Volume 4, Issue 1, January 2014 ISSN: 2277 128X

International Journal of Advanced Research in Computer Science and Software Engineering

Research Paper

Available online at: www.ijarcsse.com

An Approach to Combinatorial Problems by MapReduce Based Ant Colony Optimization

Gajanan Baburao Aochar[1]
Dept. Of Computer Engg,
Dr.D.Y.Patil SOET,Lohagaon Pune
University Of Pune, India

Prof.Omprakash Tembhurne
Faculty at Dept. Of Computer Engg,
Dr.D.Y.SOET,Lohagaon Pune
University of Pune, India

Abstract— Ant Colony Optimization (ACO) is a kind of Meta heuristics approach which is simulated from the social behaviour of ants. It could be a good alternative approach to solve NP hard combinatorial optimization problems such as 0-1 knapsack problem and the Traveling Salesman Problem (TSP).The ACO can get a solution that is quite nearer to the optimal solution; however premature input bogs the system down. Parallelization is an effective way to solve large-scale ant colony optimization problems, since the better solution requires larger number of ants and iterations which consume more time. The problem can be solved by Map Reduce based ACO approach

Keywords— Ant Colony; MapReduce; Metaheuristics; Combinatorial Problem; Traveling Salesman Problem (TSP) The capacitated arc routing problem (CARP)

I. INTRODUCTION

A. Background

Ant Colony Optimization (ACO) is a kind of Meta heuristics approach which is simulated from the social behavior of ants. It could be a good alternative approach to solve NP hard combinatorial optimization problems such as 0-1 knapsack problem and the Traveling Salesman Problem (TSP). ACO can get solution that is quite nearer to the optimal solution; however premature input bogs the system down. Parallelization is an effective way to solve large-scale ant colony optimization problems, since better solution requires larger number of ants and iterations which consume more time. The problem can be solved by Map Reduce based ACO approach.

The capacitated arc routing problem (CARP) is representative for number of practical applications such as gas pipeline designing and vehicle routing also this approach can be used in routing of packets in a wireless network. CARP's scope can be continued by including total service time and fixed speculated costs. The proposed hybrid method is used to solve instances of an extended CARP and TSP (Travelling Salesmen Problem). This approach is specified by the adaptive parameters, local optimization techniques and exploitation of heuristic information. Two types of heuristic information such as arc cluster information and arc priority information are obtained continuously from the solution which is used to guide the subsequent optimization process. The adaptive parameter reduces the burden of choosing initial values and facilitates improvement in robustness of the results. Finally, the proposed technique using the combination of two heuristic approaches is employed to improve the overall optimization performance of Combinatorial Problems.

MapReduce is a programming model and an associated implementation for processing and generating large data sets. Users can specify a map function that processes a key/value pair to generate a set of intermediate key/value pairs, and It has a reduce function that combines all intermediate values associated with the same intermediate key. Many real world tasks can be expressed with MapReduce framework, Programs written in MapReduce method are automatically parallelized and executed on a Large cluster of workstation. The run-time analysis of the method handles the details of partitioning the input data, scheduling execution of the program across a set of machines, handling hardware failures in the terme of machine failure, and it also controls the inter-machine Communication

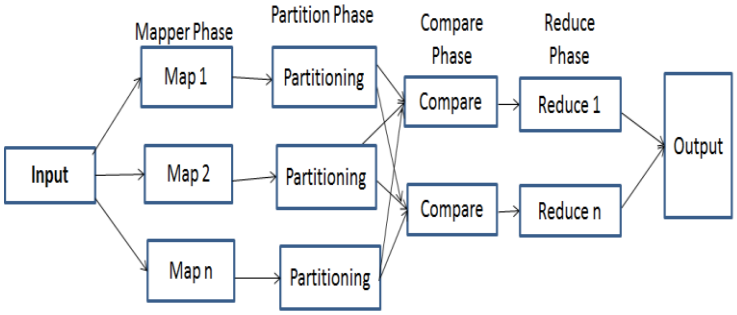

Figure 1. MapReduce Framework

MapReduce allows programmers without any experience with parallel and distributed systems to easily utilize the resources of a large distributed system. MapReduce runs on a large cluster of commodity hardware and is highly scalable.

B. Purpose

This paper focuses on Combinatorial problem which relates to choosing optimal path among the all nodes there are many algorithms for finding out optimal path however ACO can gives optimal solution in a polynomial time using parallelization it is an effective way to perform large scale computing.

The purpose of this paper includes:-

a) ACO algorithm Improvements:-It has several improvements to traditional ACO algorithms involving heuristic factor, selection probability determination and pheromone update strategy.

b) Parallel ACO implementation with MapReduce :-It has been implemented with two parallel versions of ACO algorithms. The first approach illustrates a baseline of MapReduce-based ACO algorithm, and the second approach has different designs depending on the correlation of input items of target problems. 0-1 knapsack problem and TSP problem used as an example. Showing how to partition the input with the MapReduce framework and perform efficient parallel ACO.

C. *Operational Definitions*

A metaheuristic term refers to a master idea that guides and changes other heuristics parameter to produce solutions out of that are normally generated in a quest for local optimality. It is a set of algorithmic concepts that can be used to define various heuristic methods applicable to solve a wide set of different problems. This concept increases the ability of finding a high quality solutions to hard, practically applicable combinatorial optimization problems in a reasonable amount of time and metaheuristic might inspired by the working nature of real ants. beginning with Ant System, a number of algorithmic approaches based on the ideas were expand and applied with considerable success to a variety of combinatorial optimization problems from real-world as well as academic applications.

Combinatorial optimization problems are attractive because these problems are easy to state however very difficult to solve. Many of the problems arising in applications are NP-hard, that is,for finding optimal solution it cannot be polynomially bounded with computation time ,therefore, practically solve large instances approximate methods are required which return near-optimal solutions in a relatively reasonable time.they often use some problem-specific domain knowledge to improve solution either build or algorithms called heuristics..The ACO metaheuristic has been proposed as a common framework for the existing applications and algorithmic variants of a variety of ant algorithms. Algorithms that fit into the ACO metaheuristic framework will be called in the ACO algorithms

D. Combinatorial Optimization

Combinatorial optimization problems involve finding values for discrete variables such that the optimal solution with respect to a given objective function is found. Many optimization problems of practical and theoretical importance are of combinatorial nature. as well as many other important real-world problems like finding a minimum cost plan to deliver goods to customers, an optimal assignment of employees to tasks to be performed, a best routing scheme for data packets in the Internet, an optimal sequence of jobs which are to be processed in a production line, an allocation of flight crews to airplanes, and many more. A combinatorial optimization problem is either maximization or a minimization problem which has associated a set of problem instances. The term problem refers to the general question to be answered, usually having several parameters or variables with unspecified values. The term instance refers to a problem with specified values for all the parameters. For example, the traveling salesman problem (TSP), defined is the general problem of finding a minimum cost Hamiltonian circuit in a weighted graph, while a particular TSP instance has a specified number of nodes and specified arc weights.

More formally, an instance of a combinatorial optimization problem π is a triple (S,f,Ω) where S is the set of candidate solutions, f is the objective function which assigns an objective function value f(s) to each candidate solution $s \epsilon S$, and Ω is a set of constraints. The solutions belonging to the set $\hat{S}$ in S of candidate solutions that satisfy the constraints Ω are called feasible solutions. The goal is to find a globally optimal feasible solution s* For minimization problems this consists in finding asolution $s^* \epsilon \hat{S}$ with minimum cost, that is, a solution such that $f(s^*) < f(s)$ for alls $\epsilon \hat{S}$; for maximization problems one searches for a solution with maximum objective value, that is, a solution with $f(s^*) > f(s)$ for all $s \epsilon \hat{S}$. Note that in the following basically focus on minimization problems and that the obvious adaptations have to be made if one considers maximization problems. It should be noted that an instance of a combinatorial optimization problem istypically not specified explicitly by enumerating all the candidate solutions (i.e., theset S) and the corresponding cost values, but is rather represented in a more concise mathematical form (e.g., shortest-path problems are typically defined by a weighted graph).

II. RELATED WORK

A. Improvements of Ant Colony Optimization

As Mentioned previously, ACO defines some problems that problems are premature and Stagnation problems [4], this problems lead up to stagnation in the local solution and premature solution after a certain number of iterations. Tsutsui introduced the inner mutation and the outer mutation strategies together with a metric called concentrate degree to solve the premature problem [6]. The concentration degree is used to decide whether it's necessary to perform mutation. Inner mutation mutates a certain column in the pheromone matrix based on the concentration degree of the column while outer mutation mutates the whole pheromone matrix. Mutation is performed via adding a random matrix to the existing pheromone matrix. In [3], the probability of selection is amplified in the beginning of the algorithm to speed up the expansion of search space and in the end to boost the convergence. [3] also improved the pheromone update strategy for

the TSP by dynamically weakening or strengthening historical paths. In [7], a unit consisting of two ants: a cunning ant and a donator ant, is used. An in-unit optimization strategy is introduced as well.

B. Parallelization of Ant Colony Optimization

Most existing parallelized ACO works lie in implementing ACO with traditional parallel programming models. M. Manfrin et al. implemented a parallel ACO to the TSP in a multi-machine environment with MPI [10]. Basically, there are two information exchange strategies in parallel ACO: synchronous and asynchronous. In the synchronous strategy, all execution units run synchronously, and the pheromone is updated after all units complete one iteration. In the asynchronous model, each execution unit is run independently and the wait-free pheromone update is performed. Craus and Rudeanu also implemented a MPI-based parallel ACO algorithm with a one-master-multi-slave architecture [11]. In [11], checkpoint is used to update pheromone asynchronously. The master identifies the sequential order of pheromone update committed by slaves with logical clock. Tsutsui and Fujimoto implemented a parallel version of cAS [7] in a multicore environment with Java threading [13]. [10], [11] and [12] all illustrate that the asynchronous model has a higher performance than the synchronous one. Compared with traditional models like MPI, parallelization with MapReduce has less attention in the academia. Huang and Lin implemented a genetic algorithm with MapReduce to the job scheduling problem and has a reasonable output [9]. The approach in [9] requires multiple MapReduce iterations, which drop the performance of the algorithm. The same problem also exists in [5], which implements parallel ACO with MapReduce to the TSP. Experimental result in [5] illustrates that the one-iteration approach has a far better performance than the multi-iteration approach

III. PROPOSED METHODOLOGY

A. Problem Formulation

Let **C** be the matrix of shortest distances (dimension *nxn*), where *n* is the number of nodes of graph *G*. The elements of matrix **C** represents the shortest distances between all pairs of nodes (*i,j*), *i, j*=1, 2, …,*n.* where n is the number of nodes in graph G=(V,E),E consist of two link e1and e2 the goal is to find a closed or minimum path in G that contains each node exactly once called tour and whose length is minimal thus the search space S consist of all tours in G. The travelling salesman problem can be formulated in the category programming binary, where variables are equal to 0 or 1, depending on the fact whether the route from node *I* to node *j* is realized (*xij*= 1) or not (*xit*= 0). Then, the mathematical formulation of TSP (Brezina, 2003) is as follows (the idea of this formulation is to assign the numbers 1 through *n* to the nodes with the extra variables *ui*, so that this numbering corresponds to the order of the nodes in the tour. It is obvious that this excludes subtours, as a sub-tour excluding the node 1 cannot have a feasible assignment of the corresponding *ui* variables):

1) Decision Variable or Independent Variable –In the given System Xd is Decision Variable and d=1,2……..n
2) Input Variable – i and j are independent variable i,j=1,2……n
3) Exogenous variables- L1, L2, lib, lgb,T are some parameter or outside the model variable used in the System
4) Random variables-An artificial ants pheromone value Ƭi,ρ
5) Extra Variable- In the given system Xij,Ui and Uj are extra variable used.

$MIN \sum_{i=1}^{n} \sum_{j=1}^{n} CijXij$

Subject to

$\sum_{i=1}^{n} Xij = 1$ j=1,2……n **i≠j**

$\sum_{j=1}^{n} Xij = 1$ i=1,2……n j≠i

Ui-Uj+nxij≤n-1 i,j=1,2……n

Xijϵ{0,1}

C. Combinatorial Optimization Problem

ρ=(S,f)

In which a finite set of objects S and objective function f

f : S⟶R+

That assigns a positive cost value to each of the objects s ϵ S, Xd is a decision variable d=1,2……..n

E consist of edges e1 and e2 associated with i and j

L1= Length of e1 and e2= Length of e2

Such that L2>L1 e1 represents short path and e2 represent long path.

B.Hybrid ACO

An artificial ants pheromone value Ƭi for each of two links ed such valuve indicates the strength of pheromone trail on corresponding path ηa= artificial ants starting from I an ant chooses with probability

$\rho d = \frac{Ƭd}{Ƭ1+Ƭ2}$ d=1,2……n

Between path e1 and e2 for reaching food source Vj obviously if Ƭ1>Ƭ2

The probability of choosing e1 is higher and vice versa for returning Vj to Vi an ants uses the same path to reach Vj choose edge ed an ant changes the artificial pheromone value Ƭd as follows

$Ƭd \leftarrow Ƭd + \frac{Q}{Li}$ Where Q is the positive parameter of the model The amount pheromone i.e. added depends on the length of the path choosen ,shorter the path higher the amount of added pheromone, pheromone evaporated in the artificial model as follows

$\tau d \leftarrow (1-\rho)\tau d$ d=1,2…….n

Construction step with probability

$\rho(i,j) = \frac{\tau i,j}{\sum[k\epsilon\{1\ldots|V|\}Vk\epsilon T]\tau i,k}$

$\forall j \in \{1 \ldots |V|\}Vj\epsilon T|$ where T is memory

Pheromone evaporation

$\tau i,j \leftarrow (1-\rho)\tau i,j, \ \forall\tau ij\epsilon T$

Pheromone Deposit $\tau i,j \leftarrow \tau i,j + \frac{Q}{f(s)}$

Algorithm For ACO

Algorithm 1 (1+1) ant activity

1) While termination condition not meet
2) Do
3) Schedule Activities
4) Ant Based Solution Construction()
5) Pheromoneupdate()
6) DaemonActions() {optional}
7) End schedule Activities
8) End while

Algorithm 2 ant Based solution construction()

1) S=< >
2) Determine N(s) ≠ ∅
3) C= choose form (N(s))
4) S=extend S by appending solution component c
5) Determine N(S)
6) End while

Transition Probabilities

$\rho\left(\frac{Ci}{s}\right) = \frac{[\tau i^{\alpha}].[\eta(Ci)]^{\beta}}{\sum_{Cj\epsilon N(S)}[\tau j]^{\alpha}[\eta(Cj)]\alpha}$ $\forall Ci \in N(S)$ η optional weighting function depends on current sequence

Improved ACO Algorithm

a) An Improved Solution generation Strategy
It is based on roulette wheel selection and traversal;
b) A New pheromone Update Strategy
A new formula for pheromone update.

A Hybrid ACO

Step 1) Iteration Counter nc, nc ← 0 for every item i
Set an initial pheromone Value τi(0)=τmax
Step 2) Crawling

$\rho\left(\frac{Ci}{s}\right) = \frac{[\tau i^{\alpha}(t)].[\eta(Ci)]^{\beta}}{\sum_{Cj\epsilon restN(S)}[\tau j]^{\alpha}(t)[\eta(Cj)]\beta}$, $Ci \in$ *rest* $N(S)$ *whichis not empty*

0 Otherwise

Step3)PheromoneUpdate

$\tau^{ib}(i,j) = m.\frac{lgb}{lib}$ *τ if (i,j) ϵ route(i,j)*

△ *0 Otherwise*

C. Parallelization

MapReduce, which is usually designed for data-intensive problem, provides a two-phase (map and reduce) divide-and conquer processing strategy. The principle of MapReduce framework illustrates that in the map phase no data exchange between map tasks can be performed. Reduce is the only way to aggregate outputs from different map tasks. This restriction ensures the simplicity and reliability of the model. However, at the same time, it also requires developers to design the partition and aggregation of the problem carefully. Our MapReduce algorithm design aims at solving the problem with existing approaches and make sufficient use of the simplicity and scalability of the MapReduce model. In this paper, two approaches are introduced,

1) Approach 1. The Search Space Replication

In this approach, the input is replicated *m* copies, where *m* is the number of map tasks. Each map task executes the ACO algorithm independently while emitting its output with a constant integer as the key and its ACO result as the

output. Since the outputs all have the same keys, they will be sent to the only reduce task. In the reduce task, the optimal value is selected and recorded as the output. In Approach 1, the input is not partitioned, nor does any information exchange happen between execution units.

2) Approach 2. The Search Space Partition Approach

In this approach, the solution space is partitioned to map tasks. Since the partition is carefully designed, each split has a reasonable locality that can be solved independently. For 0-1 knapsack problem, both items and the pheromone are evenly split and sent to map tasks. Ants crawl for temporary solutions of the partial input based on Equation (1), calculate the ratio of total value and the weight (v/w) of each temporary solution. The pheromone matrix is updated according to the best ant that owns the maximal ratio value.

In the reduce phase, the pheromone emitted from map tasks are connected, and ACO algorithm is further run on the whole items. With the optimized pheromone, only a small number of ants and iterations are needed. As to the TSP, it is unreasonable to divide the input to different map tasks because there may be connection between any two cities. Considering that TSP has a search space of tree structure, we assign different branches of the tree to map tasks. All map tasks have a copy of adjacent matrix, and are assigned with an even number of path prefixes so that the search space is divided into even pieces and solved in a parallel way. Then the reduce task compares all input pairs and outputs the optimal path as the final output of the algorithm.

D. System Design

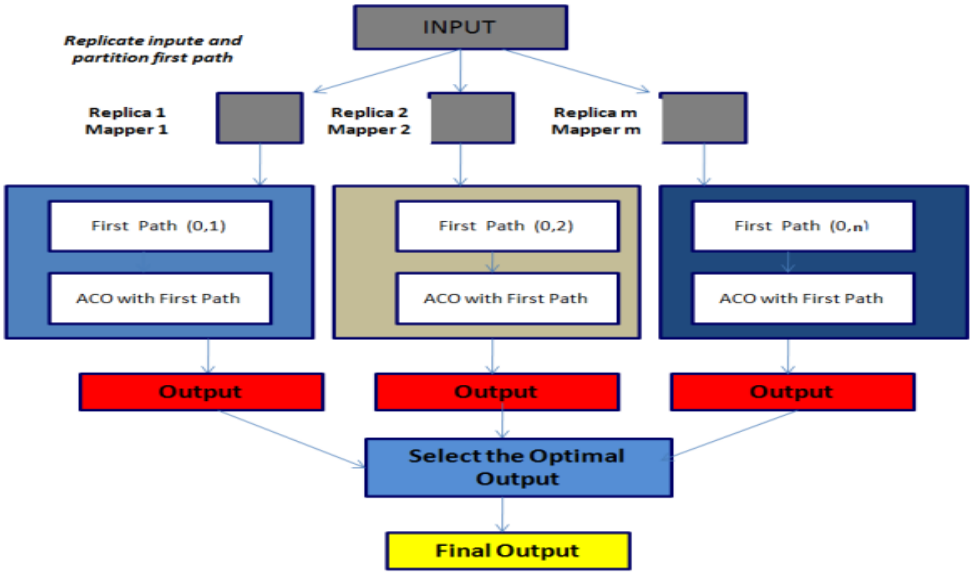

Figure 2.System Architecture (Combines Approach 1and Approach 2)

The proposed architecture kindly defines flow of input and its random execution which is based on search space replication where input are replicated into m number of copies that is it creates a set of replica and partitions the Input with a possible path and for each path the ACO algorithm is applied to each iteration to get a set of possible optimal solution with the help of MapReduce Framework.

IV. EXPERIMENT

A.Environment.

The Proposed system can be implemented with Java 1.7 update 17. Each computer in the cluster has an Intel Q8400 quad-core CPU and 4GB memory, and the operating system Ubuntu 10.04. The MapReduce cluster can be configured with Apache Hadoop 0.21.0 and it will have minimum 4 nodes.

B. ***Dataset for the 0-1 knapsack problem.***

The system will generate a dataset of 1,000 items. The item weight W and value V are independent and uniformly distributed within integer interval The maximal possible value of containable items will be 40,342 according to the dynamic programming algorithm.

C. Dataset for the TSP.

The proposed system will use TSPLIB, the dataset lin105, whose optimal path length will have 27,874

V. CONCLUSION

The proposed technique executes ACO algorithm in distributed environment in order to improve the overall performance of Combinatorial Problems. A hybrid approach is proposed which will improve the performance of ACO using MapReduce framework. The ACO algorithm will be re-designed so as to make the best use of parallel architecture

of processors in Hadoop. The proposed system facilitates the addition of more nodes as and when required. Adding more machines will keep the time complexity constant if the input grows

REFERENCES

[1] Li-Ning, Xing,Philipp Rohlfshagen "A Hybrid Ant Colony Optimization Algorithm for the Extended Capacitated Arc Routing Problem"
[2] Yuren Zhou" Runtime Analysis of an Ant Colony Optimization Algorithm for TSP Instances
[3] Bihan Wu,Gang Wu"A MapReduce based Ant colony Optimization Approach to Combinatorial Problems"
[4] M. Dorigo, V. Maniezzo, and A. Colorni, "The ant system: An autocatalytic optimizing process," Dipartimento di Elettronica, Politecnico di Milano, Milan, Italy, Tech. Rep. 91-016, 1991.
[5] M. Dorigo, V. Maniezzo, and A. Colorni, "Ant system: Optimization by a colony of cooperating agents," IEEE Trans. Syst., Man, Cybern.-PartB, vol. 26, no. 1, pp. 29–41, Feb. 1996 .
[6] M. Dorigo and T. Stützle, Ant Colony Optimization, 1st ed. Cambridge, MA: MIT Press, 2004, ch. 4, pp. 153–222.
[7] M. Dorigo, M. Birattari, and T. Stützle"Ant colony optimization," IEEE Comput. Intell. Mag., vol. 1, no. 4, pp. 28–39, Nov. 2006
[8] M. Craus and L. Rudeanu. Parallel framework for ant-like algorithms. InProceedings of the Third International Symposium on Parallel and Distributed Computing/Third International Workshop on Algorithms,Models and Tools for Parallel Computing on Heterogeneous Networks, ISPDC '04, pages 36–41, Washington, DC, USA, 2004. IEEE ComputerSociety.
[9] S. Tsutsui and N. Fujimoto. Parallel ant colony optimization algorithm on a multi-core processor. In Proceedings of the 7th internationalconference on Swarm intelligence, ANTS'10, pages 488–495, Berlin, Heidelberg, 2010. Springer-Verlag.
[10] M. Manfrin, M. Birattari, T. Sttzle, and M. Dorigo. Parallel multicolony aco algorithm with exchange of solutions.
[11] M. Craus and L. Rudeanu. Parallel framework for ant-like algorithms. In Proceedings of the Third International Symposium on Parallel and Distributed Computing/Third International Workshop on Algorithms,Models and Tools for Parallel Computing on Heterogeneous Networks,

