Experimentos computacionales en la resolución del Problema de Códigos de Identificación

D. Severin[1,2] and V. Vansteenkiste[1,2]
[1] Facultad de Cs. Exactas, Ingeniería y Agrimensura, UNR, Argentina
{daniel,vvanst}@fceia.unr.edu.ar
[2] CONICET, Argentina

Abstract. El Problema de Códigos de Identificación (PCI) es un problema NP-difícil relativamente nuevo que, además de contar con aplicaciones concretas (véase el trabajo de Karpovsky, Chakrabarty y Levitin, *On a new class of codes for identifying vertices in graphs*. IEEE Trans. Inf. Theory 44, 599–611), es desafiante tanto desde el punto de vista teórico como computacional. En particular, se han propuesto algoritmos polinomiales para resolver el PCI sobre clases particulares de grafos y, más recientemente, se ha estudiado el poliedro asociado a su formulación natural donde, en algunos casos, se ha dado la descripción completa para algunas familias de grafos (véase el trabajo de Argiroffo, Bianchi y Wagler, *Study of Identifying Code Polyhedra for Some Families of Split Graphs*, LNCS 8596, 13–25).

En esta comunicación reportamos algunos experimentos computacionales respecto a la performance de un modelo de programación entera para el PCI. Sea un grafo $G = (V, E)$, y $N[j] = N(j) \cup \{j\}$ el vecindario cerrado de $j \in V$. Un subconjunto $C \subset V$ es un *código de identificación* si es *dominante*, es decir $N[j] \cap C \neq \varnothing$ para todo $j \in V$, y a la vez *separador*, es decir $(N[j] \triangle N[k]) \cap C \neq \varnothing$ para todo $j, k \in V$ tales que $j \neq k$. El PCI consiste en hallar el código de identificación de mínimo cardinal, que denotamos con $\gamma_{ID}(G)$. De esta definición surge naturalmente la siguiente formulación de Programación Lineal Entera:

$\gamma_{ID}(G) = \{\min \sum_{v \in V} x_v : \sum_{v \in N[j]} x_v \geq 1, \quad \forall\, j \in V \ (1);$
$\sum_{v \in N[j] \triangle N[k]} x_v \geq 1, \quad \forall\, j, k \in V,\ j \neq k \ (2); \quad x_v \in \{0, 1\}, \quad \forall\, v \in V\}.$

Si eliminamos las restricciones (1), obtenemos una relajación llamada Problema Separador (PS) y cuyo parámetro denotamos $\gamma_{SEP}(G)$. Si bien, obtener $\gamma_{SEP}(G)$ también es NP-difícil, es sabido que $d = \gamma_{ID}(G) - \gamma_{SEP}(G) \leq 1$, pero hasta el momento no se conoce como se comporta esta diferencia. En este sentido, utilizando CPLEX 12.6, resolvimos ambos problemas sobre 64 instancias aleatorias de distintos tamaños y notamos que, para grafos de baja densidad (20%), $d = 0$ en el 27% de los casos, mientras que para grafos de densidad media (50%) y alta (80%), $d = 0$ en la totalidad de las instancias evaluadas. También observamos que, con la actual formulación, se puede resolver el PCI en un tiempo razonable para grafos de hasta 70 vértices, siendo los de densidad media los más difíciles.

En un segundo experimento nos propusimos evaluar las siguientes estrategias. La Estrategia 1 considera la actual formulación del PCI. En la Estrategia 2, las restricciones (1) no son incorporadas en la relajación inicial sino que son gestionadas como *lazy constraints* (se van agregando a medida que se encuentran soluciones enteras que las violan). La Estrategia 3 es similar a la 2, sólo que las restricciones (1) son gestionadas tanto como *lazy constraint* como *cortes* (también se pueden agregar cuando existen soluciones fraccionarias que las violan). De estas estrategias, la más conveniente para resolver instancias de baja densidad resultó ser la primera. Sin embargo, para instancias de media y alta densidad, la Estrategia 3 logra resolver mayor cantidad de instancias que la 1 y utiliza menos tiempo. No observamos que la Estrategia 2 sea más efectiva que la 3.