Simulated Annealing Enhanced Weighted Ant Colony Optimization for Feature Selection

YuCheng Liu  
Student MemberWaterloo, Canada  
y698liiu@edu.uwaterloo.ca

Tian Zhu  
Student MemberWaterloo, Canada  
t27zhu@edu.uwaterloo.ca Cong Hao He  
Student MemberWaterloo, Canada  
chhe@edu.uwaterloo.ca

Yiran Tian  
Student MemberWaterloo, Canada  
y62tian@edu.uwaterloo.ca

***Abstract*—This paper presents an enhanced feature selection algorithm designed to optimize the number of features required for data science discoveries algorithm such as classification or clustering. The algorithm is of the type embedded. It improves on Ant Colony Optimization (ACO) by applying Simulated Annealing to the parameters of ACO. Comparison studies have demonstrated that the improved algorithm performs more optimally than most of the existing state-of-art algorithms available, e.g. Genetic Algorithms, Tabu Search. A performance evaluation of the improved algorithm and existing ACO algorithm implementation is also presented. The recommendation is to use the improved algorithm for feature selection problems that weighs more on accuracy and weighs less on time sensitivity.**

***Keywords— feature selection, ant colony optimization***

# Introduction

To solve a specific problem in data science, models should be built accordingly. However, the selection of features for modeling is a non-trivial optimization problem. Feature selection is the process of retrieving a subset of features that are most relevant to the project and prepare the data for the further process. The most important reason to use feature selection technique is that some parts of data can be redundant or irrelevant. Therefore, evaluating the importance and relevance of each feature can enhance the model’s generalization by reducing overfitting, and remove the extra features without incurring much loss of information [1]. Feature selection thus becomes an essential component in machine learning and will skim the workload drastically once the selection was done effectively. The problem was solved in this paper using an enhanced Ant Colony Optimization algorithm by applying Simulated Annealing to the ACO parameters. Here, existing state-of-art solutions to the feature selection problem and their advantages and disadvantages are being discussed in the following sections in addition to other potential solution explored. Performance evaluations of the final solution and recommendations are also provided.

# Literature Review

There are mainly three methods to solve the problem: the filter method, the wrapper method, and the embedded method respectively. Filter method evaluates each variable by finding the correlations with the target variable, evaluating the information value (the ability of single variables to predict target). In the end, each feature gets an assigned score, and the features are ranked by the score and either selected to be kept or removed from the dataset. Wrapper method considers feature selection as a search problem and tends to find the best combination of variables instead of best values as they are measured jointly [2]. A predictive model is used to evaluate a combination of features and assign a score based on model accuracy. Embedded methods learn which features best contribute to the accuracy of the model while the model is being created. The most common type of embedded feature selection methods is regularization methods. There are two categories of algorithms, supervised and unsupervised. Supervised methods select features with a maximum representative and discriminant powers, while unsupervised methods use the inter-feature relations to determine the relevancy of features. For example, multivariate filter method is unsupervised, univariate filter method, the wrapper method, and embedded method are usually supervised [3].

The proposed solution is of type unsupervised embedded method that uses simulated annealing on top of ant colony optimization. The simulated annealing is used to tune the parameters of the ant colony optimization. The solution sacrifices performance in time for the exchange of better accuracy.

# Problem formulation and proposed solution

In recent times, it became increasingly desirable to be able to generalize the application of feature selection regardless of domain knowledge. Therefore, feature selection using unsupervised learning is a better approach as it is independent of learning algorithms and the domain knowledge. Out of the known approaches to feature selection with unsupervised learning, the ACO (ant colony optimization) is the most appealing state-of-art approach.

The main difference between the algorithm used in this work with the ACO approach is that the initial set of features are selected at random. This is the attempt to escape from the local maxima and discover the global maximum for the final solution. The content below discusses the different algorithms approach in solving this optimization problem and that of the enhanced ACO approach.

## Problem Formulation

The feature selection problem can be projected and mapped into an n-dimensional vector space where n represents the total number of features available. For an arbitrary set of features chosen, features that are included will assume a value of one in its position in the vector, and zero otherwise. After the project, the problem becomes an optimization problem of choosing the optimal combination of zeroes and ones to produce the most accurate model. With the idea in mind, the complete problem formulation is stated below.

* The goal of the problem is to select a set of features that produce the most accurate model. In other words, find the n-dimensional vector that can produce the model that has the highest accuracy for designated classifiers.
* The state space of the problem is simply an n-dimensional vector that contains only zeroes and ones. There are 2n possible states in total. For example, if n = 2, one possible state can be [0,1] indicating the inclusion of the second feature and omission of the first feature.
* The initial state can be any n-dimensional vector with values of 0 or 1. For example, if n = 3, one possible initial state is [0,1,0], indicating the inclusion of the second feature and omission of the first and third feature. The initial state can be chosen at random.
* The goal state or the end state will be an n-dimensional vector that produces the best accuracy so far.
* The neighborhood operator is defined as the addition or removal of one or more features. In the vector space, it is represented as reversing the value of any index in the n-dimensional vector. For example, let the vector***v*** be [0,1,0], apply the neighborhood operator on index 0 would result in a ***v*** *=* [1,1,0].
* The cost function is defined as 1 unit for each change of the vector. This is because only when the vector has changed, the solution is required to be revaluated.
* The heuristic used, when applicable, is a logistic regression classifier’s result based on the selected features. The selected features will be inputted to the classifier, and the output accuracy is used as the heuristic score.

## Graph Search

The first solution explored is the tree search. More specifically, Breadth-first Search has been employed. The tree is built such that note from each level contains one additional feature than the nodes from the previous level. This means that each node represents a different subset of all features available. For a set of n features, there would be n level to cover all possible permutations. There is no heuristics. Moreover, the best possible solution may be any of the nodes. Therefore, the entire tree may have to be traversed to find the best solution. BFS expands each level entirely before going to the next. Hence, if the solution is found, it is guaranteed to be the subset with the least features. Nonetheless, this method is inefficient as both time, and space complexity is exponential.

## Tabu Search

Another potential solution is Tabu search. This method significantly reduces the space complexity as a feature subset require O(n) space. Similarly, other storage required for the algorithm such as Tabu list and aspiration list are also in the same order. A feature that has been used will be added to the Tabu list. The Tabu length has been approximated to the square root of some features. This is to promote exploration. The aspiration criteria are met when using an element in the Tabu list generates a better result. However, Tabu search faces a similar problem with graph search in the fact that it tries to explore too much of the solution space, resulting in exponential running time.

## Simulated Annealing

The second proposed method is Simulated Annealing. This method simulates the physics annealing process, where the explored solutions experience more randomness when the temperature is high and converge while cooling down. Solutions are represented as a binary array of length n (number of features). This method starts the search with a random solution, and a geometric cooling schedule is selected. As in (1), initial temperature T0 is set to 0.2, final temperature T1 is set to 0.2/100, and damping factor α to 0.85. Thus, there are around 28 temperature steps. The iterations per step are set to 100, combining with the cooling steps, plenty of experiments can be guaranteed to explore the search space thoroughly.

*T0 = T0\*α* (1)

In each temperature step, when max iteration is not reached, the neighbor solution is accessed by including or excluding a random feature. If the result is better, the algorithm always jumps to the neighbor solution; if it is not, it jumps with a probability p according to (2).

*P = e(ΔE/-1\*T) where ΔE=accuracy difference* (2)

Therefore, when temperature decreases, there is the lower possibility of diversification, and when the final temperature is reached, the algorithm terminates. The final result is where the algorithm converges.

## Genetic Algorithm

The third trial is a Genetic Algorithm. This method is a good match with feature selection according to the algorithm structure, each feature corresponds to a gene, and solutions are binary arrays holding the status of each feature; thus, solutions correspond to the population. The algorithm starts with random solutions, and the initial population size is set to 1000, the maximum generation number is set to 100 to ensure the convergence of the algorithm. Firstly, the fitness function is thrown to the current population, and rank based parent selection is chosen. Then, the parents are selected with probability P1 (3).

*P1 = rank#/(Σi) for i = [1, rank#]* (3)

Uniform crossover is happened between parents to generate children pool; each gene has 50% to be flipped. After this, mutation happens with a probability between 1/population and 1/chromosome length (number of features). Thus, it ensures the mutation is neither too random nor too slow to process. Mutation brings proper diversification to the searching process. Then the fitness function is thrown again. The algorithm keeps looping until the maximum generation is reached. Then the result is the best one across top solutions in each generation.

## Ant Colony Optimization

Another solution implemented is based on the ant colony optimization algorithm. Feature selection problem was formulated by setting each feature as a node, and the path between the nodes are 1 or 0. So, there are two paths in total connecting each node, which means that the ant can choose to either include the feature in the solution or not. The decision of the ant is based on the probability function, which is related to the global pheromone. For each iteration, after all the ant finishes their trip, the ACO chooses the ant with the best accuracy. Then, the algorithm evaporates the pheromone and update the global pheromone function with the selected ant’s path.

The input parameter of the ACO is the maximum number of iteration (20), number of ants (100), the initial pheromone amount per path (1), the amount of pheromone per update (0.1), and the evaporation rate (0.95). For the probability function per path is depended on the pheromone on that path, For each ant k traverses path 1 with a probability P1 = pheromone-path\*(1/2), traverses path 0 with a probability of P0 = 1-P1. For each iteration, the global pheromone is evaporated using the evaporation function, which specifies the rate of evaporation. specifies the rate of evaporation, when it equals to 1, the move becomes random. For the pheromone update process, for each feature that appeared in the selected ant’s path, a constant value is added to global pheromone array. The final result ACO 97.8, and the time complexity is O (iterations multiple the ants), and space complexity is O(n).

## Simulated Annealing Enhanced Weighted ACO

After implementing all the different trial algorithms, w a new algorithm that optimizes the ACO was created. Named as SA-WACO, this new algorithm uses SA to determine the parameters for the ACO algorithm. Moreover, weight function is also added to penalize the solutions with more features. The decision to create a new algorithm that optimizes the ACO is since ACO give the best result. ACO has many input parameters, so there is an optimized ratio to get the best result. The reason that SA is used to find the best ratio between the ACO parameters is that at a high temperature, the SA algorithm gives a high chance of covering all different ratios. For each SA iteration, from the previous iterations’ ACO result accuracy, the amount of pheromone to update will change according to the accuracy and temperature. The final accuracy is 98.6; the algorithm time complexity is the tradeoff we decide to make for the higher accuracy. The time complexity is O(number of SA iteration\* number of Ant\*number of ACO iteration). The W in the name stands for weighted. The solutions with higher dimension are penalized, which means that if two solutions that have the same accuracy, the one that has less number of feature is chosen.

# Performance Evaluation

Each of the existing algorithms available can be used to perform feature selection to some extent. After researching about each algorithm described above, a quantitative performance test was scheduled using the most stable version of the implementation of each algorithm. The dataset used was Communities and Crime dataset [**4**] which contained features and a predictor to indicate if the crime level is above a certain threshold. The result in TABLE I. has shown the difference in performance in both running time and accuracy for each algorithm. The proposed solution of simulated annealing enhanced weighted ant colony optimization is the best solution.

1. Performance Comparison

|  |  |  |
| --- | --- | --- |
| **Performance Comparison between Different Algorithms** | | |
| ***Algorithm Type*** | ***Result*** | ***Performance*** |
| BFS | 97.2% accuracy | A long period  -Space complexity O(2n)  -Time complexity O(2n) |
| Tabu Search | 97.2% accuracy | Relatively long time  -Time Complexity  depending on some iterations  -Space complexity  O(n) |
| Simulated Annealing | 97.9% accuracy | 353s runtime  -Time Complexity  depending on the number of iterations run  -Space Complexity  O(n) |
| Genetic Algorithm | 95.8% accuracy | 91s runtime  -Time Complexity O (population size\*generation number)  -Space Complexity O(population\_size) |
| Ant Colony Optimization | 97.9% accuracy | 10s run time  -Time Complexity  O(iterations \* ants)  - Space Complexity  O(n) |
| Simulated Annealing enhanced Weighted Ant Colony Optimization | 98.6% accuracy | 100s to 312s run time  -Time Complexity O(ACOIteration\* Ant\*SimAnnealingIteration)  -Space Complexity  O(n) |

Note that for breadth-first search and Tabu search, the complete simulation was almost impossible to perform due to the exponential time complexity of both algorithms. The sample dataset chosen had 32 features in total. The graph search algorithm was only able to perform graph search on ten randomly chosen features simultaneously for the running time to be at a reasonable value. For the Tabu search, the algorithm was only able to perform search on 20 randomly chosen features simultaneously due to time constraints as well. Therefore, the performance of bread first search and Tabu search are much worse than it appears in TABLE I. In theory, graph search is supposed to search through the entire search space and return the most optimal result.

The first algorithm that was able to compute all 32 features within a reasonable period is simulated annealing. Total time of 353s was used for a stable run. The accuracy achieved, 97.9% is also better than that of Tabu search on average. The genetic algorithm outperforms simulated annealing in running time by around 288% with a value of 91s on a feature set of 32 features. However, there’s significant degradation in accuracy from 97.9% to 95.8%. In such, a trade-off occurred for better time performance.

The pure ant colony optimization was able to perform better than all existing algorithms implemented. It achieved an accuracy of 97.9%, no lower than any existing solution, and at the same performed with a stunning running time of 10s for a stable run. However, the pure ant colony optimization lacked some domain specific information for the problem of feature selection, and that is the goal is the reduce the number of features selected while maximizing the accuracy. The weighted ACO structure allows such penalization to occur by adding weight to both accuracy and features chosen as the overall heuristic. On top of such alteration, simulated annealing was also added to ensure the optimal choice of pheromone deposit value and ratio to the initial pheromone. In the end, the Simulated Annealing Enhanced Weight Ant Colony Optimization (SA-WACO) was able to outperform the traditional ACO regarding accuracy, a 98.6% to a 97.9%, at the sacrifice of approximately 10 to 30 times the running time. The different running time accounts for the different cooling schedule used for the simulated annealing process.

# Conclusion and Recommendation

There are three main points derived in conclusion. Firstly, the traditional greedy algorithm is not efficient since it is easily trapped in a local maximum and thus misses the global maximum. Secondly, according to the performance evaluation and complexity analysis, the ranking is SA-WACO>ACO>GA>TABU>BFS. SA-WACO archives best accuracy within an overhead in computation time, where BFS wastes both search space and searching time. Thirdly, the algorithm using heuristics usually has better overall performance, since they utilize the information already hidden in data.

There are two recommendations listed below. SA could be used to determine initial parameters for ACO, thus guarantees ACO to have the most suitable settings for feature selection problem. SA has the randomness that could walk through the search space and find the best parameter ratios within a controllable time. Finally, weighted functions could be used to penalize unrelated data in the high dimensional dataset, therefore, achieve a more skimmed final solution.

##### References

1. J. Brownlee, "An Introduction to Feature Selection", *Machine Learning Mastery*, 2018. [Online]. Available: https://machinelearningmastery.com/an-introduction-to-feature-selection/. [Accessed: 23- Jul- 2018].
2. S. KAUSHIK, "Feature Selection methods with example (Variable selection methods)", *Analytics Vidhya*, 2016. [Online]. Available: https://www.analyticsvidhya.com/blog/2016/12/introduction-to-feature-selection-methods-with-an-example-or-how-to-select-the-right-variables/. [Accessed: 23- Jul- 2018].
3. B. Dadaneh, H. Markid and A. Zakerolhosseini, "Unsupervised probabilistic feature selection using ant colony optimization", *Expert Systems with Applications*, vol. 53, pp. 27-42, 2016.
4. M. Redmond, "UCI Machine Learning Repository: Communities and Crime Data Set", *Archive.ics.uci.edu*, 2009. [Online]. Available: http://archive.ics.uci.edu/ml/datasets/Communities+and+Crime. [Accessed: 24- Jul- 2018].