Facility Locations Revisited: An Efficient Belief Propagation Approach

Wenye Li
School of Public Administration
Macao Polytechnic Institute
Macao SAR, China
wyli@ipm.edu.mo

Linli Xu
School of Computer Science and Technology
University of Science and Technology of China
Hefei, Anhui, China
linlixu@ustc.edu.cn

Dale Schuurmans
Department of Computing Science
University of Alberta
Edmonton, AB, Canada
dale@cs.ualberta.ca

***Abstract*— This paper studies the *fixed-charge facility location problem*—an important problem in logistics and operations research that has wide application to many areas of commerce and industry. The problem is to locate a small number of facilities among nodes in a network to provide good service to the client nodes while confining the total construction cost. The problem is NP-hard, but of sufficient importance to warrant developing practical heuristics. To handle large instances, we propose an algorithm based on recent advances in belief propagation and graphical models. In particular, we adapt a form of *affinity propagation* to approximate the problem at hand. Our experimental results demonstrate significant improvements over other popular heuristics for large-scale facility location problems.**

***Index Terms*— Facility Locations, Belief Propagation**

I. Introduction

The discrete facility location problem is a standard problem in logistics and operations research. It has broad application in many areas of business, government, transportation and communications industry [14], [4], [13], [5], [2]. For example, in business management, almost every enterprise one can think of has the problem of locating facilities at one time or another: industrial firms determine locations for fabrication and warehouses; retail outlets locate stores. In public affairs, government agencies locate offices and provide a variety of services such as schools, hospitals, and ambulance bases. In transportation, agencies set up vehicle inspections and auxiliary facilities. More recently, in tele- and wireless-communications, firms need to decide where to establish routers and base stations.

In these scenarios, the ability of a firm to get its products effectively to the market, or the ability of an agency to deliver high-quality services depends on how the firm's or agency's facilities are located relative to other facilities and to its customers. To formalize this problem, several classical facility location models have been established.

In this paper, we consider a scenario where one is given a discrete set of candidate nodes where the facilities can be located. The goal is to locate facilities in the network to optimize the service quality provided to the clients (nodes) while constraining the overall cost of facility construction. We will address the fixed-charge problem where the cost of constructing a facility at each node is known beforehand. We therefore aim to locate facilities so as to achieve small construction costs while providing good services to the client nodes under different circumstances. Obviously the final facility sites will depend both on the individual construction costs and on the resulting service qualities.

The main challenge is that the fixed-charge facility location problem is NP-hard, therefore it is usually not practical to produce a globally optimal solution. This task remains a central challenge in the field of operations research, where key methodologies have been developed for producing approximate solutions. For small-scale problems one can use relaxation-based methods, which can often achieve good accuracy while providing error bounds. For large-scale problems, however, one is compelled to seek efficient heuristics.

Recently, based on the advances in probabilistic inference and graphical models [11], [7], [1], [15], [16], [10], a method called *affinity propagation* [3] was developed to provide an efficient clustering method. In this paper, we show how this algorithmic strategy can be adapted to the fixed-charge facility location problem. In particular, we develop a natural extension of the affinity propagation method that provides an efficient and highly effective heuristic for this problem.

The remainder of the paper is organized as follows. We first formalize the fixed-charge facility location problem and review popular solution techniques. We then illustrate our new approach based on affinity propagation, and provide an experimental comparison to competing approaches.

II. Fixed-Charge Facility Location Problem

A. Problem Definition

We formulate the (uncapacitated) fixed-charge facility location problem as follows. Given a set of nodes $X = \{x_1, \cdots, x_N\}$ in a network, suppose there are Q states, and let $p_q\,(1 \leq q \leq Q)$ denote the probability of the network being in state q, and γ be the construction cost of a facility. Also let $s_q\,(j, j')$ be a function that measures the quality of service if $x_{j'}$ is chosen as a facility location to serve node

x_j ($1 \leq j, j' \leq N$) in state q. Without loss of generality, we assume $s_q(i,j) \leq 0$ and $s_q(i,i) = 0$ for all i, j, q for the remainder of the paper. We are interested in seeking an assignment $\mathbf{c} = \{c_{11}, c_{12}, \cdots, c_{1Q}, c_{21}, \cdots, c_{NQ}\}$, where c_{iq} indicates the facility that node x_i has chosen in state q, to maximize the objective:

$$L(\mathbf{c}) = \sum_{q=1}^{Q} \sum_{i=1}^{N} p_q s_q(i, c_{iq}) - \gamma C. \quad (1)$$

Here C denotes the number of distinct elements in the assignment $\mathbf{c}$, i.e., the number of facilities that we will construct.

The objective in this model is to locate a number of facilities among the nodes so that the service quality to each point by its closest facility in different states is maximized, balancing the construction costs of the facilities. Rather than requiring the number of facilities to be given *a priori*, this model automatically determines this number by taking into consideration the cost γ, and thus provides a mechanism for automatic facility selection. Sometimes $-\gamma$ is also referred as a *preference* value, which quantifies the extent of willingness or likelihood for a node to be chosen as a facility.

Different from the study of clustering problems in statistics [9], [6], the effect of different states is an important concern when studying facility locations. Let us consider the problem of locating ambulance stations in a city [4]. We hope to trade off the establishment costs versus the averaged travel times from the closest station to each spot. (We assume there are enough ambulances so that each station always has one available for service.) Based on the travel time, we divide the problem into a number of states with respective probabilities such as quiet hours, rush hours, and normal hours. In each of the states, a spot is served by a unique facility. And each spot may use different stations at different states. Generally, the optimal solution to this multi-state facility location problem often has a smaller average response time than the optimal solution if we simply average the states into one and use the same station for each spot in all the states.

When all states have the same probability, i.e. $p_1 = \cdots = p_Q = \frac{1}{Q}$, the objective becomes the maximization of $\frac{1}{Q} \sum_{q=1}^{Q} \sum_{i=1}^{N} s_q(i, c_{iq}) - \gamma C$, or equivalently, the maximization of

$$\sum_{q=1}^{Q} \sum_{i=1}^{N} s_q(i, c_{iq}) - \gamma Q C. \quad (2)$$

To facilitate our discussion, we have assumed two simplifications in the paper. One is that the construction cost at each candidate facility site is the same. The other is that all states have the same probability. This is not always the case in practice. However, our solution introduced below can handle both cases, when the individual costs are different or when the state probabilities are non-uniform, without significant alteration.

B. Conventional Heuristics

Computing optimal solutions to the problems in (1) and (2) is well-known to be hard. Much effort has been made on investigating these problems. Integer programming can be used to obtain an optimal solution if it can scale to the problem instance at hand. However, this method is often impractical. Even a very small problem (say, with a hundred nodes) requires too much time.

For larger problems, one needs to resort to approximations and heuristics to cope with the inherent NP-hardness. Construction heuristics and improvement heuristics are both typically used [2]. The ADD and DROP heuristics are two popular construction algorithms based on a greedy approach. Note that both the service quality and the construction cost increase as facilities are added to the solution. Starting from an empty set, the ADD algorithm greedily adds facilities until it fails to find one whose addition will result in a increase in the objective. The DROP algorithm works in a similar manner except from the other side.

These construction algorithms can be improved by using a substitution and a neighborhood search heuristic [8], [5]. One begins with any set of facility sites and searches for the best possible substitution before making any site exchanges. Every combination of a node in the current solution and a node that is not is evaluated and the best is identified. If that combination increases the objective, an exchange is made. If no combination results in an objective increase, halt. This algorithm can be further improved by a HYBRID approach, at the price of being more complicated. One begins with the solution obtained by a construction heuristic, e.g., by ADD heuristic. Next, the construction and improvement procedures are alternated until no more improvement can be made.

These *seemingly-old* heuristics are popular in practice. Although many years has passed, the HYBRID heuristic still gives state-of-the-art performances in accuracy for many problems and typically forms a baseline in comparison [12].

C. Affinity Propagation

Affinity propagation [3] is a fundamentally different approach that was originally proposed to tackle clustering problems. It can be shown that this algorithm applies to a special case of the problem formulated in (2), which corresponds to the case of $Q = 1$.

When applied to the location problems, the method considers all the nodes as potential facilities and each node is assigned with a preference number $-\gamma$, the negative of the construction cost at each node, that represents *a priori* knowledge of how good the node is as a facility. In the cases that all nodes are equally suitable, all numbers take the same value. This provides a control parameter: the larger the preference value, the more facilities one is likely to find.

Then the method operates by exchanging messages between each pair of nodes until a good set of facilities

emerges. Two types of real-valued messages (*responsibility* and *availability*) are transmitted between the nodes and each undertakes a different kind of competition. The messages are updated by simple rules. The magnitude of each message reflects the current *affinity* that one node has for choosing another node as its facility. At any time, the facilities can be identified by combining the messages flowing in and out. This message-passing procedure will be repeated until the algorithm terminates.

III. Solution

In this paper, we extend the affinity propagation algorithm to solve the fixed-charge facility location problem which allows $Q \geq 1$. The extended algorithm is able to deal with the node objects with dynamic affinities. Using this extension, we provide an efficient heuristic to facility locations.

A. A Factor Graph Representation

To maximize the objective in (2), we define the following:

$$g(\mathbf{c}) = \prod_{q=1}^{Q} \prod_{i=1}^{N} e^{s'_q(i,c_{iq})} \prod_{k=1}^{N} \delta_k(\mathbf{c}) \tag{3}$$

where

$$s'_q(i,c_{iq}) = \begin{cases} s_q(i,c_{iq}) & if\ c_{iq} \neq i \\ -\gamma & if\ c_{iq} = i \end{cases}.$$

$\delta_k(\mathbf{c})$ is 0 if there exist $1 \leq j \leq N$ and $1 \leq q_1, q_2 \leq Q$ such that $c_{jq_1} = k$ and $c_{kq_2} \neq k$; $\delta_k(\mathbf{c})$ is 1 otherwise.

Taking logarithm of (3), we have

$$\log g(\mathbf{c}) = \sum_{q=1}^{Q} \sum_{i=1}^{N} s'_q(i,c_{iq}) + \sum_{k=1}^{N} \log \delta_k(\mathbf{c}). \tag{4}$$

Each $\log \delta_k(\mathbf{c})$ is a penalty term and forces a constraint on the label assignment. The constraint can be understood as follows. Suppose a node x_k has been chosen as the facility site by a node x_i in one state, then x_k must decide to be its own facility in all Q states. These constraints together guarantee a valid configuration of the assignment.

When the objective is maximized, each $\log \delta_k(\mathbf{c})$ is forced to be 0 and the penalty terms are eliminated, leaving the result

$$\sum_{q=1}^{Q} \sum_{i=1}^{N} s'_q(i,c_{iq}). \tag{5}$$

Noting that $s'_q(i,c_{iq})$ contributes a $-\gamma$ if x_i is chosen as a facility and $s_q(i,i) = 0$ for all i, q, the maximizer of (5) can be easily verified to be equivalent to the maximizer of (2).

After establishing the objective equivalence in studying (2) and (3), we study (3) instead. The objective function can be represented by a factor graph. In Figure 1, each function term in $g(\mathbf{c})$ is represented by a function node, depicted in a rectangle, and each label c_{iq} by a variable node, depicted in a circle. Edges exist only between function nodes and variable nodes. A variable node is connected to a function node if and only if the function's term depends on the variable. So the term $e^{s'_q(i,c_{iq})}$ has a corresponding function connected to the single variable c_{iq}. The term $\delta_k(\mathbf{c})$ has a corresponding function connected to variables $c_{11}, \cdots, c_{1Q}, c_{21}, \cdots, c_{NQ}$. The global function, $g(\mathbf{c})$, is given by the product of all the functions represented by the function nodes.

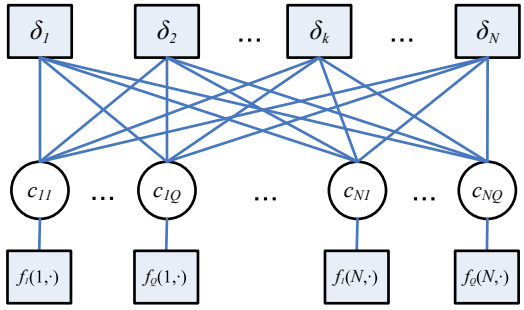


Fig. 1. A factor graph representation of the maximization problem. In the graph, $f_q(i,j)$ denotes a function $e^{s'_q(i,j)}$.

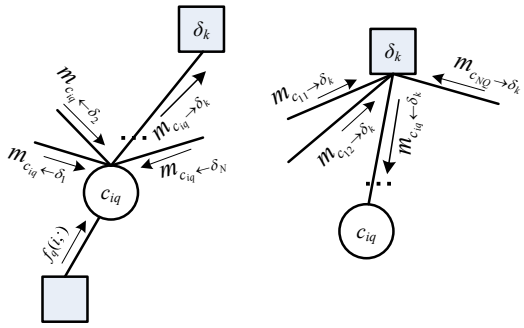


Fig. 2. The messages flow between variable nodes and function nodes. Left: the messages flowing in and out of a variable node. Right: the messages flowing in and out of a function node.

B. A Belief Propagation-based Solution

To maximize $g(\mathbf{c})$ of (3), we use the max-product procedure [7] to search over valid label configurations in the graph, which becomes the max-sum rule after taking the logarithm. Messages are sent iteratively from each variable node c_{iq} to each function node $\delta_k(\mathbf{c})$, and vice versa. In Figure 2, the message, $m_{c_{iq} \to \delta_k}(j)$, sent from c_{iq} to $\delta_k(\mathbf{c})$, consists of N real numbers, one for each possible value j of c_{iq}, and is given by:

$$\sum_{k' \neq k} m_{c_{iq} \leftarrow \delta_{k'}}(j) + s'_q(i,j). \tag{6}$$

The message, $m_{c_{iq} \leftarrow \delta_k}(j)$, which is sent from $\delta_k(\mathbf{c})$ to c_{iq}, is given by:

$$\max_{j_{11}, \cdots, j_{i,q-1}, j_{i,q+1}, \cdots, j_{NQ}} \begin{bmatrix} \log \delta_k(j_{11}, \cdots, j_{i,q-1}, j_{iq}, j_{i,q+1}, \cdots, j_{NQ}) \\ + \sum_{i',q': i'q' \neq iq} m_{c_{i'q'} \to \delta_k}(j_{i'q'}) \end{bmatrix}. \tag{7}$$

Algorithm 1 An Extended Affinity Propagation Algorithm for Fixed-Charge Facility Location Problems.

1: $t = 0$
2: Initialize messages $a_q^t(i,k) = 0 \ \ (1 \le i,k \le N, \ 1 \le q \le Q)$.
3: **repeat**
4: **for** each i, q, k **do**
5: $r_q^{t+1}(i,k) = s'_q(i,k) - \max_{k' \ne k} \left[s'_q(i,k') + a_q^t(i,k') \right]$.
6: $a_q^{t+1}(i,k) = \begin{cases} \sum_{q':q' \ne q} r_{q'}^{t+1}(i,k) + \sum_{i',q':i' \ne i} \max\left(0, r_{q'}^{t+1}(i',k)\right) & k = i \\ \min\left\{0, \sum_{q'} r_{q'}^{t+1}(k,k) + \sum_{i',q':i' \ne k \& i' q' \ne iq} \max\left(0, r_{q'}^{t+1}(i',k)\right)\right\} & k \ne i \end{cases}$
7: **end for**
8: $t = t + 1$
9: Estimate $\hat{c}_{iq}^t$ by $\arg_j \max \left[a_q^t(i,j) + s'_q(i,j) \right]$.
10: **until** some convergence criterion is satisfied
return $\hat{c}_{iq}^t$'s.

Each message update involves N numbers, which is impractical for large problems. But using a similar trick as adopted when deriving the affinity propagation (AP) algorithm, we are able to further simplify the messages significantly. This basic idea is to analyze the different values of j, k, i and eliminate the penalty terms $\log \delta_k(\mathbf{c})$. Then we cancel the constant terms and get the simplification. [1] Finally the message updates of N numbers are reduced to a single number, which makes the message passing efficient in practice.

The simplified scalar messages also involve two kinds. One is a *responsibility* message $r_q(i,k)$, sent from c_{iq} to δ_k, reflects the accumulated confidence for node i in choosing node k as its facility in state q, combining the opinions from other nodes that node k should be a facility. The other is an *availability* message $a_q(i,k)$, sent from δ_k to c_{iq}, collects the evidences from the nodes at different states in deciding whether node k would be an appropriate facility.

The details are given in Algorithm 1. It can be regarded as an extended AP algorithm. When $Q = 1$, this is exactly the standard AP algorithm.

Here we give some more illustrations of the algorithm. The messages are initialized in step 2. We simply set all the availability messages to be zero. We may also choose to initialize the responsibility messages instead.

The major computations are from the message updates in step 5 and 6. For each update, it involves only simple summation and subtraction operations. A naïve implementation would have $O\left(QN^3\right)$ operations for all updates per iteration. However, the message sums can be re-used, and the actual computational requirement becomes $O\left(QN^2\right)$. For sparse problems where a subset of M ($\ll QN^2$) pairwise service relationships between the nodes are available, the computational requirement can be further reduced to $O(M)$, which is appealing for many real applications.

[1] Here we omit the derivation. Details will be available online.

Similar to the standard AP algorithm, a damping factor λ that takes valid values within $[0.5, 1)$ will be used when updating the messages to avoid numerical difficulties in certain circumstances. Thus the message updating rules in step 5 and 6 will be followed by

$$r_q^{t+1}(i,k) = \lambda r_q^t(i,k) + (1-\lambda) r_q^{t+1}(i,k)$$

and

$$a_q^{t+1}(i,k) = \lambda a_q^t(i,k) + (1-\lambda) a_q^{t+1}(i,k).$$

After each iteration, the value of a variable c_{iq} can be estimated in step 9 by summing up all messages flowing into node c_{iq} and taking the value that maximizes the summation.

The algorithm is terminated after some convergence criterion is satisfied. In practice we often choose to terminate the algorithm when the facility assignments do not change for a fixed number of iterations, or when a maximum number of iterations have been reached.

IV. Experiments

To evaluate the effectiveness of the extended AP algorithm, we implemented it in *C* language, compiled as a Matlab subroutine, and carried out a series of experiments.

A. Settings

Our experiments were designed following the test procedure of "K-median" benchmarks and "Euclidean" benchmarks in "UflLib". [2] We randomly generated different numbers (N) of points in a 10-dimensional space according to a uniform distribution. Then we sought a number of facilities from these points to optimize the objective of (2).

For each dataset, we tested different numbers of states, $Q = 1$, $Q = 3$ and $Q = 5$ respectively. In the first state, the measure $s_1(i,j)$ ($1 \le i,j \le N$) is computed as the negative Euclidean distance of a pair of points (x_i, x_j). In

[2] http://www.mpi-inf.mpg.de/departments/d1/projects/benchmarks/UflLib/

TABLE I

COMPARISON OF ACCURACIES. THE EXPERIMENTS THAT WERE NOT FINISHED WITHIN 72 HOURS ARE SHOWN AS "N.A.". $\gamma = 1$ AND $\gamma = 5$ INDICATE THE COST PARAMETER IS SET TO BE 1 AND 5 TIMES THE MEDIAN RESPECTIVELY.

			Relative Performance (%)		
N	γ	Q	AP:ADD	AP:DROP	AP:HYBRID
100	1	1	0.49	1.6	-0.04
		3	0.13	2.3	0.10
		5	0.2	2.8	-0.0015
	5	1	0	0	0
		3	0.03	0.80	0.03
		5	0	3.0	0
500	1	1	1.1	1.3	0.020
		3	0.57	1.9	0.029
		5	0.32	2.4	0.016
	5	1	0.020	1.2	0.020
		3	0.50	1.6	-0.021
		5	0.14	0.85	-0.010
1000	1	1	0.83	2.4	0.040
		3	0.71	1.2	-0.13
		5	0.53	1.5	-0.010
	5	1	0.13	2.5	-0.011
		3	0.32	0.95	0.013
		5	0.017	1.9	-0.010
2000	1	1	0.73	*n.a.*	0.13
		3	0.54	*n.a.*	-0.013
		5	0.55	*n.a.*	0.045
	5	1	0.73	*n.a.*	0.13
		3	0.54	*n.a.*	-0.013
		5	0.55	*n.a.*	0.045

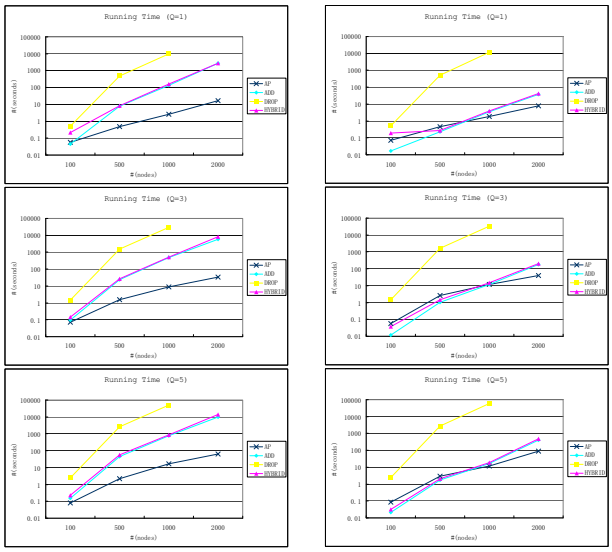
the following state q ($2 \leq q \leq Q$), each measure $s_q(i,j)$ is calculated as the product of $s_1(i,j)$ and a non-negative random number $\left|r_{ij}^q\right|$, where r_{ij}^q is generated from a normal distribution with mean 1.0 and variance 0.1.

We used two sets of cost parameters γ. One is the median of all $s_1(i,j)$'s ($1 \leq i,j \leq N$), which produces a moderate number of facilities, e.g., ˜100 facilities for a $1,000$ nodes. The other is 5 times the median of all $s_1(i,j)$'s, which produces a small number of facilities, e.g., ˜10 facilities for $1,000$ nodes. For AP, we used a default damping factor $\lambda = 0.9$. The message updates are terminated after the identified facilities remain unchanged for 100 iterations, or a maximum of 1000 iterations has been used.

B. Comparisons

We compared the extended AP algorithm with the popular heuristics mentioned above, including ADD, DROP and HYBRID. We run each algorithm on problems with different node numbers (N) and states (Q). We would like to know the *accuracy* among these algorithms. Since the exact solution and objective value of these problems are difficult to obtain, we give the accuracy by the relative residuals. We computed two objective values in (2): L_{AP} using AP and L_{HYBRID} using HYBRID, and obtained the relative performance by

$$\frac{L_{AP} - L_{HYBRID}}{-L_{AP}} \times 100\%.$$



(a) $\gamma = 1 \times median$ (b) $\gamma = 5 \times median$

Fig. 3. Comparison of running time between AP and ADD, DROP, HYBRID algorithms under different settings (from left: $Q = 1$, $Q = 3$, $Q = 5$). The experiments that were not finished within 72 hours are not reported.

With this measure, a positive value means AP performs better than HYBRID in accuracy. The higher the value, the more improvement AP has obtained. On the other hand a negative value means HYBRID gives better accuracy. Similarly we compared AP with ADD and with DROP.

Table I shows the performances. As shown in the table, AP has achieved comparable results with HYBRID. And both are better than the simple ADD and DROP heuristics.

Given the comparable performance in accuracy, we would also like to investigate the computational efficiency. An exact comparison of theoretical complexity is not feasible since the running time of the conventional heuristics depends heavily on the final number of facilities. However, the empirical studies have shown that the extended AP algorithm achieves improvement over conventional solutions. Figure 3 gives the running time under different settings. The horizontal axis depicts the number of nodes used in each experiment. The vertical axis depicts in a log-scale the number of seconds needed to finish an experiment.

When γ is set to be the median of all $s_1(i,j)$'s, for a problem with $N = 100$ nodes and $Q = 1$ state, AP runs only around several times quicker than HYBRID and slightly slower than ADD. However, when the problem gets larger, AP begins to show its clear advantage in running time. For a problem with 2000 nodes and 5 states, AP requires around 50 seconds while all the heuristics require more than $10,000$ seconds. A speedup of over 200 is achieved. Similar trends are also observed when γ is set to be 5 times the median.

From these figures, we can see, although the ADD and DROP heuristics are conceptually simple, their straightfor-

ward implementation requires significant overhead per move. In fact, a theoretic analysis may reveal a complexity of $O\left(QN^2\right)$ for each addition or deletion of facilities, which is prohibitive for large-scale problems.

Another practical concern is the memory requirement. For a problem with N nodes, all the algorithms require equally $O\left(N^2\right)$ storage if the pairwise node service relationship is dense. All the algorithms benefit from the same sparse representations if the relationship is not dense. So their memory requirements are essentially similar. We omit further discussions here.

V. Conclusion

In this paper, we have studied the fixed-charge facility location problem. With a max-product assumption in a factor graph, we propose an extension of the affinity propagation algorithm. The experimental results show that this algorithm has comparable performance in accuracy with the popular search strategies. More importantly it shows clear improvement in running efficiency, and thus provides a reasonable solution to the large-scale problems.

Our approach is fundamentally different from the previous heuristics in facility location problems. It is based on the recent advances in probabilistic inference and graphical models. Although the exact inference over graphical models is generally difficult, the approximation techniques often work surprisingly well and provide high quality solutions to many practical applications. The problem we discussed in this paper is a good example.

Besides the fixed-charge facility location problem, some other facility location and related logistics problems may also potentially benefit from the idea of affinity propagation, which deserves our further study in both scientific research and engineering designs.

Acknowledgment

Wenye Li is supported by MPI project RP/ESAP/01/2010.

References

[1] S. M. Aji and R. J. McEliece. The generalized distributive law. *IEEE Transactions on Information Theory*, 46(2):325–343, 2000.

[2] M. S. Daskin. *Network and discrete location: models, algorithms, and applications*. John Wiley and Sons, Inc., New York, NY, 1995.

[3] B. J. Frey and D. Dueck. Clustering by passing messages between data points. *Science*, 315, 2007.

[4] G. Y. Handler and P. B. Mirchandani. *Location on Networks: Theory and Algorithms*. MIT Press, Cambridge, MA, 1979.

[5] P. Hansen and N. Mladenović. Variable neighborhood search for the p-median. *Location Science*, 5(4):207–226, 1997.

[6] A. K. Jain and R. C. Dubes. *Algorithms for clustering data*. Prentice-Hall Inc., Englewood Cliffs, NJ, 1988.

[7] F. R. Kschischang, B. J. Frey, and H.-A. Loeliger. Factor graphs and the sum-product algorithm. *IEEE Transactions on Information Theory*, 47(2):498–519, 2001.

[8] S. Lin and B. W. Kernighan. An effective heuristic algorithm for the traveling salesman problem. *Operations Research*, 21:498–516, 1973.

[9] J. MacQueen. Some methods for classification and analysis of multivariate observations. In L. M. Le Cam and J. Neyman, editors, *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability*, volume 1, pages 281–297, Berkeley, Califonia, 1967. University of California Press.

[10] M. Mézard. Passing messages between disciplines. *Science*, 301(5640):1685–1686, 2003.

[11] J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann Publishers, inc., San mateo, California, 1988.

[12] M. Sun. Solving the uncapacitated facility location problem using tabu search. *Computers and Operations Research*, 33:2563–2589, 2006.

[13] B. C. Tansel, R. L. Francis, and T. J. Lowe. Location on networks: A survey; part I: The p-center and p-median problems. *Management Science*, 29(4):482–497, 1983.

[14] M. B. Teitz and P. Bart. Heuristic methods for estimating the generalized vertex median of a weighted graph. *Operations Research*, 16(5):955–961, 1968.

[15] M. Welling and Y. W. Teh. Belief optimization for binary networks: A stable alternative to loopy belief propagation. In J. Breese and D. Koller, editors, *Proceedings of the Seventeenth Conference on Uncertainty in Artificial Intelligence*, pages 554–561, San Francisco, CA, 2001. Morgan Kaufmann Publishers.

[16] J. S. Yedidia, W. T. Freeman, and Y. Weiss. Constructing free-energy approximations and generalized belief propagation algorithms. *IEEE Transactions on Information Theory*, 51(7):2282–2312, 2005.