Un algoritmo memético para Multidimensional Two-Way Number Partitioning Problem

Luis Balderas Ruiz[1]*

Resumen
En este documento nos enfrentamos al problema de particionar un conjunto de vectores en dos subconjuntos de forma que la suma por coordenadas debe ser igual o lo más cercanas posible. Este problema, introducido por Kojic([1]), recibe el nombre de Multidimensional Two-Way Number Partitioning Problem (MDTWNPP). Proponemos un algoritmo memético (genético + búsqueda local) y comparo los resultados con los resultados existentes obtenidos por CPLEX basados en la formulación basada en programación lineal entera.

Keywords
Multidimensional Two-Way Number Partitioning Problem, cruce, mutación, genético, memético, búsqueda local

[1] *Departamento de Ciencias de la Computación e Inteligencia Artificial. CITIC-UGR. Universidad de Granada*
***Email**: luisbalderas@correo.ugr.es

Índice

Introducción

El problema del particionado de números es un clásico y bastante difícil problema de optimización combinatoria. Dado un conjunto S de n enteros, el problema de particionamiento de números en dos conjuntos (TWNPP) trata de dividir S en dos conjuntos cuya suma de cada subconjunto debe ser igual o muy parecida entre ellos.

A pesar de ser un problema NP-completo ([2]), se han propuesto múltiples aproximaciones basadas en algoritmos heurísticos para resolverlo de una forma óptima o aproximada: la heurística basada en la diferencia de conjuntos por Karmarkar y Karp ([3]), el algoritmo de Johnson et al. ([4]), un algoritmo genético por Ruml et al. ([5]), GRASP por Arguello et al. ([6]), la Búsqueda Tabu por Glover y Laguna ([7]), un algoritmo memético por Berreta et al. ([8]), etc.

El problema ha adquirido una gran atención dados sus aspectos teóricos y sus importantes aplicaciones en el mundo real. Dichas aplicaciones pueden verse en ([9]).

Multidimensional two-way number partitioning problem (MDTWNPP) fue introducido por Kojic ([1]) como la generalización de TWNPP donde, en vez de tener números, tenemos conjuntos de vectores que necesitamos particionar en dos conjuntos de forma que la suma por coordenadas sea tan próximo como sea posible.

MDTWNPP es NP-difícil. No hay una gran investigación en modelos matemáticos y soluciones que resuelvan este problema. Kojic ([1]) describió una formulación basada en programación lineal y la probó en conjuntos aleatoriamente generados con CPLEX, que parece ser la única manera de resolver el problema. Los resultados experimentales obtenidos muestran que MDTWNPP es muy difícil de resolver incluso en los escenarios más triviales.

El propósito de este trabajo es describir un algoritmo memético con el objetivo de resolver MDTWNPP. Los resultados de los experimentos se presentan y se comparan con los obtenidos por Kojic ([1]).

1. Definición del problema

Dado un conjunto de n vectores de dimensión m

$$S = \{v_i | v_i = (v_{i1}, v_{i2}, \dots, v_{im}), i \in \{1, \dots, n\}, m \in \mathbb{N}\}$$

de forma que, según la definición de Kojic ([1]), el problam consiste en dividir los elementos de S en dos conjuntos, S_1, S_2, de forma que:

1. $S_1 \cup S_2 = S$ y $S_1 \cap S_2 = \emptyset$
2. la suma de los elementos de los subconjuntos S_1, S_2 son iguales o casi iguales por coordenadas.

Si introducimos la variable t que denota la mayor diferencia en la suma por coordenadas, es decir,

$$t = \text{máx}\left\{ \mid \sum_{i \in S_1} v_{ij} - \sum_{i \in S_2} v_{ij} \mid j \in \{1, \ldots, m\} \right\}$$

entonces la función objetivo de MDTWNPP es minimizar t. Si mín $t = 0$ entonces la partición se llama perfecta.

MDTWNPP se puede generalizar fácilmente en el caso de que el conjunto de vectores se particiona en un número definido de subconjuntos en vez de en dos subconjuntos.

2. Algoritmo memético para MDTWNPP

En esta sección describimos el algoritmo memético (genético + búsqueda local) para buscar una solución a Multidimensional Two-Way Number Partitioning Problem.

Utilizamos una representación binaria donde cada cromosoma es un string de bits 0 o 1 de tamaño fijo (vector n-dimensional). Esta representación asegura que el conjunto de vectores pertenecientes a S se divide en dos subconjuntos S_1 y S_2.

En lo que se refiere a la población inicial, realizamos una generación aleatoria de individuos. Definimos el valor de aptitud o fitness en MDTWNPP, para una partición de vectores en dos subconjuntos, como la mayor diferencia en la suma por coordenadas, de forma que nuestra voluntad, como ya se ha comentado, es minimizarla.

Utilizamos dos operadores genéticos para combinar las soluciones existentes en otros (operador de cruce) y para generar diversidad (operador de mutación). En mi caso, selecciono dos progenitores utilizando dos veces el método de torneo binario, de forma que dos padres se cogen aleatoriamente de la población, se comparan su fitness y el mejor se escoge para la generación de descendencia. Como digo, se realizan dos torneos para elegir a los dos progenitores. El operador de cruce se basa en elegir aleatoriamente el valor del gen del hijo de entre los valores de gen de cada padre. El operador de mutación nos permite alterar uno o más genes de de un cromosoma. Considero que la mutación cambia el bit de 0 a 1 o de 1 a 0 en el hijo generado. Para fijar el número de cambios, establezco la probabilidad de mutación en un 10 %.

Mejoramos el algoritmo genético añadiendo un operador heurístico de búsqueda local: sea t la mayor diferencia en las sumas por coordenada, obtenido en la coordenada $j, j \in \{1, \ldots, m\}$. Entonces, para el subconjunto con la mayor suma, analizamos los elementos correspondientes a la coordenada j, seleccionamos aquel que se acerque más al valor $\frac{t}{2}$ y se reasigna al otro subconjunto.

Los parámetros son muy importantes para un buen desempeño del algoritmo genético. Basándome en los resultados encontrados en la literatura ([10]), escojo una población de 5000 vectores y 10000 generaciones. En cada generación, se generan 50 descendientes, que se introducen en la población si su fitness es menor que el peor cromosoma de la población, en cuyo caso éste se elimina.

3. Resultados de la experimentación

En esta sección mostramos los resultados obtenidos. Enfrentamos los datos de CPLEX ([1]) con los encontrados con el genético y el memético. CPLEX es muy ineficiente en tiempo, por lo que estos datos son los mejores encontrados en 30 mins de ejecución. Aún así, se verá que son mucho mejores que los obtenidos con nuestro algoritmo. Esto nos brinda la oportunidad de tomar medidas para mejorar. Estas serán presentadas en la próxima sección.

Instancia problema	CPLEX		
	Mejor	Tiempo (S)	Media
500_20a	48281.977	1000.12	-
500_20b	54921.900	237.63	-
500_20c	41578.884	1382.98	-
500_20d	54293.2	1728.58	-
500_20e	41092.622	1713.03	-
	GA		
	Mejor	Media	Tiempo (S)
500_20a	1000751.69	136080.16	217.92
500_20b	76399.81	140769.17	210.06
500_20c	93303.56	141157.7	209.42
500_20d	106775.44	148521.97	219.08
500_20e	106385.06	148307.56	215.98

Cuadro 1. Resultados (CPLEX-GA)

Instancia problema	GA		
	Mejor	Media	Tiempo (S)
500_20a	1000751.69	136080.16	217.92
500_20b	76399.81	140769.17	210.06
500_20c	93303.56	141157.7	209.42
500_20d	106775.44	148521.97	219.08
500_20e	106385.06	148307.56	215.98
	MA		
	Mejor	Media	Tiempo (S)
500_20a	1000751.69	104995.07	260.28
500_20b	76399.81	85680.69	260.16
500_20c	93303.56	87573.84	257.24
500_20d	106775.44	109815.38	253.55
500_20e	106385.06	108410.76	258.16

Cuadro 2. Resultados (GA-MA)

4. Conclusiones y trabajos futuros

En este trabajo presentábamos una solución basada en un algoritmo memético para resolver Multidimensional Two-Way Number Partitioning Problem. Para comparar la mejora de la búsqueda local sobre el algoritmo genético, hemos presentado también la comparativa entre GA y MA. La primera conclusión que extraemos es que el memético desarrollado es muchísimo más eficiente en tiempo que CPLEX. Sin embargo, los resultados son bastante peores. Si comparamos GA con MA, vemos que coincide el mejor resultado pero la media es mejor para MA, por lo que la búsqueda local favorece el resultado en media. Como contrapartida, observamos que hay un incremento en el tiempo, como era de esperar.

Como trabajo futuro, en primer lugar señalamos una nueva forma, más inteligente, de elegir la población inicial. Debería elegirse, para un mejor rendimiento de los algoritmos, ir añadiendo cromosomas a la población de forma que se vaya reduciendo la máxima diferencia en las sumas por coordenadas, en lugar de aleatoriamente. Algunos estudios apuntan ([10]) que la mejora puede llegar al 50 %. Además, sería interesante explorar con mayor profundidad los valores de los parámetros del algoritmo genético. Para reducir el coste computacional, en cada generación sólo se producen 50 nuevos descendientes. Un mayor número de esos descendientes (hasta 10 veces más que el total de la población [10]) generarían más oportunidades para encontrar mejores soluciones. Vemos que en media el algoritmo mejora continuamente, por lo que sería interesante explotar esta vía hasta el final. Por último, sería posible explotar otros operadores de cruce distintos para así comparar resultados, por ejemplo, eligiendo lo que se llama un punto de cruce y combinar, de forma más literal, el material genético de los progenitores.

Referencias

[1] Jelena Kojić. Integer linear programming model for multidimensional two-way number partitioning problem. *Computers and Mathematics with Applications*, 60(8):2302 – 2308, 2010.

[2] Michael R. Garey and David S. Johnson. *Computers and Intractability; A Guide to the Theory of NP-Completeness*. W. H. Freeman and Co., USA, 1990.

[3] Narenda Karmarker and Richard M. Karp. The differencing method of set partitioning. Technical Report UCB/CSD-83-113, EECS Department, University of California, Berkeley, 1983.

[4] David Johnson, Cecilia Aragon, Lyle McGeoch, and Catherine Schevon. Optimization by simulated annealing: An experimental evaluation; part ii, graph coloring and number partitioning. *Operations Research*, 39:378–406, 06 1991.

[5] Wheeler Ruml, Jenny Ngo, J. Marks, and S. Shieber. Easily searched encodings for number partitioning. *Journal of Optimization Theory and Applications*, 89:251–291, 05 1996.

[6] Michael F. Argüello, Thomas A. Feo, and Olivier Goldschmidt. Randomized methods for the number partitioning problem. *Computers and Operations Research*, 23(2):103 – 111, 1996.

[7] Fred Glover and Manuel Laguna. *Tabu Search*. Kluwer Academic Publishers, USA, 1997.

[8] Regina Berretta. Enhancing the performance of memetic algorithms by using a matching based recombination algorithm results on the number partitioning problem. 01 2003.

[9] E. G. Coffman, D. S. Johnson, G. S. Lueker, and P. W. Shor. Probabilistic analysis of packing and related partitioning problems. *Statistical Science*, 8(1):40–47, 1993.

[10] Petrica Pop and Oliviu Matei. A genetic algorithm approach for the multidimensional two-way number partitioning problem. volume 7997, pages 81–86, 01 2013.