Qubahan Academic Journal
گوڤارا قبهان یا ئەکادیمی
مجلة قبهان الأكادمية
Doi: 10.48161/issn.2709-8206

Complexity of Computation of Dominating Sets in Geo-Mathmetics Algorithm : A Review

Şakir Işleyen
Department of Economic, Van
Yüzüncü Yıl Üniversitesi,
Van, Turkey
sakirisleyen@yyu.edu.tr
https://doi.org/10.48161/qaj.v1n1a41

Abstract- In this paper, the complexity on dominating sets of the graph is suppose the G = (V, E) is a subset D of V each head not in D is adjacent to one member on the dominating number γ (G) is the number of vertices in the smallest dominant sets of G. The dominant sets problem by testing whether γ (G) ≤ K of a given graph is G and K input; It is an electronic card NP machines decision problem in computational complexity theory. Infographics, powerful infographics plus graphic mapping. In each example, each white head is adjacent to at least one red cape, and the white cap is said to be dominated by the red cape. The graph in graph is 2: The histogram is an example that illustrates the histogram.Keywords— Boundary Value Problem, Convergence of the Method, Cubic Order, Finite Difference Method, Non-uniform Step Length.

*Keywords**: Dominating Sets, Complexity, Algorithms.***

I. INTRODUCTION

This study discuss the dominating sets on complexity algorithms such as for group-forming, tracking, or discovering nodes. In static or slow-developing groups, the solution to this problem often involves the expense of a dominant set of the groups. Dominant group of a grid (diagram). And with the node group V is a subset of nodes S ⊆ so that each node not present in S is adjacent to at least one node in S. An example of problems showing that the dominant group (or some of its variants) plays a role is determining the optimal sensor position to detect Disease outbreak, set control and social influence prevalence. The smallest dominant set of a group is the smallest subset of nodes[1,2].

Here the basic concepts, and let $\boldsymbol{G} = (\boldsymbol{V}, \boldsymbol{E})$ be an undirected simple graph, i.e., without loops and without multiple edges. **For V 0** $\subseteq \boldsymbol{V}, \boldsymbol{G}[\boldsymbol{V}\ \mathbf{0}]$ **denotes the subgraph of G induced by V 0** . The vertex- **set of G**[**V 0**] **is V 0** and the edge-set consists of those edges of G with both end points in V 0 [3]. The open neighborhood of a vertex $\boldsymbol{v} \in \boldsymbol{V}$ is denoted by $\boldsymbol{N}(\boldsymbol{v}) = \{\boldsymbol{u} \in \boldsymbol{V} : \{\boldsymbol{u}, \boldsymbol{v}\} \in \boldsymbol{E}\}$ and the closed neighborhood or simply neighborhood of vis denoted $\boldsymbol{by}\ \boldsymbol{N}[\boldsymbol{v}] = \boldsymbol{N}(\boldsymbol{v}) \cup \{\boldsymbol{v}\}$. The degree of a vertex v equals the cardinality of N(v). The neighborhood of a $\boldsymbol{set}\ \boldsymbol{S} \subseteq \boldsymbol{V}$ is defined as $\boldsymbol{N}[\boldsymbol{S}] = \cup\, \boldsymbol{s} \in \boldsymbol{SN}[\boldsymbol{s}]$. For r ∈ N, the rth neighborhood of $\boldsymbol{v} \in \boldsymbol{V}$ is defined recursively as $\boldsymbol{Nr}[\boldsymbol{v}] = \boldsymbol{N}[\boldsymbol{Nr} - \mathbf{1}[\boldsymbol{v}]]$, where N1[v] = N[v]. The $\boldsymbol{distance}\ \boldsymbol{d}\ (\boldsymbol{v}, \boldsymbol{w})$ between two vertices v, w ∈ V is the number of edges (or hops) that must be traversed to go from v to w on a shortest path. Similarly, the distance between two sets $\boldsymbol{A}, \boldsymbol{B} \subseteq \boldsymbol{V}$ is defined as the distance between two closest elements a ∈ A and b ∈ B, i.e., d(A, B) = min $\{\boldsymbol{d}(\boldsymbol{x}, \boldsymbol{y}) : \boldsymbol{x} \in \boldsymbol{A}, \boldsymbol{y} \in \boldsymbol{B}\}$, [7]. Two sets S , $\boldsymbol{T} \subseteq \boldsymbol{V}$ are called 2-separated if and only if $\boldsymbol{d}\ (\boldsymbol{S}, \boldsymbol{T}\ ) \geq \mathbf{3}, [\mathbf{7}]$. A set D ⊆ V of a simple graph G is called a dominating set if every vertex $\boldsymbol{v} \in \boldsymbol{V} \backslash \boldsymbol{D}$ is adjacent to some vertex u ∈ D. The domination number of the graph G which is usually denoted by γ (G), is the cardinality of a smallest dominating set of G and such set is called a minimum dominating set of G. The dominating set problem is to determine γ (G) and to find a dominating set of minimum cardinality[4]. Let P(V) be the set of all subsets of vertices belonging to V. **Let D** : $\mathbf{P}(\mathbf{V}) \rightarrow$ $\mathbf{P}(\mathbf{V})$ **be a function defined by**: **for every W** $\in \mathbf{P}(\mathbf{V}), \mathbf{D}(\mathbf{W})$ is a dominating set having minimum cardinality in G[W] and thus D(W) ⊆ W. Now, we introduce the definition of the Las Vegas randomized **approximation algorithm for an optimization** proble m. The terms which are used in the description of an optimization problem U are defined as follows, [2]. Let ΣI and ΣO be the set of input and output alphabets of U, respectively.

The language of feasible problem instances is denoted by L ⊆ Σ∗ I , (all strings over ΣI $\boldsymbol{with}$ **any length k**, $\mathbf{k} \geq \mathbf{0}$). LI ⊆ L is the language of the (actual) problem instances of U. M is a function from L **to P** ($\boldsymbol{\Sigma} *$ **O**) **and for every x** $\in$ **L**, **M**(**x**) **is call** ed the set of feasible solutions for x. For every pair (u, x), cost is the cost function that assigns a positive real number cost (u, x), where u ∈ M(x) for some x ∈ L. By goal, we mean either minimum or maximum[5,6].

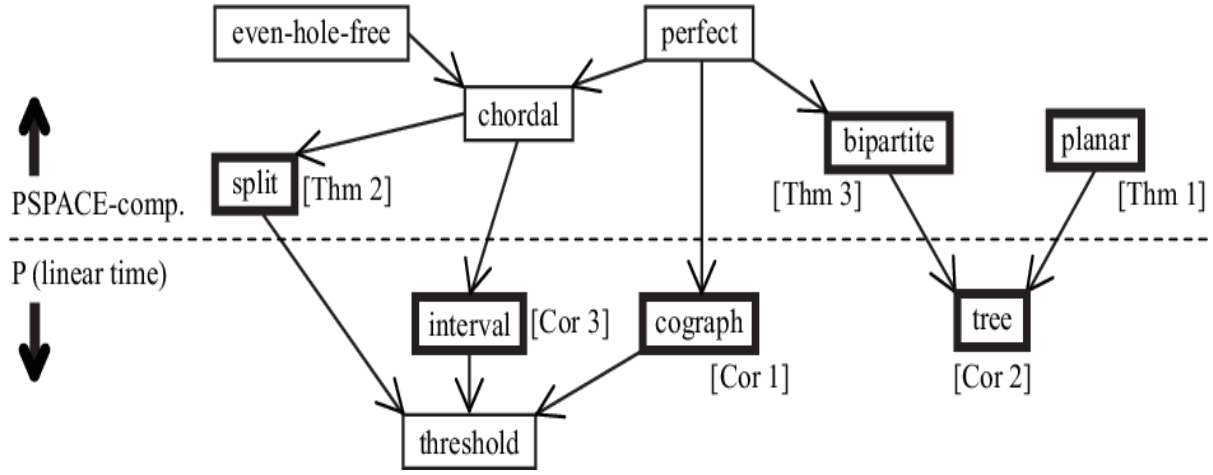

Fig.1: Complexity of Dominating Sets

II. Literature Review

Small set of size in dominating set is called a minimum set and dominating number is known in its size. A dominating set that is not a proper subset of any other dominating set is called a minimal dominating set.

1 Types of Dominating Sets:

1.1 Minimum dominance: for graph G = (V, E), let V 'be the dominant group of graph G, then for any peak u of the graph, either it belongs to group V' or is connected to the peaks in V '.

After removing any elements in V ', V' is no longer the dominant group, then the dominant group V 'is the lower dominant group. The dominant group with the smallest number of peaks in all dominant groups in G is called the dominant group smallest, and the number of peaks in the smallest dominant group is called the dominant number[7].

Minimum coverage of points: for graph G = (V, E), let V 'be the header of the graph G, then for any edge (u, v) in the graph, either belongs to u or belongs to group V' .

After removing any elements in V ', the V' is no longer a crest cap, then V 'is the minimum point cap. The cap with the fewest peaks among all points of coverage in G is called the minimum cap.[14]

Maximum independent group: for graph G = (V, E), let V 'be an independent set of graph G, then for any edge (u, v) in the graph, u and v cannot belong to group V' in Meanwhile, even u neither v nor v belongs to V '.

After adding any element that does not belong to V 'in V', V 'is no longer an independent group, then V' is a very independent group. The independent group that has the largest number of peaks in all the independent groups of peaks in G is called the largest independent group[15].

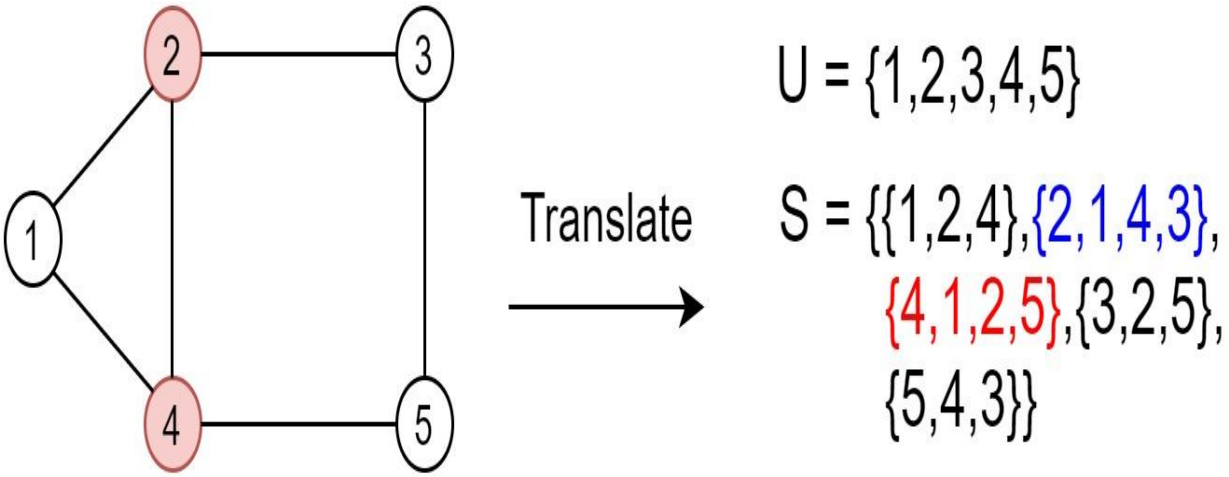

Fig.2: Minimum Dominating set

1.2 Minimal Dominating

Minimal dominating is that number of the data value that is <= to all other values in dataset.

$$S \equiv \{\, i : x(i) = 1 \,\}$$

D1: **if** $(x(i) = 0) \wedge (c(i)$ incorrect$)$
then correct $c(i)$ [rectify neighbor count]

D2: **if** $(|N(i) \cap S| = 0) \wedge (x(i) = 0) \wedge (c(i) = 0) \wedge (\nexists j \in N(i) : j < i, c(j) = 0)$
then $x(i) := 1$ [enter set]

D3: **if** $(|N(i) \cap S| > 0) \wedge (x(i) = 1) \wedge (\forall j \in N(i) - S : c(j) = 2)$
then $x(i) := 0$ and

$$c(i) := \begin{cases} 1, & \text{if } |N(i) \cap S| = 1 \\ 2, & \text{otherwise} \end{cases}$$ [leave set]

Fig.3: Minimal Dominating set Algorithm

S is denoted to sets, D1, D2, D3 denotes to dominating iterations.

2 Greedy Solution:

2.1 Lower Dominant Group: The strategy of greed is to select a point as root first, have the traversal sequence according to the traversal of the first depth, and greed in the reverse sequence order of the resulting sequence. For the point that is not in the dominant group or connected to the point in the dominant group, if the parent node does not belong to the dominant group, add the parent node to the dominant group[8].

The ranking of greed in the greed strategy is very important, according to the first depth traversal process to get the reverse direction of the greed sequence, greed can guarantee every point. Processing of this node will only be processed after its sub-trees have been processed, ensuring greedy health[8].

1 Find the entire tree with first depth point 1 and find the number of each point in the first depth traversal sequence and the parent node number for each point

2 Check according to the reverse sequence of the first sequence - depth, if the current point is not in the dominant group or connected to the point in the dominant group.

And the parent node does not belong to the dominant group, add the parent node to the dominant group, and the number of points in the dominant group plus 1. Tick the current node. Parent node of current node and parent node of current node parent node, because these nodes either belong to the predominant set (parent node of current node), either you connect to the point in the dominant group (the current node and parent node of the current node)[8].

Algorithm 3 Greedy heuristic 1 (HEU_1)

Input: Graph $G = (V, E)$
Output: k-dominating set D_k

1: sort vertices in $V(G)$ in descending order of degree
2: **for** $v \in V(G)$ **do**
3: $isCovered[v] \leftarrow False$
4: **end for**
5: $D_k \leftarrow \emptyset$
6: **for** $v \in V(G)$ **do**
7: **if** $isCovered[v] = False$ **then**
8: $D_k \leftarrow D_k \cup \{v\}$
9: **for** $u \in \mathcal{N}(v, k)$ **do**
10: $isCovered[u] \leftarrow True$
11: **end for**
12: **end if**
13: **end for**
14: **return** D_k

Fig.4: Greedy Solution Algorithm

The above algorithm explained how to inter an input then generating output as a dominating set, firstly it's sorting descending order to degree then it's checking covered of vertices if correct to be dominating at end.

2.2 Cover the Lower Point: The greedy strategy is that if the current point and the parent node of the current point do not belong to the headgear group. Then add the parent node to the headdress group, mark the current node and the parent node of the current knot does not belong to the headdress group, then add the original knot to the headdress set, mark the current knot and the original knot to be covered. Maximum Independent Cluster: The greedy strategy is to add the current node to the independent cluster if the current node is not replaced, and tick both the current node and the parent node to be replaced[9].

```
int p [maxn]; // Parent node number
bool select [maxn]; /
                    / is used for deep first judgment
int newpos [maxn]; /
/ newpos [i] denotes a point ith the first depth sequenc
int now; /
/ indicates that the current first depth sequence conta
int n, m;
void dfs (int x)
```

```
{
  newpos [now ++] = x;
  for (int k = head [x]; k! = -1; k
                = edge [k] .next) {
    if (! select [edge [k] .to]) {
      select [edge [k] .to] = true;
      p [edge [k] .to] = x;
      dfs (edge [k] .to);
    }
  }
}
int greedy ()
{
    bool s [maxn]
= {0}; // If s [i] is true, then s [i] has been replaced.
    bool set [maxn]
= {0}; /
/ set [i] indicates that the point i belongs to the desired
  int ans = 0;
  for (int i = n - 1, i >= 0, i -) {
    int t = newpos [i];
    if (! s [t]) {
      if (! set [p [t]]) {
        set [p [t]] = true;
        ans ++;
      }
      s [t] = true;
      s [p [t]] = true;
      s [p [p [t]]] = true;
    }
  }
  return ans;
}
int greedy ()
{
  bool s [maxn] = {0};
  bool set [maxn] = {0};
  int ans = 0;
    // The root node cannot be verified
  for (int i = n - 1, i >= 1, i -) {
    int t = newpos [i];
    if (! s [t] &&! s [p [t]]) {
      set [p [t]] = true;
      ans ++;
      s [t] = true;
      s [p [t]] = true;
    }
  }
  return ans;
}
int greedy ()
{
  bool s [maxn] = {0};
  bool set [maxn] = {0};
  int ans = 0;
  for (int i = n - 1, i >= 0, i -) {
    int t = newpos [i];
    if (! s [t]) {
      set [t] = true;
      ans ++;
      s [t] = true;
      s [p [t]] = true;
    }
  }
  return ans;
}
int main ()
{  memset (select, 0, sizeof (select));
  now = 0;
  select [1] = true;
  p [1] = 1;
  dfs (1);
  return 0; }
```

III. TREE DP SOLUTION

Basic algorithm is since this is the problem of finding the largest value on the tree, it is clear that dynamic programming can be used in the form of a tree, but the case design is more complex. In order to ensure the correctness of dynamic programming, three nodes are designed for each point, as follows[10]:

I. p dp [i] [0]: shows that point i has a place to the dominating set, and the minimum number of points

in the dominant group is covered when all the sub-tree is covered by point i as root.

II. dp [i] [1]: I do not belong to the dominant group, the sub-trees with i as root are covered, and they are covered by at least one sub-node, which is the minimum number of points in the dominant group.

III. dp [i] [2]: I do not belong to the dominant group, and the sub-trees with i as root are completely covered, and they are not covered by sub-nodes, take points in dominating set as minimum number.

For the first case, dp [i] [0] is equal to the sum of the minimum values for the three states of each son node (whether or not its son is covered) plus 1, that is, as long as each of them has a root of son i the sub-trees are covered, plus point i Current, minimum number of points required, the equation is as follows:

$$dp\,[i]\,[0] = 1 + \Sigma min\,(dp\,[u]\,[0], dp\,[u]\,[1], dp\,[u]\,[2])\,(p\,[u] = i).$$

For the second case, if point i has no dependent nodes, then dp [i] [1] = INF; Otherwise, it is necessary to make sure that each sub-tree has a root in the son of i, each of which is taken the sum of the minimum values for the first two states of the son node, because at this stage I do not belong to the dominant group and I cannot control the nodes belonging to it, the node must be Sub-node is already dominant, regardless of the sub-node's third state. If in the currently selected case, not every son is selected to enter the dominant group (it may appear that every v takes F [v] [1] when combined, which means that we have finally discovered that the sub-node v of u in the solution represented by F [u] [1] To cover other points! This contradicts our definition of F [u] [1].) In the first two states of each son, the first case is not the least needed, then in order to meet the definition of the second case, you need to redefine the state of the son when Point i as a first case. Take the point at the lowest cost, i.e[13]. take the son node u of min (dp [u] [0] -dp [u] [1]) and force it to take its first state, and the other son nodes take the second state, the conversion equation is:

$$If\ (I\ don't\ have\ dependent\ nodes)\ dp\,[i]\,[1] = INF$$

$$else\ dp\,[i]\,[1] = \Sigma min\,(dp\,[u]\,[0], dp\,[u]\,[1]) + inc\ of\ between\ it:$$

$$If\ (\Sigma min\,(dp\,[u]\,[0], dp\,[u]\,[1])\ in\ the\ above\ formula\ co = 0;$$

$$\boldsymbol{else\ inc = min\,(dp\,[u]\,[0] - dp\,[u]\,[1])}$$

As for the third case, it's do not belong to the dominant group, and the sub-trees that have been rooted in i are covered, and are not covered by sub nodes, it means that there are no son points for point i and the point does not belong to the dominant group, then point i for the third case is only related In the second case for his son, the equation[11,12].

$$dp\,[i]\,[2] = \Sigma dp\,[u]\,[1]$$

$$dp\,[i]\,[0] = 1 + \Sigma min\,(dp\,[u]\,[0], dp\,[u]\,[1])\,(p\,[u] = i).$$

$$dp\,[i][1] = \Sigma dp\,[u][0]$$

```
// the lower bound of the dominant group:
void DP(int u, int p)
{   dp[u][2] = 0;
  dp[u][0] = 1;
  bool s = false;
  int sum = 0, inc = INF;
  int k;
  for(k = head[u]; k! = −1; k = edge[k].next)
  {     int to = edge[k].to;
    if(to == p)continue;
    DP(to, u);
    dp[u][0]+= min(dp[to][0], min(dp[u][1], dp[u][2]));
    if(dp[to][0] <= dp[to][1])
    {       sum+= dp[to][0];
      s = true;      }
    else
    {       sum+= dp[to][1];
      inc = min(inc, dp[to][0] − dp[to][1]);      }
```

```
    if(dp[to][1]! = INF&&dp[u][2]! = INF)dp[u][2]+= dp[to][1];      else dp[u][2] = INF;   }
  if(inc == INF&&!s)dp[u][1] = INF;
  else   {     dp[u][1] = sum;
    if(!s)dp[u][1]+= inc;   } }
// cover the lower point:
void DP(int u, int p)
{  dp[u][0] = 1;   dp[u][1] = 0;   int k, to;
  for(k = head[u]; k! = -1; k = edge[k].next)
  {     to = edge[k].to;
    if(to == p)continue;
    DP(to, u);
    dp[u][0]+= min(dp[to][0], dp[to][1]);
    dp[u][1]+= dp[to][0];   } }
// maximum independent group:
void DP(int u, int p)
{  dp[u][0] = 1;   dp[u][1] = 0;   int k, to;
  for(k = head[u]; k! = -1; k = edge[k].next)
  {     to = edge[k].to;
    if(to == p)continue;
    DP(to, u);
    dp[u][0]+= dp[u][1];
    dp[u][1]+= max(dp[to][0], dp[to][1]);   } }
```

IV. Conclusion

Complexity algorithms and Techniques of the theory of computation in the theoretical computer science field help to solve mathematician problems on any models. Two main theories of computation are (Computational theory and computational complexity theory). Then in this paper discussed the complexity algorithms on computation dominating sets and the goal is understanding the descriptive statistics. Also in this paper discussed types of dominating (minimum and minimal dominating sets) with their benefits to easy solving statistics problems. Finally, complex sets have been clearly explained, and some models have been presented in this paper about the understanding and applying minimum and minimal complexity algorithm and their benefits on networks and security.

References

[1] A.A. BertossiDominating sets for split and bipartite graphs Inform. Process. Lett., 19 (1984), pp. 37-40.

[2] Grandoni, F. (2006). A note on the complexity of minimum dominating set. Journal of Discrete Algorithms, 4(2), 209-214.

[3] KCS, B. L., & Manurangsi, P. (2018, April). On the parameterized complexity of approximating dominating set. In Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing, STOC (pp. 1283-1296).

[4] Eisenbrand, F., & Grandoni, F. (2004). On the complexity of fixed parameter clique and dominating set. Theoretical Computer Science, 326(1-3), 57-67.

[5] de Berg, M., Kisfaludi-Bak, S., & Woeginger, G. (2019). The complexity of dominating set in geometric intersection graphs. Theoretical Computer Science, 769, 18-31.

[6] Haddadan, A., Ito, T., Mouawad, A. E., Nishimura, N., Ono, H., Suzuki, A., & Tebbal, Y. (2015, August). The complexity of dominating set reconfiguration. In Workshop on Algorithms and Data Structures (pp. 398-409). Springer, Cham.

[7] Megiddo, N., & Vishkin, U. (1988). On finding a minimum dominating set in a tournament. Theor. Comput. Sci., 61, 307-316.

[8] Ruan, L., Du, H., Jia, X., Wu, W., Li, Y., & Ko, K. I. (2004). A greedy approximation for minimum connected dominating sets. Theoretical Computer Science, 329(1-3), 325-330.

[9] Li, Y., Thai, M. T., Wang, F., Yi, C. W., Wan, P. J., & Du, D. Z. (2005). On greedy construction of connected dominating sets in wireless networks. Wireless Communications and Mobile Computing, 5(8), 927-932.

[10] Guha, S., & Khuller, S. (1999). Improved methods for approximating node weighted Steiner trees and connected dominating sets. Information and computation, 150(1), 57-74.

[11] Fomin, F. V., Kratsch, D., & Woeginger, G. J. (2004, June). Exact (exponential) algorithms for the dominating set problem. In International Workshop on Graph-Theoretic Concepts in Computer Science (pp. 245-256). Springer, Berlin, Heidelberg.

[12] Nacher, J. C., & Akutsu, T. (2016). Minimum dominating set-based methods for analyzing biological networks. Methods, 102, 57-63.

[13] Bourgeois, N., Della Croce, F., Escoffier, B., & Paschos, V. T. (2013). Fast algorithms for min independent dominating set. Discrete Applied Mathematics, 161(4-5), 558-572.

[14] Abdulfattah, Ghassan Marwan, Mohammad Nazir Ahmad, and Renas Rajab Asaad. "A RELIABLE BINARIZATION METHOD FOR OFFLINE SIGNATURE SYSTEM BASED ON UNIQUE SIGNER'S PROFILE." INTERNATIONAL JOURNAL OF INNOVATIVE COMPUTING INFORMATION AND CONTROL 14.2 (2018): 573-586.

[15] Asaad, Renas R. "Güler and Linaro et al Model in an Investigation of the Neuronal Dynamics using noise Comparative Study." Academic Journal of Nawroz University 8.3 (2019): 10-16.

Asaad, Renas Rajab. (2014). An Investigation of the Neuronal Dynamics Under Noisy Rate Functions. Thesis (M.S.), Eastern Mediterranean University, Institute of Graduate Studies and Research, Dept. of Computer Engineering, Famagusta: North Cyprus.

Asaad, R. R., Abdurahman, S. M., & Hani, A. A. (2017). Partial Image Encryption using RC4 Stream Cipher Approach and Embedded in an Image. Academic Journal of Nawroz University, 6(3), 40–45. https://doi.org/10.25007/ajnu.v6n3a76

Rajab Asaad, R., & Masoud Abdulhakim, R. (2021). The Concept of Data Mining and Knowledge Extraction Techniques. Qubahan Academic Journal, 1(2), 17–20. https://doi.org/10.48161/qaj.v1n2a43

Asaad, R. R., Ahmad, H. B., & Ali, R. I. (2020). A Review: Big Data Technologies with Hadoop Distributed Filesystem and Implementing M/R. Academic Journal of Nawroz University, 9(1), 25–33. https://doi.org/10.25007/ajnu.v9n1a530

Asaad, R. R. (2019). Güler and Linaro et al Model in an Investigation of the Neuronal Dynamics using noise Comparative Study. Academic Journal of Nawroz University, 8(3), 10–16. https://doi.org/10.25007/ajnu.v8n3a360

Asaad, R. R. (2021). Penetration Testing: Wireless Network Attacks Method on Kali Linux OS. Academic Journal of Nawroz University, 10(1), 7–12. https://doi.org/10.25007/ajnu.v10n1a998

Almufti, S., Marqas, R., & Asaad, R. (2019). Comparative study between elephant herding optimization (EHO) and U-turning ant colony optimization (U-TACO) in solving symmetric traveling salesman problem (STSP). Journal Of Advanced Computer Science & Technology, 8(2), 32.

Asaad, R. R., & Abdulnabi, N. L. (2018). Using Local Searches Algorithms with Ant Colony Optimization for the Solution of TSP Problems. Academic Journal of Nawroz University, 7(3), 1–6. https://doi.org/10.25007/ajnu.v7n3a193

Almufti, S., Asaad, R., & Salim, B. (2018). Review on elephant herding optimization algorithm performance in solving optimization problems. International Journal of Engineering & Technology, 7, 6109-6114.

Asaad, R. R., & Ali, R. I. (2019). Back Propagation Neural Network(BPNN) and Sigmoid Activation Function in Multi-

Layer Networks. Academic Journal of Nawroz University, 8(4), 216–221. https://doi.org/10.25007/ajnu.v8n4a464

Rajab Asaad, R. (2021). Review on Deep Learning and Neural Network Implementation for Emotions Recognition . Qubahan Academic Journal, 1(1), 1–4. https://doi.org/10.48161/qaj.v1n1a25

Asaad, R. R., Abdulrahman, S. M., & Hani, A. A. (2017). Advanced Encryption Standard Enhancement with Output Feedback Block Mode Operation. Academic Journal of Nawroz University, 6(3), 1–10. https://doi.org/10.25007/ajnu.v6n3a70

Abdulfattah, G. M., Ahmad, M. N., & Asaad, R. R. (2018). A reliable binarization method for offline signature system based on unique signer's profile. INTERNATIONAL JOURNAL OF INNOVATIVE COMPUTING INFORMATION AND CONTROL, 14(2), 573-586.

Almufti, S. M., Ahmad, H. B., Marqas, R. B., & Asaad, R. R. (2021). Grey wolf optimizer: Overview, modifications and applications. International Research Journal of Science, Technology, Education,and Management, 1(1),1-1.

Asaad, R. R., Sulaiman, Z. A., & Abdulmajeed, S. S. (2019). Proposed System for Education Augmented Reality Self English Learning. Academic Journal of Nawroz University, 8(3), 27–32. https://doi.org/10.25007/ajnu.v8n3a366

Asaad, R. R. (2020). Implementation of a Virus with Treatment and Protection Methods. ICONTECH INTERNATIONAL JOURNAL, 4(2), 28-34. https://doi.org/10.46291/ICONTECHvol4iss2pp28-34

Boya Marqas, R., M. Almufti, S., & Rajab Asaad, R. (2022). FIREBASE EFFICIENCY IN CSV DATA EXCHANGE THROUGH PHP-BASED WEBSITES. Academic Journal of Nawroz University, 11(3), 410–414. https://doi.org/10.25007/ajnu.v11n3a1480

Ihsan, R. R., Almufti, S. M., Ormani, B. M., Asaad, R. R., & Marqas, R. B. (2021). A survey on Cat Swarm Optimization algorithm. Asian J. Res. Comput. Sci, 10, 22-32.

Rajab Asaad, R., & Luqman Abdulnabi, N. (2022). A Review on Big Data Analytics between Security and Privacy Issue. Academic Journal of Nawroz University, 11(3), 178–184. https://doi.org/10.25007/ajnu.v11n3a1446

Yahya Hussien , A., & Rajab Asaad, R. (2022). Review on Social Media and Digital Security. Qubahan Academic Journal, 2(2), 1–4. https://doi.org/10.48161/qaj.v2n2a119

Asaad, R. R. (2022). Keras Deep Learning for Pupil Detection Method . Academic Journal of Nawroz University, 10(4), 240–250. https://doi.org/10.25007/ajnu.v10n4a1328

Asaad, R. R., & Segerey, R. I. (2018). School Management Application Using iOS. Academic Journal of Nawroz University, 7(4), 38–44. https://doi.org/10.25007/ajnu.v7n4a269

Asaad, R. R., Mustafa, R. F., & Hussien, S. I. (2020). Mortality Statistics and Cause of Death at Duhok City from The Period (2014-2019) Using R Language Data Analytics. Academic Journal of Nawroz University, 9(3), 1–7. https://doi.org/10.25007/ajnu.v9n3a699

Asaad, R. R. (2021). A Study on Instruction Formats on Computer Organization and Architecture. ICONTECH INTERNATIONAL JOURNAL, 5(2), 18-24. https://doi.org/10.46291/ICONTECHvol5iss2pp18-24

Asaad, R. R. (2021). Virtual reality and augmented reality technologies: A closer look. Virtual reality, 1(2).

Asaad, R. R. A Review: Emotion Detection and Recognition with Implementation on Deep Learning/Neural Network.

Asaad, R. R., Saeed, V. A., & Abdulhakim, R. M. (2021). Smart Agent and it's effect on Artificial Intelligence: A Review Study. ICONTECH INTERNATIONAL JOURNAL, 5(4), 1-9.

Asaad, R. R. A Asaad, R. R. A Review: Emotion Detection and Recognition with Implementation on Deep Learning/Neural Network.