A Computational Study for the Graph-Theoretic Version of the Union-Closed Sets Conjecture

M. I. Moussa
Computer Science Department, Faculty of Computers & Information, Benha University, Benha, Egypt,

E. M. Badr
Department of scientific computing, Faculty of Computers & Information, Benha University, Benha, Egypt,

ABSTRACT
An induced subgraph S of a graph G is called a derived subgraph of G if S contains no isolated vertices. An edge e of G is said to be residual if e occurs in more than half of the derived subgraphs of G. We prove some theorems which calculate the number of derived subgraphs for some special graphs. We also present a new algorithm *SDSA* that calculates the number of derived subgraphs for a given graph G and determines the residual and non-residual edges. Finally, we introduce a computational study which supports our results.

Keywords
Union closed sets conjecture, induced graphs, derived subgraphs.

1. INTRODUCTION
A union-closed family of sets $\mathcal{A}$ is a finite collection of sets not all empty such that the union of any two members of $\mathcal{A}$ is also a member of $\mathcal{A}$. The following Conjecture is due to Peter Frankl [1, 2, 3].

Conjecture 1. Let $\mathcal{A} = \{ A_1, A_2, \ldots, A_n \}$ be a union-closed family of n distinct sets. Then there exists an element which belongs to at least $n / 2$ of the sets in $\mathcal{A}$.

Let $A = \cup A_i$. If we replace each set A_i by $B_i = A - A_i$ then we get an intersection-closed family of sets, which we call the dual family of A. Therefore Conjecture 1 is equivalent to the following.

Conjecture 2. Let $\mathcal{B} = \{ B_1, B_2, \ldots, B_n \}$ be an intersection-closed family of n distinct sets. Then there exists an element which belongs to at most $n / 2$ of the sets in $\mathcal{B}$.

An induced subgraph S of a graph G is called a derived subgraph of G if S contains no isolated vertices. An edge e of G is said to be residual if e occurs in more than half of the derived subgraphs of G otherwise e is non-residual. Let $D(G)$ denote the set of derived subgraphs of G and put $n_d(G) = |D(G)|$. A graph-theoretic version of the union-closed sets conjecture due to El-Zahar [4]. He formulated a weaker version of Conjecture 1 specialized for graphs as the following.

Conjecture 3. Every non-empty graph contains a non-residual edge.

***Example* 1.** The derived subgraphs of C_6 are $\varnothing$, C_6 and the subgraphs S_1, S_2, … , S_5 together with their cyclic permutations as shown in Figure 1. In all, we have $n_d(C_6) = 29$ (compare to 64 induced subgraphs of C_6). Each edge of C_6 is contained in exactly 12 derived subgraphs and, therefore, is non-residual.

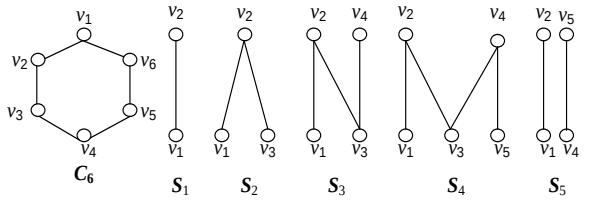

Fig.1: ***Derived Subgraphs of C_6***

***Example* 2.** Consider the graph G_1 of Figure 2. This graph has $n_d(G_1) = 34$. Each of the edges e_1, e_2, e_3 occur in 18 derived subgraphs so that it is residual. The remaining edges are non-residual belonging only to 13 derived subgraphs.

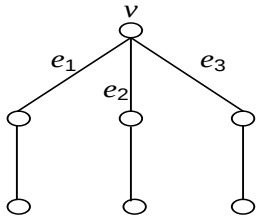

Fig. 2: ***Residual Edges***

B. Llano et al proved that the every simple graph with at least one edge contains a non-residual edge (Conjecture 3) [5].

In this work , our aim is to introduce a computational study for derived subgraphs.

In Section 2, we prove some theorems which calculate the number of derived subgraphs for some special graphs. In Section 3, we present a new algorithm *SDSA* that calculates the number derived subgraphs for a given graph G and determines the residual and non-residual edges. In Section 4 we give a computational study which supports our results. Finally, we give our conclusions and discuss possible extensions of the algorithm.

2. THE MAIN RESULTS
In this Section, we prove some theorems which calculate the number of derived subgraphs for some special graphs using the Principle of Mathematical Induction and the Principle of Inclusion-Exclusion.

Theorem 1. Let P_n be the path v_1, v_2, v_3, …, v_n . Then the number of derived subgraphs of P_n is given by the following relation :

$$n_d(P_n) = \begin{cases} 1 & \text{if } n = 0,1 \\ 2 & \text{if } n = 2 \\ 2n_d(P_{n-1}) - n_d(P_{n-2}) + n_d(P_{n-3}) & \text{if } n \geq 3 \end{cases} \quad (1)$$

Proof. We prove this theorem by the Principle of Mathematical Induction. The proof will now proceed in two steps : the initial step and the inductive step.

Initial Step. It is clear that the statement (1) is true for n = 0, 1, 2.

Inductive Step. Here we must prove that the following assertion: " If there is a positive integer k such that $n_d(P_k) = 2n_d(P_{k-1}) - n_d(P_{k-2}) + n_d(P_{k-3})$ is true then $n_d(P_{k+1}) = 2n_d(P_k) - n_d(P_{k-1}) + n_d(P_{k-2})$ is true. Thus, we assume there is a positive integer k such that

$$n_d(P_k) = 2n_d(P_{k-1}) - n_d(P_{k-2}) + n_d(P_{k-3}) \quad (2)$$

Since $n_d(P_k)$ is the number of all derived subgraphs of P_k so by adding one vertex v_{k+1} to the relation (2), we will $n_d(P_{k+1})$ as follows:

$$n_d(P_k \cup \{v_{k+1}\}) =$$
$$2n_d(P_{k-1} \cup \{v_{k+1}\}) - n_d(P_{k-2} \cup \{v_{k+1}\}) + n_d(P_{k-3} \cup \{v_{k+1}\})$$

then $n_d(P_{k+1}) = 2n_d(P_k) - n_d(P_{k-1}) + n_d(P_{k-2})$ so the relation (1) is true for $n \geq 0$. ■

Theorem 2 . Let C_n be a cycle on $n \geq 3$ vertices. Then the number of derived subgraphs of C_n is given by the relation :

$$n_d(C_n) = n_d(P_{n-1}) + 2(n-1) + \sum_{i=2}^{n-3} n_d(P_{n-i-2}) .$$

Proof. Let C_n be the cycle v_1, v_2, . . . , v_n, v_1 . Let x_1 denote the number of derived subgraphs of C_n not containing v_1, then x_1 = $n_d(P_{n-1})$. On the other hand, let x_2 denote the number of derived subgraphs which contain v_1 . Such a derived subgraph contains a path P_i of length (i – 1) that contains v_1 and a derived subgraph of path P_i of order (n – i – 2), where $2 \leq i \leq n-1$. Then for fixed i this number is $i\, n_d(P_{n-i-2})$. Thus

$$x_2 = \sum_{i=2}^{n-3} i\ n_d(P_{n-i-2}) + (n-2) + (n-1)$$

Moreover C_n is a derived subgraph of itself, therefore $n_d(C_n)$ = x_1 + x_2 + 1 so

$$n_d(C_n) = n_d(P_{n-1}) + 2(n-1) + \sum_{i=2}^{n-3} i\ n_d(P_{n-I-2}) \quad ■$$

Theorem 3. Let $G(n, n)$ be a bipartite graph with two partitioning sets V_1 and V_2, where $|V_1| = |V_2| = n$ and $deg(v) = n$ – 1 for each $v \in V(G(n, n))$. Then the number of derived subgraphs of $G(n, n)$ is given by the relation : $n_d(G(n, n)) = 2^{2n} + n + 2 - n2^n - 2^{n+1}$, and each edge $uv \in E(G(n,n))$ is contained in exactly $2^{n-1}(2^{n-1} - 1)$ derived subgraphs.

Proof. Let $V_1 = v_1, v_2, \ldots, v_n$ and $V_2 = u_1, u_2, \ldots, u_n$ where $u_i v_i \notin E(G(n, n))$ for each i = 1, 2, . . . , n.

To form a derived subgraph, we take $S_1 \cup S_2$ where $S_i \subset V_i$ for i = 1, 2. If $|S_1| \geq 2$ and $|S_2| \geq 2$ then we get a derived subgraph.

If $|S_1|$ =1, say, $S_1 = v_i$ then $\phi \neq S_2 \cup V_2 \setminus \{u_i\}$. This shows that

$$n_d(G(n, n)) = (2^n - n - 1)^2 + 1 + 2n(2^n - 1)$$

Now we fix an i = 1, 2, . . . , n. We count the number of derived subgraphs which contain the edge v_1u_i . Such derived subgraph will have the form $S_1 \cup S_2$ where $v_1 \in S_1 \subset V_1$.

Again if $|S_1| \geq 2$ and $|S_2| \geq 2$ then we have a derived subgraph.

If, say $S_1 = \{v_1\}$ then $u_1 \notin S_2$. This shows that the number of derived subgraphs which contain v_iu_i is equal to $(2^{n-1} - 1)^2 + 2(2^{n-2}) - 1 = 2^{2n-2} - 2^{n-1}$.■

3. THE SERIAL DERIVED SUBGRAPH ALGORITHM

In this Section, we introduce a serial derived subgraphs algorithm *SDSA* which calculates the number of derived subgraphs for a given graph G . The algorithm also determines the residual and non-residual edges. The parameters of the algorithm are :

A[i, j] : the adjacency matrix of G.

S [i] : all of the subsets of $V(G)$.

(i,j) : the entry of the matrix $E(i,j)$ which is equal to the number of derived subgraphs that contain v_iv_j .

total : the number of all derived subgraphs of G.

Let G be a graph which has n vertices and m edges. We can represent the graph G by the Adjacency-Graph class, where $a[i][j]$ is the entry element (i,j) in the adjacency matrix A. The algorithm finds all subsets of the vertex set $V(G)$; then it checks if the current subset induces a derived subgraph or not. The algorithim finds the number of derived subgraphs that contain any edge $e \in E(G)$.

Our main algorithm SDSA calls three procedures Initialize-Subset, Get-Next-Subset and Check-Subset as follows:

Algorithm 1: A serial derived subgraphs algorithm *SDSA*

Input : $A[i][j]$ the adjacency matrix of G.

Output : (total) the number of all derived subgraphs of G.

1**:** Call Algorithm2 (Initialize-Subset)

2: **while** Not Done **do**

3**:** Call Algorithm3 (Get-Next-Subset)

4: Call Algorithm4 (Check-Subset)

5: **if** DERIVED **then**

6: $total \leftarrow total + 1$

7: **for** i=1 $\rightarrow$ n **do**

8: **for** j=i+1 $\rightarrow$ n **do**

9: **if** $S[i] = S[j] = 1$ **then**

10: $E[i,j] \leftarrow E[i,j] + 1$

11: **end if**

12**:** **end for**

13: **end for**

14: **end if**

15**: end while**

16: **Return** total

17: For each $e = (v_i, v_j)$ if $E[i,j] > total / 2$ the edge e is residual otherwise is non-residual.

The Initialize-Subset procedure initializes the initial subset of $V(G)$ as array $S[j] = 0$. The subgraph induced by the initial S is the empty derived subgraph. We outline below the initialize-Subset procedure which considers the empty subgraph as the first derived one.

Algorithm 2: Initialize-Subset

1: Take the empty set to be the initial subset S

2: Set the value of $total = 1$

3: For every edge $e = (i, j)$ let $E[i,j] = 0$

4: Done $\leftarrow$ False

The Get-Next-Subset procedure generates all subsets of $V(G)$ by the method is known as

a binary counting representation.

Algorithm 3: Get-Next-Subset

1: $j \leftarrow n + 1$

2: **repeat**

3: $j \leftarrow j - 1$

4: **until** (($S[j] = 0$) or ($j = 0$)

5: **if** $j \neq 0$ **then**

6: $S[j] \leftarrow 1$

7: $MAX \leftarrow j$

8: **for** $i = MAX + 1 \rightarrow n$ **do**

9: $S[i] = 0$

10: **end for**

11: **else**

12: Done $\leftarrow$ True

13: **end if**

The Check-Subset verifies the current subset S as a derived subgraph or not. A precise description of this process is the following.

Algorithm 4: Check-Subset

1: DERIVED $\leftarrow$ False

2: $count \leftarrow 1$

3: **for** $k = 1 \rightarrow n$ **do**

4: **if** $S[k] = 1$ **then**

5: $sum = 0$

6: **for** $j = 1 \rightarrow n$ **do**

7: $sum = sum + a[k][j] * S[j]$

8 : **if** $sum \neq 0$ **then**

9: $sum \leftarrow 1$

10: $count \leftarrow count * sum$

11: **end if**

12: **end for**

13: **end if**

14: **end for**

15: **if** $count \neq 0$ **then**

16: DERIVED $\leftarrow$ True

17: **end if**

4- COMPLEXITY ANALYSIS AND COMPUTATIONAL RESULTS

The *SDSA* algorithm can be shown to run in $O(n2^n)$ time where n is the number of vertices in the given graph G. There are 2^n subsets of $V(G)$. We check every one by calling Algorithm 4 (Check-Subset)**.** The Algorithm 3 (Get-Next-Subset) requires $O(n)$ time. We need exactly 2^n-1 calls of Algorithm 3 (Get-Next-Subset), each one runs in $O(n)$ time. Then the total running time of *SDSA* algorithm is $O(n2^n)$ sequential time.

The algorithm describe in Section 3 has been experimentally implemented. In this Section, the numerical experiments are presented. It must be mentioned that the computational results demonstrates the number of derived subgraphs and its residual edges for some special graphs.

Our numerical experiments were performed on a PC with 2.000 MHz Pentium 4 processor, RAM 512 Mb and windows XP operating system. Our implementation was done under the C environment.

Columns of the following tables contain problems size ***n*** and ***m***, number of all derived subgraphs ***No_DS***, number of all non-residual edges ***No_NRE*** and the running time ***CPU Time (secs.)*****.**

Table 1. ***Derived subgraphs for the path graph*** P_n

n	***No_DS***	***No_NRE***	***CPU Time (secs.)***
2	2	1	0.000181
4	7	2	0.000367
6	21	5	0.001085
8	65	7	0.004173
10	200	9	0.018365
12	616	11	0.079329
14	1897	13	0.352686
16	5842	15	1.74612
18	17991	17	7.962174
20	55405	19	33.045809

Table 2. Derived subgraphs for the cyclic graph C_n

n	***No_DS***	***No_NRE***	***CPU Time (secs.)***
4	10	4	0.000356
6	29	6	0.001080
8	90	8	0.004166
10	277	10	0.17741
12	853	12	0.079052
14	2627	14	0.355268
16	8090	16	1.593558
18	24914	18	7.371244
20	76725	20	33.170715

Table 3. Derived subgraphs for the complete graph K_n

n	***No_DS***	***No_NRE***	***CPU Time (secs.)***
4	12	6	0.000358
6	58	15	0.001103
8	248	28	0.004330
10	1014	45	0.018671
12	4084	66	0.084653
14	16370	91	0.387049
16	65520	120	1.754086
18	262126	153	8.062705
20	1048556	190	37.126845

Table 4. Derived subgraphs for the complete bipartite graph $K_{m,n}$

m	***n***	***No_DS***	***No_NRE***	***CPU Time (secs.)***
1	1	2	1	0.000185
2	2	10	4	0.000365
3	3	50	9	0.001098
4	4	226	16	0.004296
5	5	962	25	0.018593
6	6	3970	36	0.084717
7	7	16130	49	0.385195
8	8	65026	64	1.753573
9	9	261122	81	8.358592
10	10	1046530	100	37.569930

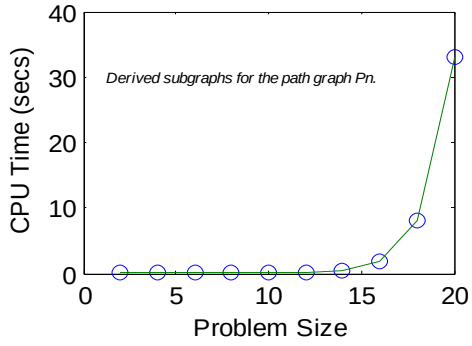

Fig.3: Executable time for P_n graph by *SDSA*

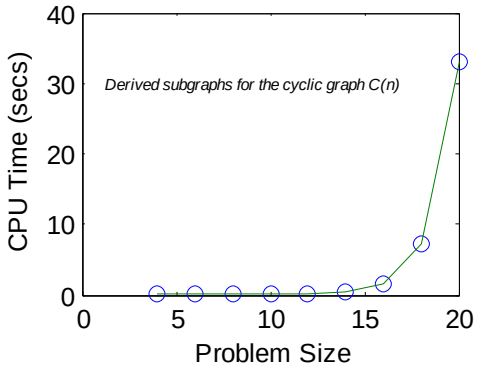

Figure 4. Executable time for C_n graph by *SDSA*

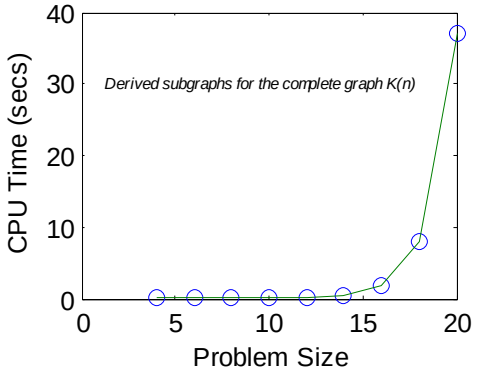

Figure 5. Executable time for K_n graph by *SDSA*

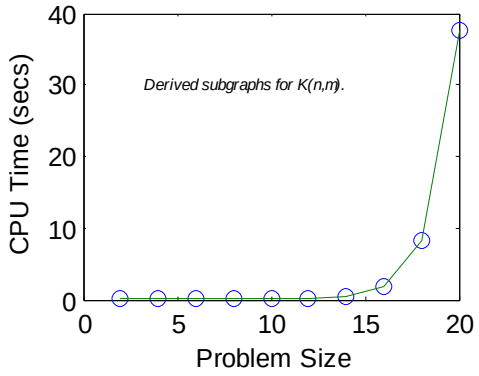

Figure 6. Executable time for $K_{n,m}$ graph by *SDSA*

From the above Tables, we can note that every graph for any size contains a non-residual edge (Conjecture 3) while from the above Plots, we can see that the running time of the algorithm *SDSA* increases as an exponential time in the problem size *n*.

5. CONCLUDING REMARKS

In this work we proved some theorems which calculate the number of derived subgraphs for some special graphs. We also presented a new algorithm *SDSA* that calculates the number of derived subgraphs for a given graph *G* and determines the residual and non-residual edges. Finally, we introduced a computational study which supports our results. Our algorithm *SDSA* has $O(n2^n)$ sequential time so there is room for its further improvement as an sequential or parallel algorithm. These possible improvements will be the subject of our future work.

6. REFERENCES

[1] I. Rival (Ed.), ***Graphs And Order***, Reidel, Dordrecht-Boston,(1985), p.25.

[2] R. P. Stanley, ***Enumerative Combinatorics***, vol. I, Wadsworth & Broks/Cole, Belmont, CA, (1986).

[3] B. Poonen, ***Union-Closed Families***, J. Combin. Theory, A 59 (1992), 253-268.

[4] M. H. El-Zahar , ***A Graph-Theoretic Version Of The Union-Closed Sets Conjecture***, J.Graph Theory 26 (1997), no. 3, 155-163.

[5] B. Llano, J. Montellano-Ballesteros, E. Rivera-Campo and R. Strauz " On Conjecture of Frankl and El-Zahar" J. Graph Theory 57: 344-352 (2008).

[6] G. Chartrand and L. Lesniak " ***Graphs & Digraphs***" (third edition) Chaman & Hall, London, (1996) .