WWW.IJRPR.COM

International Journal of Research Publication and Reviews

Journal homepage: www.ijrpr.com ISSN 2582-7421

A Step-by-Step Mathematical Derivation of Ant Colony Optimization for Solving Subset Selection Problems

Akshaya Kumar Mandal

Department of Computer Science, Assam University, A Central University of India Assam, Silchar- 788011, Assam, India
Email: akshayacs207@gmail.com
DOI: https://doi.org/10.55248/gengpi.4.1223.123541

A B S T R A C T

This article presents a thorough, step-by-step mathematical derivation of Ant Colony Optimization (ACO) tailored specifically for solving Subset Selection Problems. ACO, inspired by the foraging behavior of ants, has proven its efficacy as a metaheuristic for combinatorial optimization, and this derivation focuses on its application to subset selection scenarios. Beginning with an exploration of the foundational principles of ACO, the derivation delves into the adaptation of these principles to address subset selection challenges, emphasizing the formulation of the objective function, pheromone updating mechanisms, and solution construction procedures. The mathematical rigor is complemented by intuitive explanations, bridging the theoretical and practical aspects of ACO. Additionally, the article highlights the algorithm's versatility in handling diverse subset selection objectives, showcasing its adaptability to various problem domains. In essence, this comprehensive derivation provides readers with a profound understanding of ACO's inner workings, enabling them to apply and customize the algorithm effectively for subset selection problems in different contexts.

Keywords: Ant Colony Optimization, Subset Selection Problems, Combinatorial Optimization Problem, Mathematical Derivation, Optimization Algorithms.

1. Introduction

Subset selection problems are completely different from ordering problems. Out of a set S of n objects we have to select the best subset of' n' objects, it satisfying some more constraints. In subset selection problems are finding an optimal feasible subset of an initial set of objects with respect to an objective function or some constraints [4-13]. Subset selection is a heuristic search process where search space contains states, each of which generate a candidate subset for evaluation. Two things must be determined for subset generation, Search starting point and Search strategy [12]. In a subset selection the starting point is chosen randomly without any consideration and objects are added or deleted as per the requirement. The subset selection problem is one of NP-hard combinatorial problems can be formulated subset, which is easiest to describe and understand. Available algorithms to solve this problem needs an exponential time, thus finding a solution to this problem is not currently feasible but verifiable in polynomial time.

1.1 Subset Selection Problems

Subset Selection problems involve finding an optimal feasible subset of objects within an initial set of objects. Generally a subset selection problem may be defined by a triple (S, S_{Fitness} , f) where

- S is a set of objects.
- $S_{\text{Fitness}} \subseteq$ P(S) is a set that contains all feasible subsets of S.
- f: $S_{\text{Fitness}} \rightarrow$ GR is an objective function that associates a real-valued cost f(S^*) with every feasible subsets of objects $S' \in S_{\text{Fitness}}$.

The goal of subset selection problem (S, S_{Fitness} , f) is to find $S^* \in S_{\text{Fitness}}$ and f(S^*) is maximal. Here a few examples of subset selection problems.

1.2 Subset Selection process

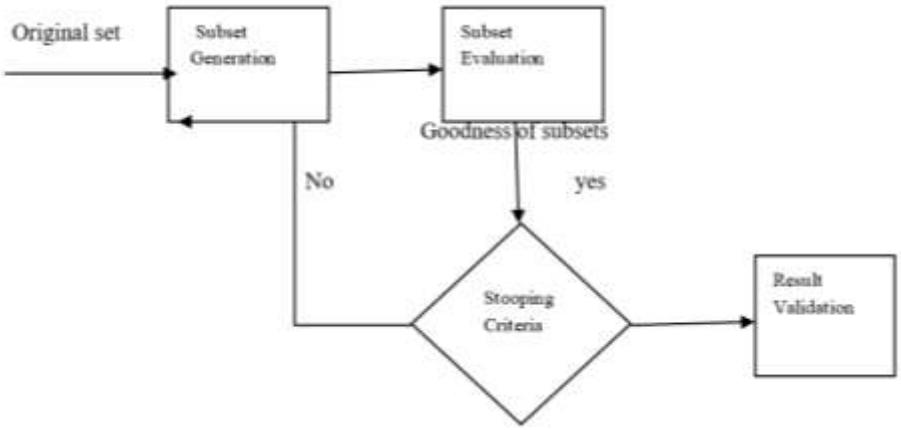

Figure 1 Subset Selection Processes [10]

Minimum vertex covers problems (MVC): The goal is to find a subset of vertices such that for each edge in E, at least one of its two end vertices is in the subset and that its cardinality is minimum[10]. For a non-empty vertex set S ⊆ V, if each edge in E has at least one endpoint in S, then S is called a vertex cover set of graph G. The vertex cover with the minimum number is the minimum vertex cover set of graph G. For any graph G (V,E) and S ⊆ V ,the following statements are equivalent:

- S is the minimum vertex cover in G.
- V-S is the maximum independent vertex of G.

A mathematical model of Minimum Vertex Cover (MVC) problem is presented in chapter 6.

Travelling Salesman Problem (TSP): The travelling salesman problem (TSP) is the problem of finding the shortest tour among a set of 'n' cities starting from one city and journey all other cities once and only once before returning to the home (starting) city. The goal is to find the smallest subset of arc($E' \subseteq$ E) of the graph G, i.e. best path in term of minimum distance (or the minimum cost) travel by salesman. Graphically, it can be represented by a complete weighted graph G (V, E), where V is the set of vertices representing cities, and E is the set of edges representing paths fully connecting all cities [1-3]. Each edge (i,j) ∈ E is associated with a cost d_{ij} ,which is the distance between cities i and j. Different examples of the TSP are also split into different classes based on the distance between the cities or the type of graph.

2. Subset-Sum Problems

The Subset-Sum problem (SSP) is defined as follows: given a set of positive integers S, and a positive integer C. This problem is to find one/all subsets of S that sum as close as possible to, but do not exceed C [3, 12]. For example, consider the set S = {1, 2, 3, 4, 5, 6} and let the target sum C be 12. The total number of subsets of S in this case is 64. Some of the valid solutions to this problem are the sets, {1,2,3,6}, {1,5,6}, {2,4,6},{3,4,5}.In general, we notice that the total number of subsets taken from a set of n elements is 2^n. An algorithm that tests all of these possible solution subsets needs an exponential time. Here a few examples of subset sum problems:

Knapsack Problem: The subset sum problem is a special case of 0/1 Knapsack problem consists of loading objects in to a knapsack load capacity. Each object can be loaded or not loaded into the knapsack, relating 0-1 decision about object loading. The 0/1 knapsack problem does not allow the user to put multiple copies of the same items in their knapsack. The goal is to find a subset of objects that maximizes a total profit [5, 14-17].

2. Graph Representation of SSP

Let G=(V,E) denoted the graph for Subset Selection problem and the solution to this instance is an unordered vertex subset S ⊆ V .Each object of the set represent vertices in the graph and selected object are represented as a combination of arcs where ants have traversed through the graph. We construct a complete graph G_c= (V,E_c) be the complete graph of G_c . By using Ant Colony Algorithm, the ants would choose the next node arbitrarily. The connected function for each edge in graph G is defined as follows: C: $E_c \rightarrow \{0,1\}$ for each edge (i, j) as [10]:

$$C(i,j) = \begin{cases} 1, & (i,j) \in E \\ 0, & (i,j) \notin E_c - E \end{cases}$$

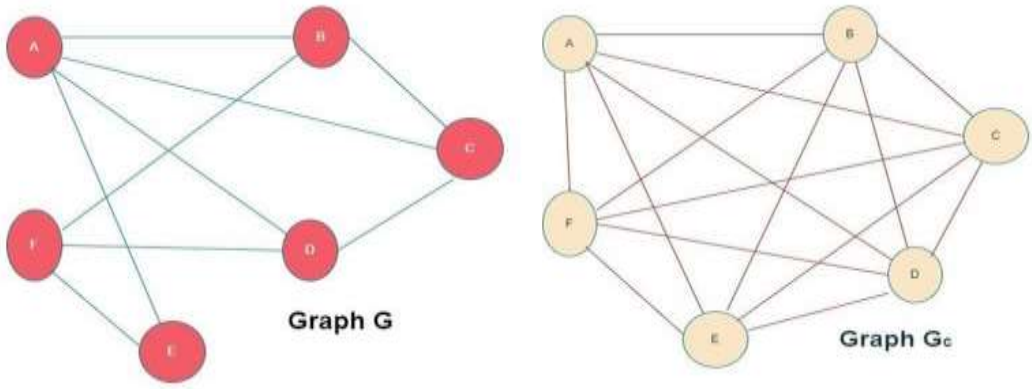

Figure 2 Graph Representation of SSP with 6-Vertices

By defining the connected value for each edge, we can distinguish the edges between E(G) and

$E(G_c)$-E (G).Meanwhile, the connected value is also the pheromone. For ant k , let any vertex be the initial point. In order to ensure that ant k does not choose the vertices which have been reached, we update the connected value of edge (i, j) when the ant k reaches the point i . C (i,j)=0 ,(j∈ H) (Select H vertices arbitrarily as the initial points and place ant on it.). Then calculating the total value for the edges incident with each vertex in graph G. The value of an edge, connectivity value, in E is updated when one of its edge vertices [90] was visited by some ant, say ant k, in the following way: C(i, j)= $\frac{1}{n}$, if edge (i, j) ∈ E and either vertex i or j is visited by ant k , in which n is the number of graph vertices,| V| . First, at the end of each cycle, the pheromone left on the vertices of the currently best solution. Suppose that S is the currently best solution. For each vertex i ∈S we will update its pheromone according to global updating by: $\tau_i = (1-\rho)\tau_i$

Where $\Delta\tau_i=\frac{1}{|s_k|}$ and ρ ∈ (0,1) is a parameter which simulates the evaporation rate of the pheromone intensity. So the updated pheromone is: $\tau_i=\tau_i+\rho\Delta\tau_i$

2.1 ACO for the solution of Subset Selection Problem

The ACO literature typically development of a construction graph for the problem to be solved as necessary to the application of ACO as it is a graph based on shortest path problem that real ants solve when traveling from the nest to a food source.

According to G.Leguizamon and Z.Michalewicz [6], there is no real concept of path in subset problems. According to Dorigo et al.[1-3], a construction graph can be representing solution components as vertices on which pheromone is associated with the objects and these problems in terms of a graph is quite artificial. According to Lee et al.[18] where they adopted the graph based ant system to solve feature selection where candidate solution can be represented in directed graphs. Finally, Jensen and Shen [19] represented feature selection as graph where the nodes represent features and the edges between the nodes denote the choice of the next feature. The search for the optimal feature subset is by an ant traversal through the graph where a minimum number of nodes are visited that satisfies the traversal stopping criterion .According Aghdam et al.[20] follow similar method like Jensen and Shen but the pheromone is associated with the vertices of graph instead of edges of the graph. The vertices represent features and the edges between them indicate the choice of the next feature. Each ant starts with a random feature, from these initial positions, they visits edges probabilistically until a traversal stopping criterion is satisfied.

The previous papers show the need to an alternative problem representation and which follows the general characteristics of ACO meta-heuristic. A problem may be modeled as subset problem or a kind of ordering problems. Each objects of the set represents as the node of the graph and the selected objects are represented as a combination of arcs where ants have traversed through the graph. Every ant must visits every node in the graph at most once.

2.3 ACO algorithm for SSP

Step-1:Initialization parameters: by using formula as follows to give the connected value for each edge in graphG_c.

$$C(i,j)=\begin{cases}1, & (i,j)\in E\\ 0, & (i,j)\notin E_c-E\end{cases}$$

and $S_k=\varphi$,i=1,k=1, Select H vertices arbitrarily as the initial points and place ant on it.

Step-2:Use function C(i,j)=0 to update the connected values of all edges incident with vertex i

and compute $C_j^k = \frac{1}{n}$ (j∈ V). If $C_j^k = 0$, we get the vertex cover set S_k . Else if k<H, let

k = k +1 , i = i +1, then go to Step 1. else if k=H, go to Step 4. else go to Step 3.

Step-3:Compute C_j^k the vertex has the maximum value then $S_k = S_k \cup \{u_i\}$.and

Obtain subset from set S by $S_k=\{S_k,\{u_i\},\{S_k \cup \{u_i\}\}$, go to Step-2.

Step-4 : Print all subsets of S is S_k={∅ ∪ S_1 ∪ ... ...∪ S_H },then stops algorithm .

3. Numerical Evaluation of ACO in SSP

We use graph G in Figure 1 to illustrate the specific implementation of the algorithm. And select the subset of vertex cover of graph G would be obtained.

3.1 Initialization of Parameters

Assume that $\alpha = 1, \beta = 1, \rho = 0$,m=6 (i.e number of ants) and

$$\tau_{ij} = \begin{cases} 1, & (i,j) \in E \\ 0, & (i,j) \notin E \end{cases}$$

3.2 A Step by Step Illustration

Step-1 (ant-1)(iteration-1)

Let vertex A be the initial point used by ant 1 to search vertex cover set. Give the connected value for each edge in graph G , we have

C(A,B)=C(A,C)=C(A,D)=C(A,E)=C(B,C)=C(B,F)=C(C,D)=C(D,F)=C(E,F)=1 and other edges have the value zero. Let S_1={A}.

Step-2: Let the connected value of all edges incident with vertex A be 0 , and then compute C_i^1 ,

C_B^1=C_C^1=C_D^1=$\frac{1}{6}$+1+1=2.18, C_F^1=3

Step-3: C_F^1=3 is the maximum then S_1={A,F},goto step 2.

Step-2:Let the connected value of all edges incident with vertex F be 0 , and then compute C_i^1 ,

C_B^1=C_D^1=$\frac{1}{5}$+1=1.02, C_E^1 =$\frac{1}{5} = 0.2$, C_C^1= 1+1=2 goto step 3

Step-3: C_c^1=2 is the maximum then S_1={A,F,C},goto step 2.

Step-4: Update the connected values of all edges, and C_A^1=C_B^1=C_c^1=C_D^1=C_E^1=C_F^1=0

So the algorithm stops. Output is $S_1 = \{\emptyset, \{A\}, \{A, F\}, \{A, F, C\}\}$.

Iteration-2

Step-1 (ant-2)

Let vertex B be the initial point used by ant 2 to search vertex cover set. Give the connected value for each edge in graph c G , we have

C(B,A)=C(A,C)=C(A,D)=C(A,E)=C(B,C)=C(B,F)=C(C,D)=C(D,F)=C(E,F)=1 and other edges have the value zero. Let S_2={B}.

Step-2: Let the connected value of all edges incident with vertex B be 0 , and then compute C_i^2 ,

C_A^2= $\frac{1}{6} + 1 + 1 + 1 = 3.16$, C_C^2=C_F^2=$\frac{1}{6}$+1+1=2.18, C_E^1=1+1=2, C_E^1=1+1+1=3,

Step-3: C_A^2=3.16 is the maximum then S_2={B,A},goto step 2.

Step-2: Let the connected value of all edges incident with vertex A be 0 , and then compute C_i^2 ,

C_E^1=C_C^1=$\frac{1}{5}$+1=1.02, C_D^1 =1+1+1 = 3, C_F^1= 1+1=2 goto step 3

Step-3: C_D^2=3 is the maximum then S_2={B,A,D},goto step 2.

Step-2: Let the connected value of all edges incident with vertex D be 0 , and then compute C_i^2 ,

C_E^1= 1 , C_C^1=0, , C_F^1= 1+$\frac{1}{4}$=1.25 goto step 3

Step-3: C_F^2=1.25 is the maximum then S_2={B,A,D,F},goto step 2.

Step-4: Update the connected values of all edges, and C_A^2=C_B^2=C_c^2=C_D^2=C_E^2=C_F^2=0

So the algorithm stops. Output is $S_2 = \{\emptyset, \{B\}, \{B,A\}, \{B,A,D\}, \{B,A,D,F\}\}$.

Iteration-3

Step-1 (ant-3)

Let vertex C be the initial point used by ant 3 to search vertex cover set. Give the connected value for each edge in graph c G , we have

C(A,B)=C(A,C)=C(A,D)=C(A,E)=C(B,C)=C(B,F)=C(C,D)=C(D,F)=C(E,F)=1 and other edges have the value zero .Let S_3={C}.

Step-2:Let the connected value of all edges incident with vertex C be 0 , and then compute C_i^3 ,

$C_A^3=\frac{1}{6}$+1+1+1=3.16, $C_B^3=C_D^1=\frac{1}{6}$+1+1=2.18, C_F^3=3, C_E^3=2

Step-3: C_A^3=3.16 is the maximum then S_3={C,A},goto step 2.

Step-2: Let the connected value of all edges incident with vertex A be 0 , and then compute C_i^3 ,

$C_B^3=C_D^3=C_E^3=\frac{1}{5}$+1=1.02, C_F^3= 1+1+1=3 goto step 3

Step-3: C_F^3=3 is the maximum then S_3={C,A,F},goto step 2.

Step-4: Update the connected values of all edges, and $C_A^3=C_B^3=C_c^3=C_D^3=C_E^3=C_F^3=0$

So the algorithm stops. Output is S3 = {∅ ,{C},{C,A},{C,A,F}}.

Iteration-4

Step-1 (ant-4)

Let vertex D be the initial point used by ant 4 to search vertex cover set. Give the connected value for each edge in graph G , we have

C(B,A)=C(A,C)=C(A,D)=C(A,E)=C(B,C)=C(B,F)=C(C,D)=C(D,F)=C(E,F)=1 and other edges have the value zero. Let S_4={D}.

Step-2: Let the connected value of all edges incident with vertex B be 0 , and then compute C_i^2 ,

$C_A^4=\frac{1}{6}+1+1+1=3.16$, $C_C^4=C_F^4=\frac{1}{6}$+1+1=2.18, C_E^4=1+1=2, C_F^4=1+1+1=3,

Step-3: C_A^4=3.16 is the maximum then S_4={D,A},goto step 2.

Step-2: Let the connected value of all edges incident with vertex D be 0 , and then compute C_i^4 , $C_E^4=C_C^4=\frac{1}{5}$+1=1.02, $C_B^4=\frac{1}{5}+1+1=2.2$, C_F^4= 1+1=2 goto step 3

Step-3 : C_B^4=2.2 is the maximum then S_4={D,A,B},goto step 2.

Step-2: Let the connected value of all edges incident with vertex B be 0 , and then compute C_i^4 ,

C_E^4= 1 , $C_C^1=\frac{1}{4}$, , C_F^1= 1+$\frac{1}{4}$=1.25 goto step 3

Step-3: C_B^4=1.25 is the maximum then S_4={D,A,B,F},goto step 2.

Step-4: Update the connected values of all edges: $C_A^4=C_B^4=C_c^4=C_D^4=C_E^4=C_F^4=0$

So the algorithm stops. Output is S4 = { ∅,{D},[D,A},{D,A,B},{D,A,B,F}}.

Iteration-5

Step-1 (ant-5)

Let vertex E be the initial point used by ant 4 to search vertex cover set. Give the connected value for each edge in graph G , we have

C(B,A)=C(A,C)=C(A,D)=C(A,E)=C(B,C)=C(B,F)=C(C,D)=C(D,F)=C(E,F)=1 and other edges have the value zero. Let S_5={E}.

Step-2: Let the connected value of all edges incident with vertex B be 0 , and then compute C_i^2 ,

$C_A^5=\frac{1}{6}+1+1+1=3.16$, $C_B^5=C_C^5=C_D^5$=1+1+1=3, $C_F^5=\frac{1}{6}$+1+1=2.16,

Step-3: C_A^5=3.16 is the maximum then S_5={E,A},goto step 2.

Step-2: Let the connected value of all edges incident with vertex D be 0 , and then compute C_i^5 ,

$C_B^5=C_C^5=C_D^5=\frac{1}{5}$+1+1 = 2.2, C_F^4= 1+1=2 goto step 3

Step-3: C_B^5=2.2 is the maximum then S_5={E,A,B},goto step 2.

Step-2 : Let the connected value of all edges incident with vertex B be 0 , and then compute C_i^5 ,

C_D^5= 1 + 1 = 2 , C_C^5= C_F^5= 1+$\frac{1}{4}$=1.25 goto step 3

Step-3: C_D^5=2 is the maximum then S_5={E,A,B,D},goto step 2.

Step-4 : Update the connected values of all edges, and $C_A^5=C_B^5=C_c^5=C_D^5=C_E^5=C_F^5=0$

So the algorithm stops. Output is S_5 = { ∅,{E},{E,A},{E,A,B},{E,A,B,D}}.

Iteration-6

Step-1 (ant-6)

Let vertex F be the initial point used by ant 1 to search vertex cover set. Give the connected value for each edge in graph G , we have

C(A,B)=C(A,C)=C(A,D)=C(A,E)=C(B,C)=C(B,F)=C(C,D)=C(D,F)=C(E,F)=1 and other edges have the value zero. Let S_6={F}.

Step-2: Let the connected value of all edges incident with vertex F be 0 , and then compute C_i^6 ,

C_B^6=C_D^6=$\frac{1}{6}$+1+1=2.18, C_A^6=4, C_C^6=1+1+1=3

Step-3: C_A^6=4 is the maximum then S_6={F,A},goto step 2.

Step-2: Let the connected value of all edges incident with vertex A be 0 , and then compute C_i^6 ,

C_B^1=C_D^1=$\frac{1}{5}$+1=1.02, C_E^1 =$\frac{1}{5}$ = 0.2, C_C^1=$\frac{1}{5}$ + 1+1=2.2 goto step 3

Step-3: C_c^1=2.2 is the maximum then S_6={F,A,C},goto step 2.

Step-4 Update the connected values of all edges, and C_A^6=C_B^6=C_c^6=C_D^6=C_E^6=C_F^6=0

So the algorithm stops. Output is S_6 = { ∅,{F},{F,A},{F,A,C}}.

Table 4.1 ACO-SSP Computation results

Ant	Initial point	Subsets
1	A	S_1={∅,{A},{A,F},{A,F,C}}
2	B	S_2={∅, {B},{B,A},{B,A,D},{B,A,D,F}}
3	C	S_3={∅ ,{C},{C,A},{C,A,F}}
4	D	S_4={∅,{D},{D,A},{D,A,B},{D,A,B,F}}
5	E	S_5={∅,{E},{E,A},{E,A,B},{E,A,B,D}}
6	F	S_6 = { ∅,{F},{F,A},{F,A,C}}

So the Subset is

S_k={∅,{A},{B},{C},{D},{E},{F},{B,A},{C,A},{A,F},{D,A},{E,A},{F,A},{A,F,C},{C,A,F},

{B,A,D},{E,A,B},{B,A,D,F},{D,A,B,F},{E,A,B,D}}.

4. Conclusion

In conclusion, the step-by-step mathematical derivation of Ant Colony Optimization (ACO) for solving Subset Selection Problems has provided valuable insights into the algorithm's effectiveness in tackling combinatorial optimization challenges. The presented application of ACO to subset selection, with ants navigating through the selection space, has yielded a final sequence S_k that represents optimal subsets based on the algorithm's decision-making process. The subsets in S_k reflect the chosen combinations of elements, showcasing the algorithm's ability to adapt and converge towards solutions that satisfy the subset selection criteria. This study not only reinforces the robustness of ACO as a metaheuristic but also highlights its versatility in handling diverse problem domains. Future research can further explore parameter tuning, scalability, and the algorithm's performance on different subset selection scenarios, contributing to the continued enhancement of ACO's applicability in real-world optimization problems. Overall, the presented derivation and results contribute to the understanding and application of Ant Colony Optimization in solving subset selection problems, offering a foundation for further exploration and refinement in optimization methodologies.

Reference

[1] A. K. Mandal & S. Dehuri. A survey on ant colony optimization for solving some of the selected np-hard problem. In International Conference on Biologically Inspired Techniques in Many-Criteria Decision Making, Springer, Cham, 85--100, (2020).

[2] A. K. Mandal, P.K.D. Sarma, S. Dehuri. A Study of Bio-inspired Computing in Bioinformatics: A State-of-the-art Literature Survey. The Open Bioinformatics Journal, 2023 Jun 23; 16(1). https://doi.org/10.2174/18750362-v16-e230517-2022-17

[3] A. K. Mandal, P.K.D. Sarma. Novel Applications of Ant Colony Optimization with the Traveling Salesman Problem in DNA Sequence Optimization. In2022 IEEE 2nd International Symposium on Sustainable Energy, Signal Processing and Cyber Security (iSSSC) 2022 Dec 15 (pp. 1-6). IEEE. https://doi.org/10.1109/iSSSC56467.2022.10051206

[4] S. Martello. P. Toth Worst-Case,"' Analysis of Greedy Algorithms for the Subset Sum Problem. Mathematical Programming", 1984.

[5] H.Oscar ,C. Ibarra, E. Kim, "Fast Approximation Algorithms for the Knapsack and Sum of Subset Problems", JACM, 1975.

[6] G. Leguizam´on, Z. Michalewicz, "Ant Systems for subset problems. Unpublished manuscript", 2000.

[7] A.Naseer, W.Shahzad, A.Ellahi," A Hybrid approach for Feature Subset Selection using Ant Colony Optimization and Multi-Classifier Ensemble", IJACSA, Vol.9, PP. 306-313, 2018.

[8] S.Fidanova, K.Atanassov, P.Marinov," Start Strategies of ACO applied on Subset Problems", LNCS 6046,PP.248-255,2011.

[9] N.Abd-Alsabour, "Binary Ant Colony Optimization for Subset Problems", S.Dehuri et. al(eds.),Multi-objective Swarm Intelligence, Studies in Computational Intelligence 592,@Springer-Verlag Berlin Heidelberg, 2015.

[10] H.A.Isa, S.Oqelli, S.Bani-Ahmad, "The Subset-Sum Problem: Revisited with an Improved Approximated Solution", IJCA, Vol. 114-No.14, 2015.

[11] B.Crawford, C.Carlos, E.Monfroy, "An Ant-based Solver for Subset problems", ICACCT , PP.268-270, 2009.

[12] H.Peng, C.Ying, S.Tan, Z.Sun, "An Improved Feature Selection Algorithm Based on Ant Colony Optimization", IEEE 2018.

[13] S.Sharma, K.M.Buddhiraju, "A Novel Ant Colony Optimization based Training Subset Selection Algorithm for Hyper spectral Image Classification", IEEE 2018.

[14] G.Leguizamon, Z.Michalewicz, "A New Version of Ant System for Subset Problems", 0/1 knapsack, Proceeding of CEC, 1999.

[15] JE. Diaz, J. Handl, Xu D-L "Integrating meta-heuristics, simulation and exact techniques for Production planning of a failure-prone 0/1 knapsack", European Journal of Oper.Res. Vol.266,No.3, pp.976-989, 2018.

[16] S.Samanta, S.Chakraborty, S.Acharjee, A.Mukherjee, N.Dey, " Solving 0/1 Knapsack problem using any Weight Lifting Algorithm", IEEE , 2013.

[17] S.Fidanova, K.Atanassov, P.Marinov, R.Parvathi, "Ant Colony Optimization for Multiple Knapsacks Problem with Controlled Starts", BIO-Automation, 13(4), PP.271-280, 2009.

[18] K. Lee, J. Joo, J. Yang, V. Honavar, " Experimental Comparison Of Feature Subset Selection Using Ga And Aco Algorithm", LNAI 4093, pp. 465–472, @ Springer- Verlag Berlin Heidelberg 2006.

[19] R. Jensen, Q. Shen , "Webpage Classification with ACO-Enhanced Fuzzy-Rough Feature Selection ", S. Greco et al. (Eds.): RSCTC 2006, LNAI 4259, pp. 147–156, @Springer-Verlag Berlin Heidelberg 2006.

[20] S. Nemati, M. E. Basiri, N. Ghasem-Aghaee, M. H. Aghdam, "A novel ACO–GA hybrid algorithm for feature selection in protein function prediction ", S. Nemati et al. / Expert Systems with Applications 36, 12086–12094, 2009