave al implementar soluciones en sistemas embebidos, donde los recursos son limitados y se necesita una selección cuidadosa del algoritmo.

La elección adecuada depende del tipo de problema, las restricciones del sistema y los objetivos deseados. Conociendo sus categorías, es posible integrar o adaptar técnicas para lograr un mejor desempeño sin comprometer la viabilidad del sistema embebido.

Bibliografía

❖ *Talbi, E. G. (2009). Metaheuristics: From Design to Implementation. Wiley.*
❖ *Blum, C., & Roli, A. (2003). Metaheuristics in combinatorial optimization: Overview and conceptual comparison. ACM Computing Surveys.*
❖ *Marwedel, P. (2011). Embedded System Design: Embedded Systems Foundations of Cyber-Physical Systems. Springer.*
❖ *Dorigo, M., & Di Caro, G. (1999). Ant Colony Optimization: A New Meta-Heuristic.*
❖ *Kennedy, J., & Eberhart, R. (1995). Particle Swarm Optimization.*
❖ *Kirkpatrick, S. et al. (1983). Optimization by Simulated Annealing.*
❖ *Givargis, T., & Vahid, F. (2002). Embedded System Design: A Unified Hardware/Software Introduction. Wiley.*