

2007

Problema NP-Completo: "Minimum Bin Packing"

Aplicación a la Gestión de Hardware 2D
Dinámicamente Reconfigurable

RONALD IVÁN VACA POQUIOMA

www.seccperu.org/ivanvaca

RIVP



“Minimum Bin Packing y su Aplicación a la Gestión de Hardware 2D Dinámicamente Reconfigurable”

Ronald Iván Vaca Poquioma^{1,2}

¹Sociedad de Estudiantes de Ciencia de la Computación

²Universidad Nacional de Trujillo

ivan@seccperu.org

Resumen

En el presente artículo presentaremos el problema Bin Packing como un problema NP-Completo y verificaremos si pertenece a éste tipo de problemas, además describiremos una de sus posibles aplicaciones en la cual mostraremos una heurística que permitirá mostrar una solución satisfactoria a éste problema.

1. Introducción

Cuando hablamos de algoritmos para la resolución de un problema se nos viene a la cabeza la idea de “*coste del problema*”, es decir cuánto costará el problema en términos de memoria y tiempo procesamiento específicamente. Ello conlleva a pensar que existirán problemas tratables e intratables, tratables llamaremos a los algoritmos que permiten resolver un problema computacional en un tiempo polinomial (P), algo así como resolver un problema con una solución determinada; y por último los problemas intratables, a quienes llamaremos a los algoritmos que permiten resolver un problema computacional en un tiempo no polinomial (NP), para estos tipos de problemas en los que no existe una solución determinada, podremos encontrar algoritmos que nos podrán dar una solución satisfactoria, a estos algoritmos llamaremos Algoritmos Heurísticos.

Ahora nos encontramos con un problema NP-Completo como lo es el problema Minimum Bin Packing, problema que consiste específicamente en organizar óptimamente un almacén de un tamaño B con diferentes elementos, tal que la suma de espacios estos elementos es menor o igual al área o capacidad de B. Éste problema se dice que es NP-Completo porque el problema es reducible.

2. Trabajos Previos

Se ha realizado aplicaciones utilizando listas de vértices como patrón para el

recorrido de las áreas de cada elemento (tareas), de tal manera que permite tener el control de la capacidad que ocupa cada área para que ésta pueda ser calculada y posteriormente reubicada.

3. Problemática

3.1 Problema

Instancia: Conjunto Finito de U términos, un tamaño $s(u) \in Z^+$ para cada $u \in U$ y un número entero de capacidad B .

Solución: Una partición de U en conjunto disjuntos U_1, U_2, \dots, U_m tal que la suma de las particiones de U de los ítems de cada U_i es B o menor.

El algoritmo que soluciona el problema se llama FIRST FIT; tiene una complejidad de $O(n^2)$, y produce soluciones medianamente buenas. FIRST FIT significa que se coloca por vez cada objeto en el en el contenedor. El algoritmo primero ordena los objetos, de forma tal que los considera en orden decreciente en cuanto al tamaño.

El ordenamiento puede realizarse en $O(n) \log n$, y j se incrementa, a lo sumo $n(n-1)/2$ veces. El resto de las instrucciones se ejecuta a lo sumo n veces, de forma tal que la complejidad en el peor de los casos es $O(n^2)$.

Otra estrategia heurística que puede emplearse para Bin Packing es BEST FIT: un objeto de tamaño s se coloca en una parte del áreas del contenedor B_j que es el más completo entre aquellos en los que cabe el objeto; es decir, b_j es el elemento maximal para la condición $b_j + s \leq B$. Si los s_i están ordenados en forma decreciente, la estrategia BEST FIT funciona de forma similar a FIRST FIT. Si los s_i no están ordenados (para BEST FIT o FIRST FIT) los resultados pueden ser peores, pero la cantidad de área usada podría aún ser menor que el doble de la optima.

Algoritmo 1 FIRST FIT (S)

Entradas: $U = (U_1, \dots, U_n)$, donde $0 < U_i \leq 1$, para $1 \leq i \leq n$

Salida: $B = (B_1, \dots, B_n)$, donde para cada $1 \leq j \leq n$, B_j es un subconjunto de $\{1, 2, \dots, n\}$; los elementos de B_j son los índices de los objetos ubicados en la j -ésimo parte del contenedor.

Comentario: Para cada $1 \leq j \leq n$, b_j es la cantidad de espacio en el j -ésimo parte del contenedor que ya ha sido cubierto.

Inicio

Para $j \in 1$ hasta n

$B_j \in \emptyset$

$b_j \in 0$

Fin-Para

Para $t \in 1$ hasta n

$j \in 1$

 Mientras $b_j + u_{it} > 1$ hacer:

$j \in j + 1$

 Fin-Mientras

Fin-Para

$B_j \in B_j \cup \{t\}$;

$b_j \in b_j + U_{it}$

Fin

Como hemos visto el problema algoritmo se resuelve en tiempo polinomial

3.2 Verificación

Sea el Algoritmo Verificador A compuesto por una iteración que comprueba que

$$\sum_{i=0}^n b_i = \|B\|$$

tal que $b_i \in B$. Entonces se demuestra que el algoritmo verificador se ejecuta en tiempo polinomial.

Sea el conjunto de entradas $W = \{w_1, w_2, w_3, \dots, w_n\}$ en donde W representa al conjunto $U = \{U_1, U_2, \dots, U_n\}$ tal que $n > 1$ y un certificado y tal que está representado por la lista de áreas de los elementos que serán ubicados en el contenedor.

Entonces vemos que:

- El algoritmo verificador A , la que verifica la factibilidad de heurística del problema NP-Completo *Minimum Bin Packing*, se ejecuta en tiempo polinomial.
- Además el lenguaje L el cual tiene el alfabeto de las entradas $w_i \in W$ (conjunto de todas las entradas) y con un certificado y , se representa $L = \{w \in \{\text{entradas}\}^* / \text{un certificado } y \text{ y con } |y| = O(|w|^c) \text{ tal que } A(w, y) = 1\}$.

Por lo tanto $L \in NP$.

4. Aplicación

El problema de los últimos años del hardware reconfigurable con una funcionalidad de un hardware multitarea, éste hardware cuenta con recursos como tarjetas con dispositivos como los Field Programmable Gate Arrays (FPGAs) a las que se puede configurar dinámicamente, para desarrollar nuevas funcionalidades. Su evolución en cuanto a su incremento de tamaño, densidad, sino también de muchas características novedosas permiten reconfigurar estos dispositivos de tal manera que permite considerar la configuración de múltiples tareas. De modo que éste recurso se puede ver como un área de proceso en dos dimensiones, capaz de ejecutar un conjunto de tareas de modo concurrente.

Entonces podemos ver que el problema de los FPGAs, son la reubicación de las tareas que se ejecutarán sin afectar a las tareas existentes en los FPGAs que se encuentran en ejecución. Esto tendrá gran influencia en la fragmentación y rendimiento de los FPGAs. A continuación vemos como es el funcionamiento del Gestor del hardware que está formado por tres componentes principales: planificador, asignador y el gestor de tareas.

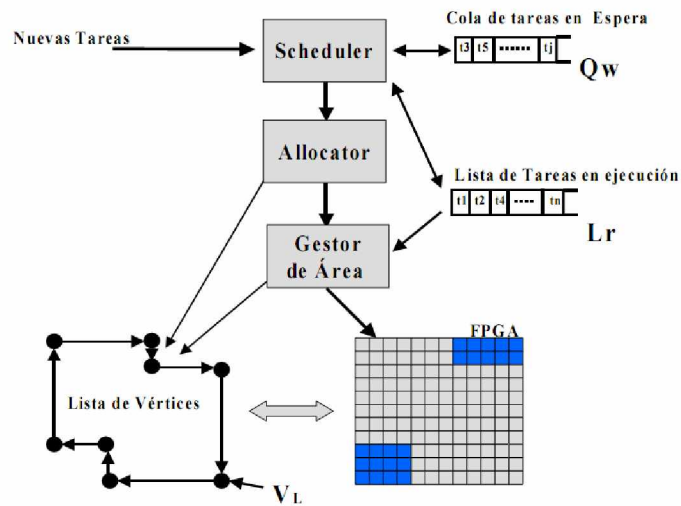


Fig 1. La figura que representa el funcionamiento del Gestor del hardware

5. Conclusiones

Hemos comprobado que existe un algoritmo heurístico que da una solución al problema NP-Completo *Minimum Bin Packing* que se resuelve en tiempo polinomial.

El Algoritmo Verificador de la solución al problema *Minimum Bin Packing* se ejecuta en tiempo polinomial, además el lenguaje L verificado por el algoritmo verificador A pertenece a NP ($L \in NP$).

Por lo tanto se demuestra que el problema *Minimum Bin Packing* es un problema NP-Completo.

6. Referencias

- [1] LEA P, ASAF LEVIN. Artículo: On Bin Packing with conflicts, University of Haifa, Israel.
- [2] TABERO J, SEPTIEM J, MECHA H. Artículo: *Gestión de Hardware 2D Multitarea en FGPA Dinámicamente Reconfigurables basado en listas de Vértices*, Univeridad Complutense de Madrid, España, 2003.
- [3] SEPTIEM J, MECHA H, Artículo: *Hardware Dinámicamente Recongifurable*. Univeridad Complutense de Madrid, España, 2003.