# Waterflow model

## A new distributed approach to solving combinatorial problems

Benu Madhab Changmai
Computer Science & Engineering Dept.
National Institute of Technology, Karnataka
Surathkal, India
benuchangmai@gmail.com

Shrukul Habib
Computer Science & Engineering Dept.
National Institute of Technology, Karnataka
Surathkal, India
shrukul99@gmail.com

***Abstract*—We propose a new model called the Waterflow model which follows a distributed approach towards solving various combinatorial problems. This model is the manifestation of a very simple and basic concept of the inherent nature of an uninterrupted flow of water to traverse through the shortest path. This model strives to overcome computation time wastage that occurs through multiple iterations as well as ensures the most optimal path unaffected by local optimal greedy solutions. Results show that for systems that require immediate discovery of the most optimal way or method, accurately and precisely at the earliest, this model is a useful approach.**

### I .Introduction

In this paper, we propose a simple yet very efficient model to solve combinatorial problems through a distributed approach.

In the present scenario of computation across the globe and the growing degree of complexities and intensities involved, it has become indispensible to divide the tasks over a network of computational units which in conjunction complete tasks in time and efficiency which otherwise would have never been possible by a single system. However, in order to distribute the tasks it is also required to predetermine the member units of the assisting distributed system that provides the most optimal service under the given conditions and requirements. This task falls under the domain of combinatorial problems. Our model has been proposed keeping in mind the same purpose. The Ant Colony Optimization Algorithm , Particle Swarm Optimization, Simulated Annealing , Tabu Search are some of the existing algorithms that in addition to finding heuristic solutions to combinatorial solutions, also strive to self-optimize on its own.

Our model works on a single iteration computation of solution which determines the most optimal solution to a combinatorial problem from the given sample space.

Let us first give a brief idea of what inspired the algorithm. Consider a pipelining system consisting of a network of pipes and we need to find the find the shortest distance from a particular junction 'a' to another junction 'b'. For this purpose, we consider the starting junction 'a' as the inlet and the last junction 'b' as the outlet. We start the waterflow at the inlet and let it flow through the system. We have a few assumptions to consider which deviations from the natural behaviour of water are:

1. Every segment which has water flowing through it will have water flowing through its complete volume. This is not possible in real life situation because if water flows in two segments of volume V from a segment of volume V, the water gets divided at the junction due to which the branched out water cannot flow through the complete cross section in the latter segments. In our artificial system, we ensure complete coverage of the cross-section by replenishing of water at the junction whenever a new flow of water has to start at the junction.

2. The speed of water at each segment is same. This is not always true in real life situations as hydraulic speed depends on the volume of water as well as the cross-sectional area and other dimensional parameters.

3. As a flow leaves a node, a node-indicator is en-queued to a path trail that the flow carries along its journey.

At the receipt of first flow of water at the outlet, the path through which the flow has arrived through is checked by the trail of node-indicators it has carried along. The trail gives us the most optimal path solution

### II. Literature

Ant Colony and other contemporary combinatorial algorithms like Tabu Ssearch, Simulated Annealing have the most crucial drawback that decision points are driven by greedy approach which often leads to bad final results. Solutions are based on locally optimum moves due to which the apparent optimum results lead to very bad final steps even though started with good initial steps. Hence the solutions are constrained by early steps.

In our algorithm no such constraints are confronted as all the possible solutions are equally considered seeking for the global optimum.

The classical and the existing algorithms are also constraints are also limited in the instances where an immediate and accurate need for the most optimum solution is required. It's a trivial idea that the algorithm provides with the most optimal solution as the solution yielded by our algorithm is a proof and validation on its own that its the most optimal, had it been not, the particular solution won't have showed up.

The existing algorithms although have its own advantage of following a heuristic and a build-up method to get the solution because:

Till the desired and acceptably optimum solution is reached for the whole problem, the components of the task can still be carried out based on the locally available optimal solutions. Over the time the population gradually shifts over to more and more efficient solutions. Throughout the time, the process never stopped. Hence, in real life applications where the functioning of the system as a whole cannot be put into halt for seeking out for the optimal solution, ant colony algorithm suits perfectly.

It also has the advantage that in due course of time it also welcomes any new possible solutions which might not have been present before. And if that component solution proves to make the whole solution better, it replaces a lesser optimal sub solution to the main body, however it takes time for the population to shift over to the newer solution.

Our algorithm follows a distributed approach to distribute the sub-solution tasks to multiple computing units which further distribute their subsequent tasks to other units and finally the merged solutions that arrive through the various paths and techniques at the concerned node, are decided over their attributes by the node and then that series of tasks can be carried on through that path from next time onwards.

## III. Our Algorithm

Flow(0, destination, trail, adjacency, res, attribute params)

Flow(node, destination, trail, adjacency, res, atr_params)

```
if (node is destination)
  account_rank(trail, res, atr_params)
  break

else if ( i not_in_trail(trail,i))
  updated_res = compute(res,node)
 updated_attr_params=analyse_attr(updated_res)

  for i in adjacency
     if i.is_adjacent.node
```

Flow(i, destination, trail.add(node), adjacency, updated_res, updated_atr_params)

Our algorithm is an iterative algorithm which is executed over two phases as follows:

In the ***first phase***, we discover the path in the same way as done by the ant colony algorithm. This leads to feasible and explorable sample space where we can apply the second phase of our algorithm to find the optimal solution. We leave out ants in an open space in search of food. After discovering the food source they come back to their homes leaving pheromone traces over the path they traced back to their homes. These paths marked by pheromone traces would be our sample space for the second phase of our algorithm to obtain the most optimal solution.

In the ***second phase***, we apply the main part of our algorithm. As was said in the waterflow analogy presented before, we trigger the whole system at the starting node A where the first inlet flow is started , the flow reaches with equal speeds to the possible adjacent nodes as was determined in the first phase.

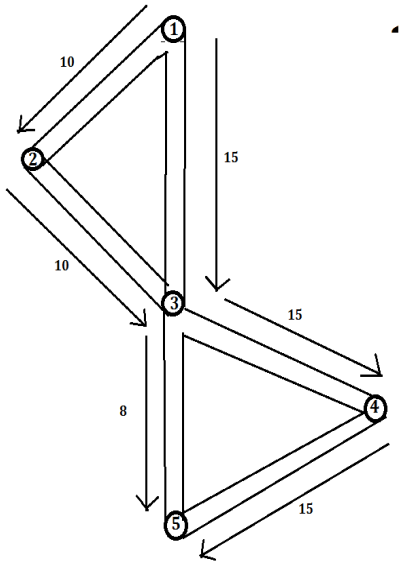


Fig.1(a) A distributed system consisting of 5 nodes. Here the requirement is to find the most optimal way to reach from node 1 to node 5.As can be very easily judged from the segment lengths above that the minimal path from the starting point 1 to the destination point is through the path 1->3->5 and of length 23 units.

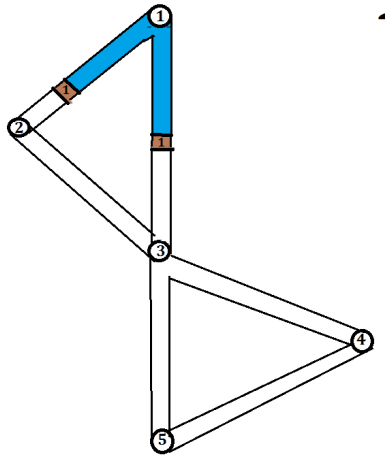

Fig. 1(b). The flow starts at node 1 and branches out towards nodes 2 and 3. It carries along the node-indicators of the nodes it has travelled through.

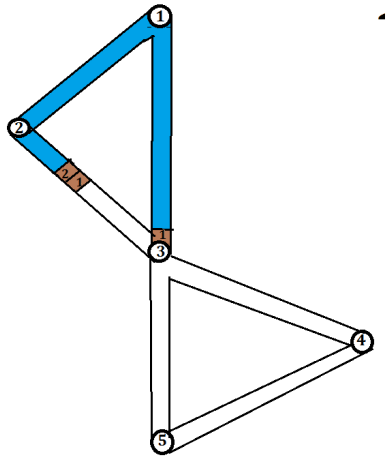

Fig. 1(c). The node-indicators are enqueued as the nodes are crossed.

At each node, as the flow reaches, flows are triggered which starts from the concerned node to all its adjacent nodes. At the same time it takes into account that the flow does flow though the same segments again and again leading to an infinite loop. Suppose the flow that has reached node **3** has arrived with the trail **1->2** and nodes adjacent to **3** adre **4** and **5**, So two separate flows will be branched out from the parent flow as **1->2->3->4** and **1->2->3->5**.

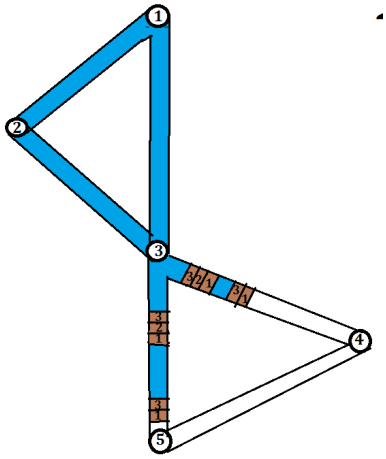

Fig. 1(d). The flow with the traversed trail 1->3 reaches the final destination first , hence the most optimal path

Once it reaches its destination node, that particular trail of flow is stopped and its rank is accounted based on its arrival position. The nodes it came across on its way is taken into account which gives us the optimal paths in order. This way we obtain the most optimal paths rankwise.

## IV. RESULTS

In order to demonstrate our approach we have built a simple multi-threaded program which mimics the distributed approach to solve the problems by handing over tasks to **computation units** which are here threads in our case. Whenever the sub-solution arrives at their subsequent threads computation its displays the nodes traversed. Finally when the solutions reach the final node, it displays the complete path as well as their paths in order of their arrivals. We can rank the threads based on any attributes. In case of our program we have considered arrival time as the deciding attribute.

In order to demonstrate our effectiveness, we have compared our results on a standard dataset of ***Travelling Salesman Problem*** (**TSP**) with an implementation program of Ant Colony Optimization.

It can be clearly seen that the ACO model takes a number of iterations however sm all the number of cities for the **TSP** problem is.

The nature of the graph the ACO model generate when plotted with number of iterations and optimal path distance can be seen in *fig. 2(a)*. It converges to a certain value .


Fig. 2(a). TSP simulation graph using ACO. The graph converges to a minimal tour length after iterations.

On the other hand, it can be clearly observed in our results that under ideal conditions, the optimal solution is found at the very first iteration without further need to converge to a value.

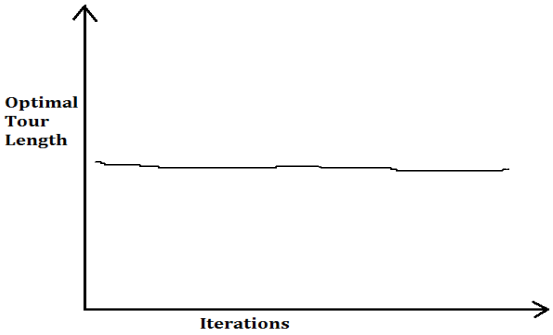


Fig. 2(b) TSP simulation graph using Waterflow model. The model requires just one iteration to arrive at the most optimal solution.

On a standard set of TSP data P01 of 15 cities, when both the model carried out their mechanism to find out the most minimal tour covering all the cities, it was found that the results by ACO converged to a minimal value of 315 after 6-7 iterations or cycles while that by the Waterflow model achieved a minimum value of 291 which is infact the most optimal solution at on its first attempt and continued to show consistent results. The variation in the result though seemingly less but when the number of cities or the node points are more, the variation would be significant. We chose to demonstrate this on just 15 cities so that we can visually describe it on the paper.

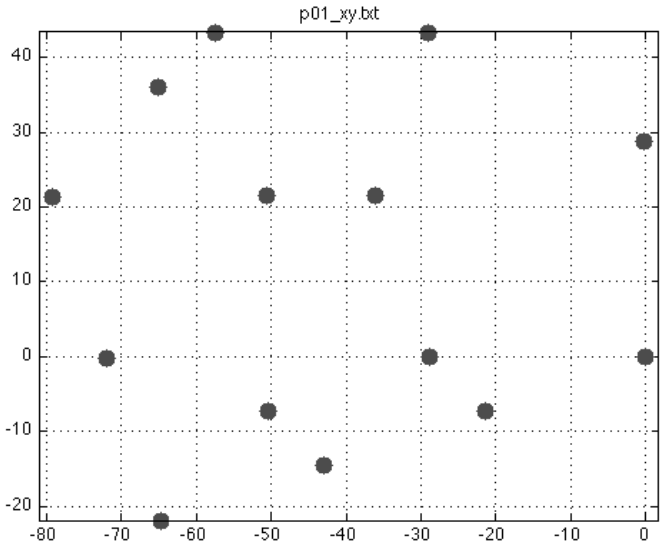


Fig. 3(b). The P01 standard TSP dataset coordinates of 15 cities.

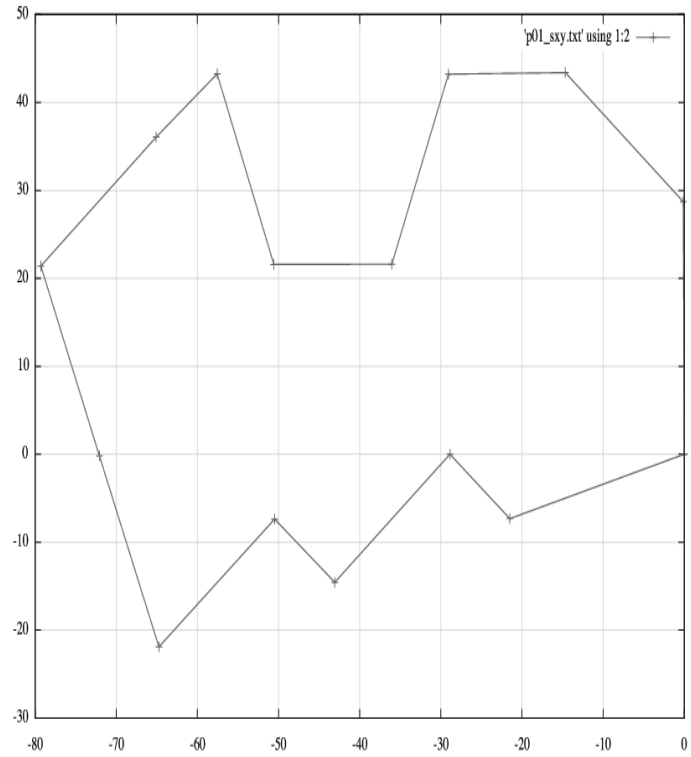


Fig. 3(b). The minimal distanced tour obtained by applying the waterflow model which is of length 291 distance units which is also the most optimal solution.

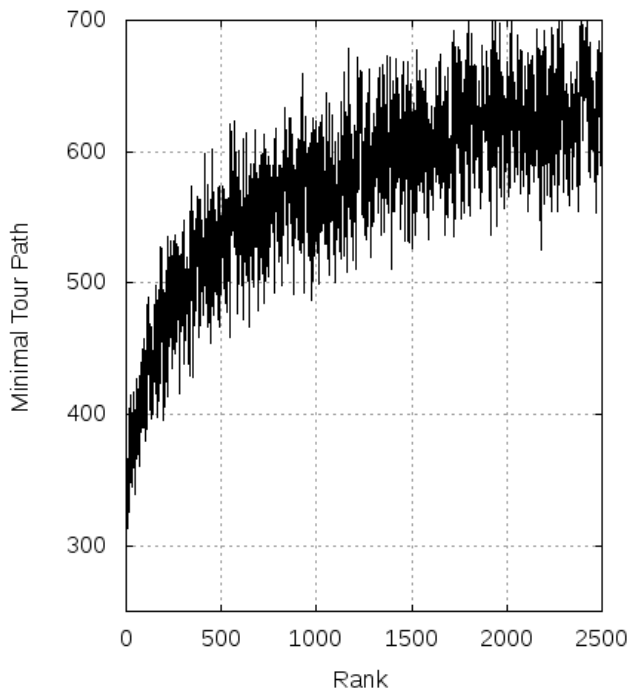


Fig. 3(c). A graphical representation of the minimal tour lengths from the results obtained by running the Waterflow model against their ranks.

## V. Conclusion

This paper introduces a new approach to achieve very accurate and precise solutions to complex combinatorial problems at a significantly faster way. The waterflow model rules out even the slightest of the possibility of being driven by local optimum solutions. It provides a range of possible solutions rankwise which can be analysed and chosen based upon their attributes. Inspired by the simple analogy of water flow, this model promises to cater to the growing requirement for accuracy and speed in distributed computing systems.

.

## VI. References

[1] A. Colorni, M. Dorigo, V. Maniezzo and M. Trubian, "Ant system forjob-Shop scheduling," JORBEL-Belgian J. Oper. Res., Statist. Conzp. Sci., vol. 34, no. 1, pp. 39-53.

[2] A. Colomi, M. Dorigo and V. Maniezzo, "An investigation of someproperties of an ant algorithm," in Proc. Parallel Problem Solving from Nature Conference (PPSN '92), R. Manner and B. Manderick Eds. Brussels, Belgium: Elsevier, 1992, pp. 509-520.

[3] A. Colorni, M. Dorigo, F. Maffioli, V. Maniezzo, G. Righini and M.Trubian, "Heuristics from nature for hard combinatorial problems," Tech. Rep. 93425, Dip. Elettronica e Informazione, Politecnico di Milano, Italy, 1993.

[4] J. L. Denebourg, J. M. Pasteels and J. C. Verhaeghe, "Probabilisticbehavior in ants: A strategy of errors?," J. Theoret. Biol., vol. 105, pp.259-271, 1983.

[5] M.Young,TheTechnicalWriter'sHandbook.MillValley,CA:UniversityScience,1989.

[6] Bianchi L, Gambardella LM, Dorigo M. An ant colony optimization approach to the probabilistic traveling salesman problem. In: Merelo JJ,Adamidis P, Beyer H-G, Fernández-Villacanas J-L, Schwefel H-P, editors. Proceedings of PPSN-VII, seventh international conference onparallel problem solving from nature. Lecture Notes in Comput Sci, vol. 2439. Berlin: Springer; 2002. p. 883–92.

[7] M. Birattari, G. Di Caro, M. Dorigo, Toward the formal foundation of ant programming, in: M. Dorigo,G. Di Caro, M. Sampels (Eds.), Ant Algorithms, Proc. ANTS 2002, Third Internat.Workshop, Lecture Notesin Computer Science, Vol. 2463, Springer, Berlin, Germany, 2002, pp. 188–201.

[8] Dorigo M, StÄutzle T. Ant Colony Optimization. the MITPress, 2003.

[9] Dorigo M, Gambardella L M. Ant colony system: A cooperative learning approach to the traveling salesman problem.IEEE. Trans. Evol. Comput., 1997, 1(1): 53-66.

[10] C. Blum, Theoretical and Practical Aspects of Ant Colony Optimization, Dissertations in ArtificialIntelligence, Vol. 282, Akademische Verlagsgesellschaft Aka GmbH, Berlin, Germany, 2004.

[11] C. Blum, Beam-ACO—Hybridizing ant colony optimization with beam search: an application to open shopscheduling, Comput. Oper. Res. 32 (6) (2005) 1565–1591.

[12] C. Blum, M. Dorigo, Deception in ant colony optimization, in: M. Dorigo, M. Birattari, C. Blum, L.M.Gambardella, F. Mondada, T. Stutzle (Eds.), Proc. ANTS 2004, Fourth Internat. Workshop on Ant ColonyOptimization and Swarm Intelligence, Lecture Notes in Computer Science, Vol. 3172, Springer, Berlin,Germany, 2004, pp. 119–130.

[13] C. Blum, M. Dorigo, The hyper-cube framework for ant colony optimization, IEEE Trans. Systems, Man,Cybernet.-Part B 34 (2) (2004) 1161–1172.

[14] C. Blum, M. Dorigo, Search bias in ant colony optimization: on the role of competition-balanced systems,IEEE Trans. Evol. Comput. 9 (2) (2005) 159–174.

[15] C. Blum, A. Roli, Metaheuristics in combinatorial optimization: overview and conceptual comparison, ACMComput. Surveys 35 (3) (2003) 268–308.

[16] C. Blum, M. Sampels, Ant Colony Optimization forFOPshop scheduling: a case study on different pheromonerepresentations, Proc. 2002 Congr. on Evolutionary Computation (CEC'02), Vol. 2, IEEE Computer SocietyPress, Los Alamitos, CA, 2002, pp. 1558–1563.

[17] C. Blum, M. Sampels, When model bias is stronger than selection pressure, in: J.J. Merelo Guervos et al.(Eds.), Proc. PPSN-VII, Seventh Internat. Conf. on Parallel Problem Solving from Nature, Lecture Notes inComputer Science, Vol. 2439, Springer, Berlin, Germany, 2002, pp. 893–902.

[18] C. Blum, M. Sampels, An ant colony optimization algorithm for shop scheduling problems, J. Math. Model.Algorithms 3 (3) (2004) 285–308.

[19] C. Blum, M. Sampels, M. Zlochin, On a particularity in model-based search, in: W.B. Langdon et al. (Eds.),Proc. Genetic and

Evolutionary Computation Conf. (GECCO-2002), Morgan Kaufmann Publishers, SanFrancisco, CA, 2002, pp. 35–42.

[20] V. C˘ erny, Athermodynamical approach to the travelling salesman problem: an efficient simulation algorithm,J. Optim. Theory Appl. 45 (1985) 41–51.

[21] Zhang J, Hu X M, Tan X, Zhong J H, Huang Q. Implementation of an ant colony optimization technique for jobshop scheduling problem. Transactions of the Institute ofMeasurement and Control, 2006, 28(1): 1-16.

[22] M. Dorigo, T. Stutzle, Ant Colony Optimization, MIT Press, Cambridge, MA, 2004.

[23] H.H. Hoos, T. Stutzle, Stochastic Local Search: Foundations and Applications, Elsevier, Amsterdam, The Netherlands, 2004.

[24] A. Bauer, B. Bullnheimer, R. F. Hartl, and C. Strauss, "An ant colonyoptimization approach for the single machine total tardiness problem,"in Proc. 1999 Congr. Evolutionary Computation, 1999, pp. 1445–1450.

[25] H. M. Bottee and E. Bonabeau, "Evolving ant colony optimization," *Adv.Complex Syst.*, vol. 1, no. 2/3, pp. 149–159, 1998.

[26] K. Bouleimen and H. Lecocq, "A new efficient simulated annealing algorithmfor the resource constrained project scheduling problem," Servicede Robotique et Automatization, Univ. de Liège, Liège, Belgium,1998.

[27] P. Brucker, A. Drexel, R. H. Möhring, K. Neumann, and E. Pesch, "Resource-constraint project scheduling: Notation, classification, models,and methods," *Eur. J. Oper. Res.*, vol. 112, no. 1, pp. 3–41, Jan. 1999.

[28] A. Colorni, M. Dorigo, V. Maniezzo, and M. Trubian, "Ant system forjob-shop scheduling," Belg. J. Oper. Res. Stat. Comput. Sci., vol. 34, no.1, pp. 39–53, 1994.

[29] E.W. Davis and J. H. Patterson, "A comparison of heuristic and optimumsolutions in resource-constrained project scheduling," *Manage. Sci.*, vol.21, no. 8, pp. 944–955, Apr. 1975.

[30] R.C. Eberhart, J. Kennedy, A new optimizer using particle swarm theory, in: Proceedings of the Sixth International Symposium onMicromachine and Human Science, Nagoya, Japan, 1995, pp. 39–43.

[31] Y. Shi, R.C. Eberhart, A modified particle swarm optimizer, Proceedings of IEEE International Conference on EvolutionaryComputation, Anchorage, Alaska, 1998.

[32] J. Dre´o, P. Siarry, Continuous interacting ant colony algorithm based on dense heterarchy, Future Generation Computer Systems 20(2004) 841–856.

[33] K. Socha, ACO for continuous and mixed-variable optimization, in: M. Dorigo et al. (Eds.), Lecture Notes in Computer Science, vol.3172, Springer-Verlag, Berlin, 2004, pp. 25–36.