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 4:

Problema de optimización en sistemas

Problema de optimización en sistemas embebidos

Introducción

Los problemas de optimización son fundamentales en el diseño y desarrollo de sistemas embebidos, ya que estos sistemas operan bajo estrictas limitaciones de recursos como energía, tiempo de procesamiento, memoria y tamaño físico. Un problema de optimización busca encontrar la mejor solución posible, dentro de un conjunto de soluciones factibles, que maximice o minimice una o más funciones objetivo, respetando ciertas restricciones.

En sistemas embebidos, los problemas de optimización pueden abarcar desde la asignación eficiente de tareas en tiempo real hasta la selección de componentes hardware que cumplan con requisitos específicos de rendimiento y coste. Resolver estos problemas de forma efectiva permite mejorar la eficiencia global del sistema, su estabilidad, su consumo energético y su tiempo de respuesta, lo que es especialmente crítico en aplicaciones como dispositivos médicos, automóviles autónomos, redes de sensores y sistemas industriales.

Desarrollo

1. Definición formal del problema de optimización

Un problema de optimización en su forma general está definido por:

- Una o varias funciones objetivo: f(x)f(x)f(x), que se desea minimizar o maximizar.

- Un conjunto de variables de decisión: x=(x1,x2,...,xn)x = (x_1, x_2, ..., x_n)x=(x1,x2,...,xn).

- Un conjunto de restricciones: gi(x)≤0g_i(x) \leq 0gi(x)≤0, hj(x)=0h_j(x) = 0hj(x)=0, etc.

En sistemas embebidos, estas variables pueden representar frecuencia del procesador, voltajes, asignación de tareas, uso de memoria, o parámetros de control. Las restricciones pueden ser de tiempo real, consumo energético, espacio disponible o capacidad de cómputo.

2. Tipos de problemas de optimización en sistemas embebidos

- **Monoobjetivo:** Se busca optimizar una sola métrica (por ejemplo, minimizar el consumo de energía).

- **Multiobjetivo:** Involucra múltiples criterios que pueden estar en conflicto (por ejemplo, minimizar consumo energético y maximizar rendimiento).

- **Con restricciones:** Se deben cumplir límites físicos o lógicos del sistema.

- **Discretos o continuos:** Dependiendo de si las variables de decisión toman valores continuos (como una frecuencia de reloj) o discretos (como seleccionar un algoritmo entre varios).

3. Aplicaciones comunes

- **Optimización energética:** Ajuste de voltajes, escalado dinámico de frecuencia (DVFS), y modos de suspensión.

- **Asignación de tareas en sistemas multiprocesador:** Cómo distribuir procesos para evitar cuellos de botella y cumplir con tiempos de respuesta.

- **Diseño de hardware/software codesign:** Decidir qué funciones implementar en hardware y cuáles en software.

- **Compilación y optimización del código:** Selección de instrucciones, organización de memoria, e inlining de funciones para reducir ciclos de ejecución.

4. Métodos y algoritmos utilizados

- **Algoritmos exactos:** Como la programación lineal, programación entera y métodos de ramificación y poda. Son precisos pero costosos computacionalmente.

- **Algoritmos heurísticos:** Como algoritmos genéticos, búsqueda tabú, recocido simulado. Útiles para problemas grandes y complejos.

- **Algoritmos metaheurísticos:** Combinaciones sofisticadas que permiten un buen compromiso entre calidad de solución y tiempo de cómputo.

- **Algoritmos en tiempo real:** Para sistemas que deben tomar decisiones óptimas rápidamente, como controladores adaptativos o redes de sensores inalámbricos.

5. Consideraciones prácticas

- **Complejidad computacional:** Los sistemas embebidos no siempre pueden ejecutar algoritmos pesados debido a sus limitaciones.

- **Trade-offs:** A menudo se debe balancear entre precisión de la solución y velocidad de cálculo.

- **Herramientas disponibles:** Existen entornos como MATLAB, herramientas EDA, y bibliotecas en C/C++ o Python que permiten modelar y resolver estos

problemas.

Ejemplos

- **Optimización del tiempo de respuesta en un sistema de frenos ABS:** Distribuir tareas de control y monitoreo en procesadores dedicados para asegurar respuestas en milisegundos.

- **Minimización del consumo en un dispositivo IoT:** Configuración óptima de períodos de muestreo, modos de sueño y transmisión de datos para extender la duración de la batería.

- **Asignación de memoria en un sistema de adquisición de datos:** Distribuir eficientemente buffers y variables para evitar desbordamientos sin desperdiciar RAM.

- **Ajuste de parámetros de un algoritmo de visión artificial:** Afinar la resolución de captura, el tamaño de ventana y los umbrales de detección para lograr precisión sin sobrecargar el microcontrolador.

Conclusión

El problema de optimización es un elemento central en el diseño de sistemas embebidos, ya que permite encontrar configuraciones que maximizan el rendimiento del sistema respetando sus restricciones inherentes. Si bien su resolución puede ser compleja y demandante, el uso de técnicas adecuadas puede traducirse en soluciones eficientes, confiables y competitivas.

Los diseñadores deben considerar cuidadosamente el tipo de problema de optimización, las restricciones de su sistema, y las herramientas disponibles para abordar este desafío. Dominar esta área permite no solo mejorar el diseño técnico

de los sistemas embebidos, sino también tomar decisiones estratégicas que impactan en la viabilidad comercial de los productos desarrollados.

Bibliografía

❖ Marwedel, P. (2011). *Embedded System Design: Embedded Systems Foundations of Cyber-Physical Systems*. Springer.

❖ Deb, K. (2001). *Multi-Objective Optimization using Evolutionary Algorithms*. Wiley.

❖ Kirkpatrick, S., Gelatt, C. D., & Vecchi, M. P. (1983). *Optimization by Simulated Annealing*. Science.

❖ Givargis, T., & Vahid, F. (2002). *Platune: A Tuning Framework for System-on-a-Chip Platforms*. IEEE Transactions on CAD.

❖ Zitzler, E., Laumanns, M., & Thiele, L. (2001). *SPEA2: Improving the Strength Pareto Evolutionary Algorithm*.

❖ Buttazzo, G. C. (2011). *Hard Real-Time Computing Systems*. Springer.

❖ Kuo, T.-W., & Li, C.-L. (2003). *Optimizing energy consumption for real-time systems with limited resources*. Journal of Systems Architecture.

❖ Wolf, W. (2012). *Computers as Components: Principles of Embedded Computing System Design*. Morgan Kaufmann.