 Acta Univ. Sapientiae, Informatica **11,** 2 (2019) 131–141

DOI: 10.2478/ausi-2019-0009

Metric space method for constructing splitting partitions of graphs

Sándor SZABÓ
University of Pécs, Pécs, Hungary
email: sszabo7@hotmail.com

Abstract. In an earlier work [6] the concept of splitting partition of a graph was introduced in connection with the maximum clique problem. A splitting partition of a graph can be used to replace the graph by two smaller graphs in the course of a clique search algorithm. In other words splitting partitions can serve as a branching rule in an algorithm to compute the clique number of a given graph. In the paper we revisit this branching idea. We will describe a technique to construct not necessary optimal splitting partitions. The given graph can be viewed as a metric space and the geometry of this space plays a guiding role. In order to assess the performance of the procedure we carried out numerical experiments.

1 Introduction

Throughout this note the word graph is used for in the restricted meaning of finite simple graph, that is, each graph will have finitely many vertices and finitely many edges. Further, neither loops nor double edges may occur. Let $\mathsf{G} = (\mathsf{V}, \mathsf{E})$ be a finite simple graph, where V is the node set and E is the edge set of G. The set of edges E consists of unordered pairs of elements of V. The simplicity of the graph G means that it has neither double edges nor loops. The finiteness of the graph G means that the sets V and E have finitely many elements.

Computing Classification System 1998: G.2.2
Mathematics Subject Classification 2010: 05C15,05B45, 52C22
Key words and phrases: clique number, maximum clique, branching rules.

A subgraph Δ of G is called a clique in G if two distinct nodes of Δ are always adjacent in G. If the clique Δ has k nodes we will say that Δ is a k-clique in G. A node of G as a subgraph of G is of course a 1-clique and an edge of G as a subgraph can be viewed as a 2-clique. A k-clique Δ is maximal if it cannot be extended to a $(k+1)$-clique in G by adding a further node of G to Δ. A k-clique Δ in G is a maximum clique if G does not contain any $(k+1)$-clique. A maximum clique in G is always maximal in G but a maximal clique in G is not necessarily a maximum clique in G. For each finite simple graph G there is a number k such that G contains a k-clique but G does not contain any $(k+1)$-clique. This well defined number k is called the clique number of G and it is denoted by $\omega(G)$.

Problem 1 *Given a finite simple graph* $G = (V, E)$. *Determine* $\omega(G)$.

Problem 2 *Given a finite simple graph* $G = (V, E)$ *and given a positive integer* k. *Decide if* G *contains a* k*-clique.*

Problem 3 *Given a finite simple graph* $G = (V, E)$ *list all maximum cliques that appear in* G.

Problem 4 *Given a finite simple graph* $G = (V, E)$ *list all maximal cliques that appear in* G.

Problem 1 is referred to as the maximum clique problem. It is an optimization problem and by the complexity theory of the algorithms it belongs to the NP-hard complexity class. (For further details see [2].)

Problem 2 is referred to as the k-clique problem. It is a decision problem and by the complexity theory of the algorithms it belongs to the NP-complete complexity class. (For further details see [4].) The four problems above all have important applications in discrete applied mathematics.

Some of the clique search problems are optimization problems and many of these algorithms have the following outline. Using computationally affordable techniques upper and lower bounds for the clique number of the given graph are established. If the lower and upper estimates agree, then the clique number of the graph is computed. If there is a gap between the upper and lower estimates, then we divide the clique search instance into smaller instances. In other words one carries out an optimality test and when this test is inconclusive a branching takes place.

Let $G = (V, E)$ be a finite simple graph and let P, Q, R be subsets of the set of nodes of G. The ordered triplet (P, Q, R) is called a splitting partition of the graph G if the following conditions are all satisfied.

(1) $P \cup Q \cup R = V$.

(2) $P \neq \emptyset$, $R \neq \emptyset$.

(3) $P \cap Q = P \cap R = Q \cap R = \emptyset$.

(4) $p \in P$, $r \in R$ implies that the unordered pair $\{p, r\}$ is not an edge of the graph G.

Let H be the subgraph of G induced by the set of nodes $P \cup Q$ and let K be the subgraph of G induced by the set of nodes $Q \cup R$. Let us suppose that Δ be a clique in G. In [6] it was proved that either Δ is a clique in H or Δ is a clique in K. This result is in an intimate relation with clique search procedures.

Let us suppose that we are looking for a maximum clique in the graph G. By the observation above we may restrict our attention to look for a maximum clique in the smaller graphs H and K. The larger are the sizes of the sets P and R the smaller are the subgraph H and K. Thus setting up a computationally economic branching rule in a maximum clique or in a k-clique algorithm depends on our ability to locate a splitting partition in a computationally economic manner.

As the main result of this paper we will propose a method to speedily locate splitting partitions in a given graph. The procedure we propose is rather myopic and so there is no any guarantee that the procedure provides splitting sets with optimal P and R sets. Unfortunately we do not possess theoretical tools to establish performance measurements of the splitting set spotting algorithm. We will carry out numerical experiments to demonstrate that the procedure works reasonably well.

2 Metric spaces and splitting partitions

Let $G = (V, E)$ be a finite simple graph and let u and v be two nodes of G. Set $d(u, v)$ to be the length of a shortest path leading from node u to node v. If there is no path from node u to node v we set $d(u, v)$ to be ∞. It may happen that there are more than one shortest paths leading from u to v. But their lengths must be the same. The quantity $d(u, v)$ can play the role of a distance between the nodes of G and the graph G can be viewed as a metric space equipped with this distance function.

For a vertex v of G the set of nodes adjacent to v is called the set neighbors of v and it is denoted by $N(v)$. In notation $N(v) = \{u : u \in V, \{v, u\} \in E\}$. The number of the elements of the set $N(v)$ is referred to as the degree of the node

v and it is denoted by $\deg(v)$. In a more general setting for a vertex v of G and for a subset U of V we define the degree of v with respect to the subset U as the number of neighbors of v in the subset U. We denote this restricted degree of v by $\deg_U(v)$. Plainly, $\deg_U(v) = |N(v) \cap U|$ and further $\deg(v) = \deg_G(v)$.

Set $\mathrm{ball}_1(v) = \{v\} \cup N(v)$ and note that it is a ball of radius 1 centered at the point v in the metric space. For a subset U of V we define U^c to be the union of $\mathrm{ball}_1(u)$ as u ranges over the elements of U. We may call the set U^c the closure of the set U. Condition (4) in the definition of the splitting partition can be expressed coveniently in terms of closure of the sets involved.

Lemma 5 *Let $G = (V, E)$ be a finite simple graph and let P, Q, R be subsets of V such that the ordered triplet (P, Q, R) is a splitting partition of G. Then*

$$P^c \cap R = \emptyset, \quad P \cap R^c = \emptyset \tag{1}$$

must hold.

Proof. Let us assume assume on the contrary that the ordered triplet (P, Q, R) is a splitting partition of G and in addition $P^c \cap R \neq \emptyset$ holds. In this situation there is a $p \in P$ and an $r \in R$ such that $r \in \mathrm{ball}_1(p)$. It follows that the unordered pair $\{p, r\}$ is an edge of G. This contradicts condition (4) in the definition of the splitting partition. Assuming that $P \cap R^c \neq \emptyset$ a similar reasoning gives the contradiction again that the unordered pair $\{p, r\}$ is an edge of G. □

Lemma 6 *Let $G = (V, E)$ be a finite simple graph and let P, R be subsets of V. Suppose that beside condition (1) in Lemma 5 the condition*

$$P \neq \emptyset, \quad R \neq \emptyset \tag{2}$$

also holds. Then setting $Q = V \setminus (P \cup R)$ the ordered triplet (P, Q, R) is a splitting partition of G.

Proof. It is easy to see that each of the conditions (1), (3) in the definition of the splitting partition holds. Clearly, condition (2) in the definition of the splitting partition holds as a consequence of condition (2) in Lemma 6.

It remains to show that condition (4) in the definition of the splitting partition also holds. In order to do so assume on the contrary that condition (4) does not hold, that is, there is a $p \in P$ and an $r \in R$ such that the unordered pair $\{p, r\}$ is an edge of G. In this situation $r \in \mathrm{ball}_1(p)$ and consequently $P \cap R^c \neq \emptyset$. This is in contradiction with the first part of condition (1) in Lemma 5. □

Lemma 7 *Let* $G = (V, E)$ *be a finite simple graph and let* P *be a subset of* V. *If* P *satisfies the condition*

$$P \neq \emptyset, \quad P^c \neq V. \tag{3}$$

Then setting $R = V \setminus P^c$ *for the sets* P *and* R *condition (1) in Lemma 5 and condition (2) in Lemma 6 hold.*

Proof. As $P \neq \emptyset$ holds by assumption, we need to prove only $R \neq \emptyset$ to get condition (2) in Lemma 6. But $R \neq \emptyset$ is a consequence of the assumption $P^c \neq V$.

The way the set R is constructed from P^c shows that the equation $P^c \cap R = \emptyset$ must hold. The equation $P \cap R^c = \emptyset$ follows from $P^c \cap R = \emptyset$. This gives that condition (1) in Lemma 5 is satisfied. □

By Lemmas 5, 6, 7, constructing a splitting partition (P, Q, R) for G can be reduced to finding a subset P of V satisfying condition (3) in Lemma 7. This condition can be satisfied easily. For example the choice $P = \{v\}$ is a suitable choice whenever v is node of G that is not adjacent to at least one node of G. In this case $P = \{v\}$, $Q = N(v)$, $R = V \setminus (\{v\} \cup N(v))$. In fact, the splitting partition (P, Q, R) constructed in this way is the most commonly used branching rule in clique search algorithms. It is part of the Carraghan-Pardalos algorithm [1] and it is part of the Östergård algorithm [3]. A splitting partition (P, Q, R) for which either $|P| = 1$ or $|R| = 1$ is coming free of charge. From this reason we call such splitting partition of G a trivial splitting partition.

If (P, Q, R) is a splitting partition for G we may construct a new splitting partition (P', Q', R') for G. We set $U = Q \cup R$ and locate a node u of U for which $\deg_R(u)$ is a minimum. Then we move u from U to P and move the neighbors of u in R to Q to get the sets P', Q', R'. For the sake of simplicity we may use the initial setting $P = \emptyset$, $Q = \emptyset$, $R = V$ and construct new triplets (P, Q, R) while the condition $|P| < |R|$ holds.

3 Two small size examples

In order to illustrate the results presented so far we work out a small size example in details.

Example 8 *Let us consider the graph* $G = (V, E)$. *Here* $V = \{1, \ldots, 6\}$. *The adjacency matrix of* G *is depicted in Table 2. Figure 1 shows a geometric representation of* G.

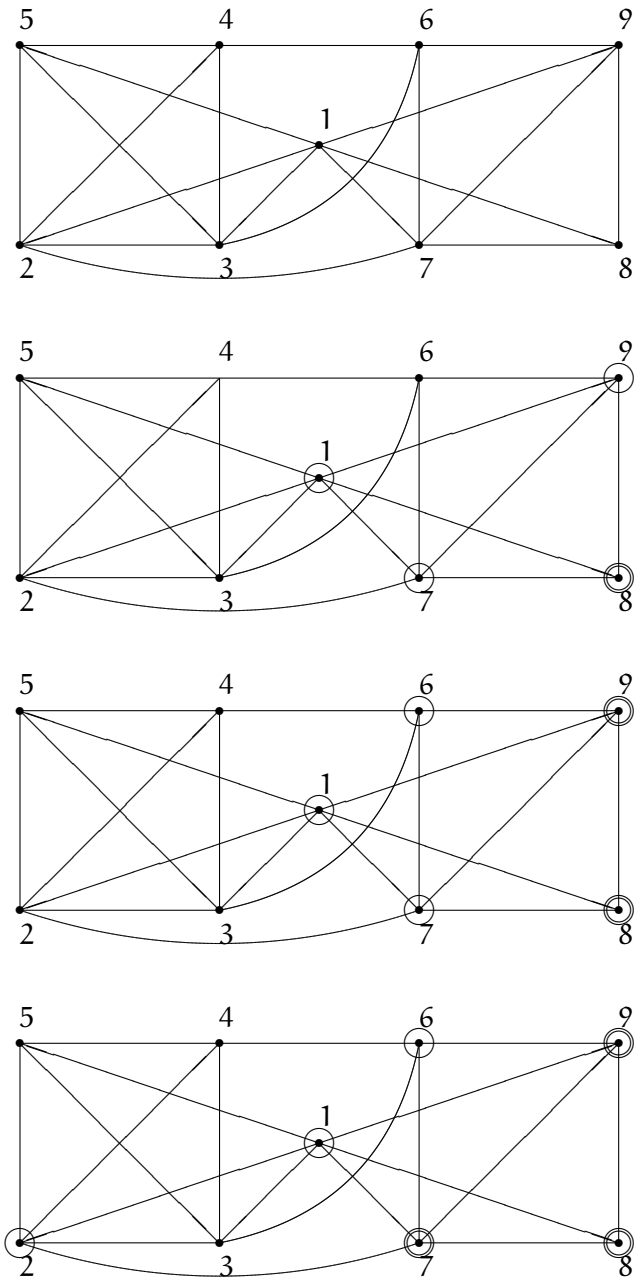

Figure 1: A graphical representation of the graph G in Example 8 and the steps of the procedure of spotting a splitting partition.

	1	2	3	4	5	6	7	8	9
1	×	●	●		●		●	●	●
2	●	×	●	●	●		●		
3	●	●	×	●	●	●			
4		●	●	×	●	●			
5	●	●	●	●	×				
6			●	●		×	●		●
7	●	●				●	×	●	●
8	●						●	×	●
9	●					●	●	●	×

	3	4	5	1	2	6	7	8	9
3	×	●	●	●	●	●			
4	●	×	●		●	●			
5	●	●	×	●	●				
1	●		●	×	●		●	●	●
2	●	●	●	●	×		●		
6	●	●				×	●		●
7				●	●	●	×	●	●
8				●			●	×	●
9				●		●	●	●	×

Table 1: The adjacency matrices of the graph in Example 8. In the second adjacency matrix we rearranged the rows and columns to make the splitting partition more apparent.

The reader can verify easily that the triplet (P, Q, R) of the subsets

$$P = \{3, 4, 5\}, \quad Q = \{1, 2, 6\}, \quad R = \{7, 8, 9\} \tag{4}$$

is a splitting partition of the graph G. Note that upper right and the lower left three by three submatrices are unfilled in the second adjacency matrix in Table 2.

We try to construct a splitting partition (P, Q, R) for the graph G. We set

$$P = \emptyset, \quad Q = \emptyset, \quad R = \{1, \ldots, 9\}.$$

The conditions (1), (3), (4) in the definition of splitting partition are satisfied. Condition (2) is not satisfied. We compute the degree of each node in $U = Q \cup R$ with respect to the set R.

node	1	2	3	4	5	6	7	8	9
degree	6	5	5	4	4	4	5	3	4

Node 8 has a minimum degree. We move node 8 from set R to set P. We move the neighbors of 8 from set R to set Q. In this way we get

$$P = \{8\}, \quad Q = \{1, 7, 9\}, \quad R = \{2, 3, 4, 5, 6\}.$$

The conditions (1), (2), (3), (4) in the definition of splitting partition are satisfied and consequently we have a genuine splitting partition of G. The

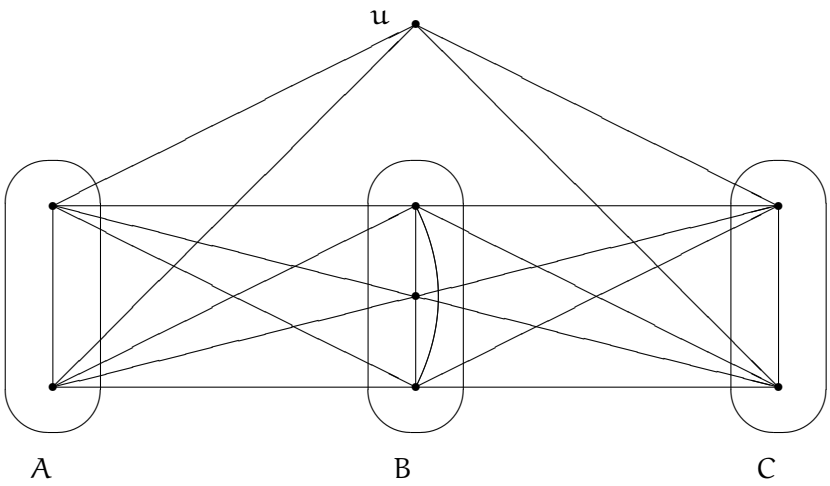

Figure 2: The graph G in Example 9.

sizes of the sets P and R are far from each other. We try to enlarge |P| even if this results a smaller |R|.

We compute the degree of each node in U = Q ∪ R with respect to the set R.

node	1	7	9	2	3	4	5	6
degree	3	2	1	3	4	4	3	2

Node 9 has a minimum degree. We move node 9 from set Q to set P. We move the neighbors of 9 from set R to set Q. In this way we get

$$P = \{8, 9\}, \quad Q = \{1, 7, 6\}, \quad R = \{2, 3, 4, 5\}.$$

The conditions (1), (2), (3), (4) in the definition of splitting partition are satisfied and consequently we have a splitting partition of G where the difference between |P| and |R| is reduced.

We compute the degree of each node in U = Q ∪ R with respect to the set R.

node	1	7	6	2	3	4	5
degree	2	1	1	3	3	4	3

Node 7 has a minimum degree. We move node 7 from set Q to set P. We move the neighbors of 7 from set R to set Q. In this way we get the splitting partition

$$P = \{7, 8, 9\}, \quad Q = \{1, 2, 6\}, \quad R = \{3, 4, 5\}$$

	1	2	3	4	5	6	7	8
1	×	•	•				•	•
2	•	×	•	•	•	•		
3	•	•	×	•	•	•		
4		•	•	×	•	•	•	•
5		•	•	•	×	•	•	•
6		•	•	•	•	×	•	•
7	•			•	•	•	×	•
8	•			•	•	•	•	×

Table 2: The adjacency matrix of the graph G in Example 9.

of G. This splitting partition is essentially the same as (4). The steps of the procedure can be followed on the geometric version of the graph G. Figure 1 shows these steps. The elements of the set R are marked with a double circle and the elements of the set Q are marked with a simple circle. Finally the elements of the set P are left unmarked.

We exhibit now an example to illustrate that the algorithm for spotting splitting partition described in the paper is a myopic one. Let A, B, C be pair-wise disjoint sets and let u be an element such that $u \notin (A \cup B \cup C)$. Let us assume that $|A| = |C| = n$ and $|B| = n + 1$. Using the sets A, B, C, $\{u\}$ we construct a graph $G = (V, E)$. We set $V = A \cup B \cup C \cup \{u\}$. We draw edges between nodes such that the subgraph induced by the set $A \cup B$ is a clique in G and similarly the subgraph induced by the set $B \cup C$ is a clique in G. Finally, we connect node u to each node in $A \cup C$. The reader can verify that with the $P = A$, $Q = B$, $R = C$ choices the ordered triplet (P, Q, R) is a splitting partition of G. Here $|P| = n$ and $|R| = n$. On the other hand, the greedy algorithm proposed by the paper will locate the splitting partition (P, Q, R), where $P = \{u\}$, $Q = A \cup C$, $R = B$. Here $|P| = 1$ and $|R| = n + 1$. We can see that for $n \geq 2$ the graph G has a non-trivial splitting partition. But the greedy algorithm locates a trivial splitting partition. The $n = 2$ particular case of the above construction is the content of the next example.

Example 9 *Set* $A = \{2, 3\}$, $B = \{4, 5, 6\}$, $C = \{7, 8\}$, $u = 1$. *Let us consider the graph* $G = (V, E)$, *where* $V = \{1, \ldots, 8\}$. *The adjacency matrix of* G *is depicted in Table 3. Figure 2 shows a possible geometric representation of* G.

4 Numerical experiments

For testing purposes we have selected three infinite families of graphs that are connected to the existence and construction of certain error detecting and error correcting codes. The so-called monotonic matrices are in intimate connection with codes over the alphabet $\{1, \ldots, n\}$. Each code words has length three. The problem is to find a code whose inner distance is at least two. (See [6], [8].) The deletion error detecting codes are consisting of binary code words of length n. These words are sent over a noisy channel. Due to transmission error on the receiver side a shorter word may arrive. The task is to devise a code that makes possible to detect a one bit deletion error. (For further details see [5].) The Johnson codes we are considering here are binary codes with word length n. Each code word consists of 4 1's and $n-4$ 0's. The Hamming distance of two distinct code words is at least 3.

Monoton				Deletion				Johnson			
n	$\lvert V\rvert$	α	β	n	$\lvert V\rvert$	α	β	n	$\lvert V\rvert$	α	β
3	27	4	4	3	8	2	4				
4	64	5	7	4	16	4	4				
5	125	7	8	5	32	4	5				
6	216	9	9	6	64	4	5	6	15	2	4
7	343	10	12	7	128	4	7	7	35	2	5
8	512	12	13	8	256	5	5	8	70	2	6
9	729	14	14	9	512	5	8	9	126	3	3
10	1 000	15	17	10	1024	6	6	10	210	3	4
11	1 331	17	18	11	2048	6	10	11	330	4	4
12	1 728	19	19	12	4096	7	7	12	495	4	5
13	2 197	20	22					13	715	5	5
14	2 744	22	23					14	1 001	5	6
15	3 375	24	24					15	1 365	6	6
16	4 096	25	27					16	1 820	6	7
17	4 913	27	28					17	2 380	7	7
								18	3 060	7	8
								19	3 876	8	8
								20	4 845	8	9

Table 3: Numerical results in connection with graphs coming from coding theory.

The results of the numerical experiments are summarized in the Table 3. We describe the meaning of the entries using the 10-th row of Table 3 as an illustration. A graph G is associated with a monotonic matrix of parameter $n = 10$. The graph has $|V| = 1000$ vertices. These values are in the first two columns of the table. The splitting partition (P, Q, R) we have spotted has the parameters $|P| = \alpha = 15$, $|R| = \beta = 17$ and the next two columns contain these α, β values.

At this stage we may conclude that the algorithm spots splitting partitions rapidly and works reliably in connection with non-trivial size graphs. Only after working with the algorithm for a longer period of time involving a much wider variety and range of graphs would enable us to assess the merits of the proposed procedure.

References

[1] R. Carraghan, P. M. Pardalos, An exact algorithm for the maximum clique problem, *Operation Research Letters* **9** (1990), 375–382. ⇒135

[2] M. R. Garey, D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-completeness*, Freeman, New York, 2003. ⇒132

[3] P. R. J. Östergård, A fast algorithm for the maximum clique problem, *Discrete Applied Mathematics* **120** (2002), 197–207. ⇒135

[4] C. H. Papadimitriou, *Computational Complexity*, Addison-Wesley Publishing Company, Inc., Reading, MA 1994. ⇒132

[5] N. J. A. Sloane, Challenge Problems: Independent sets in graphs, http://neilsloane.com/doc/graphs.html ⇒140

[6] S. Szabó, Parallel algorithms for finding cliques in a graph,, *Journal of Physics, Conference Series* **268** (2011) 012030 DOI:10.1088/1742-6596/268/1/012030. ⇒131, 133, 140

[7] S. Szabó, Monoton matrices and finding cliques in a graph, *Annales Univ. Sci. Budapest., Sect. Computatorica* **41** (2013), 307–322. ⇒

[8] E. W. Weisstein, Monotonic Matrix, In: *MathWorld–A Wolfram Web Resource.* http://mathworld.wolfram.com/MonotonicMatrix.html ⇒140

Received: September 26, 2019 • Revised: October 19, 2019