J. Engg. and Appl. Sci. Vol.36 No,1 January-June 2017 ISSN 1023-862X, eISSN 2518-4571

A DISTRIBUTED GENETIC ALGORITM AND A-PRIORI ALGORITHM FOR THE HUB AND FACILITY LOCATION PROBLEMS

[1]Muhammad Naeem, [2]Sahib Khan,[3]Nasir Ahmad

ABSTRACT

A-priori is an influential data mining algorithm employed in market basket analysis to understand the purchase behavior of buyers. It has many other applications. In this study, we combine a-priori with a genetic algorithm (GA) to solve two classical NP-hard location problems namely the Un-capacitated Single Allocation Problem (USAHLP) and Un-capacitated Facility Location Problem (UFLP). A distributed model of the proposed algorithm has been implemented. The performance of the algorithm has been evaluated with standard benchmark problems for USAHLP and UFLP. Results have been found encouraging.

Keywords: Hub Location Problem, A-Priori, Genetic Algorithms, Data Mining, Heuristic, Distributed Computing

INTRODUCTION

The Un-capacitated Single Allocation Hub Location Problem (USAHLP) and Un-capacitated Facility Location Problem (UFLP) are classical problems in distributed and transportation systems where flow of commodities takes place between nodes via special facilities/nodes called hubs. Examples of such systems are air-transportation systems, telecommunication networks, sensor networks, and mail delivery systems. The objective in USAHLP and UFLP is to select hubs and assign nodes to hubs such that the total distribution cost through the network is minimized. USAHLP and UFLP are NP-hard [1][2] and, cannot be solved in finite time with exact methods especially when the input data is large. Therefore, heuristic methods such as genetic algorithm (GA), Tabu Search (TS), Simulated Annealing (SA), and Branch-and-Bound (BB) algorithm have been proposed to USAHLP and UFLP[3][4].

In this work, we have proposed a genetic algorithm (GA) and a-priori based solution to USAHLP and UFLP. We have implemented a distributed model of the algorithm. Genetic algorithm (GA) is a stochastic population based meta-heuristic for solving NP-hard problems [5][6]. Although, not guaranteed to find optimal solution, a GA can find good-quality solution to a problem in finite time. Since its introduction, GA has been applied to many optimization problems with good success. Some of the problems that have been tackled with GA are the Vehicle Routing Problem (VRP), Job Scheduling (JSP), and Time Tabling problems.

A-priori is an influential data mining algorithm heavily used in market basket analysis to assess buying trends of customers in daily market transactions. Primarily, it is used to determine frequent itemsets in a set of market transactions by customers and discover association rules between items. The result is supposed to facilitate business decisions and help boost profit.

Our method has been inspired by the search heuristic used in the GA in which many generations of candidate solutions to a problem are evolved through the operations of selection, crossover, and mutation. The solutions are improved in each successive generation until solutions of acceptable quality are found. Depending on the behavior of the GA, a good number of locally or globally sound solutions are produced during the search process in which only the best solution is selected as the final solution while the remaining solutions are discarded. Our approach is grounded in the perception that the large number of solutions evolved by the GA may contain fragments of locally or globally optimal solutions that may be mined and used to build better solutions. Based on this assumption, the approach presented in this paper retains a few best solutions in each generation and extracts frequently occurring fragments from them using the a-priori algorithm, which are then fed back to the GA to improve the search. Intuitively, we think that with such an approach not only better solutions may be found but also they may be found more frequently.

We have applied our algorithm to USAHLP and UFLP because being combination problems, a solution in these problems resembles market-basket and, therefore, is a good target for the a-priori. As the solutions generated by the GA can be large, mining them takes a long time. We, therefore, have implemented a distributed model of the proposed GA. We also consider distribution useful due to the reason that the a-priori may discover a large number of patterns in the solutions, e.g., for large-sized problems, which can then be processed by the GA in parallel.

We have tested our algorithm on standard benchmark problems selected from OR-Library [7][8]. Large-sized problems have been chosen for this purpose because on smaller problems, the simple GA without a-priori is equally efficient. Preliminary results from the experiment are encouraging underlining the need for further investigation of the approach.

1 Department of Computer Science, University of Peshawar, Pakistan
2 Department of Electronics and Telecommunication, Politecnico Di Torino, Turin, Italy
3 Department of Electronic and Electrical Engineering, Loughborough University, UK

The rest of this paper is organized as follows. In Section 2, we describe the USAHLP (Section 2.1) and UFLP (Section 2.2) problems. In subsections 2.3 and 2.4, we give an overview of GA and a-priori algorithms. In Section 3, we give a review of the existing work on USAHLP and UFLP. Section 4 describes the proposed approach. Section 5 and 6 give experimental evaluation and performance comparison. Section 7 includes conclusion and future work. Section 8 gives references.

THE UN-CAPACITATE HUB AND FACILITY LOCATION PROBLEMS

The Un-capacitated Single Allocation Hub Location Problem (USAHLP) and Un-capacitated Facility Location Problem (UFLP) occur in distribution and facility location networks. We describe them below.Detail can be found in [1] [2] [7].

UN-CAPACITATED SINGLE ALLOCATION HUB LOCATION PROBLEM (USAHLP)

In the USAHLP, a set of geographically distributed nodes called *spokes* send commodities/freight, e.g., passengers, mail, information, etc., to each other via specially designated nodes called *hubs*. The spokes are the origin-destination points for the flow of commodities in the network, whereas hubs are the transshipment points where flows from the origin nodes are collected and sent over the network to the destination nodes usually via other hubs [1] [7] [9].

An illustration of hub-spoke network is given in Figure 1. In the network, nodes $k, l, m,$ and n are hubs whereas the rest of the nodes, e.g., $i, j,$ are spokes. Hubs are fully connected with each other whereas spokes are connected only to hubs and can exchange with each other through hubs only. At least one and at most two hubs are allowed to handle the flow between an origin-destination spoke-pair. Each spoke can be allocated only to a single hub. Moreover, the number of hubs is a decision variable in SAHLP and a fixed cost for establishing a hub is also included in the overall cost of the network.

The USAHLP involves the following decisions.

- Determining the number of hubs to be used.
- Location of hubs, i.e., where in the network should the hubs be located or which nodes should be selected as hubs? This is called the *location sub-problem.*
- Allocation of spokes to hubs, i.e. how are spokes to be assigned to hubs? This is called *allocation or assignment sub-problem.*

The objective in the SAHLP is to minimize the *cost of establishing* hubs and *cost of transportation* by tackling the location and allocation problems. The constraints include the assignment of a spoke to only a single hub and the routing of flows only through hubs (at least one and at most two).

The following cost types give rise to transportation cost in single allocation hub location problem.

- The *Collection Cost*χ, is the cost of flow from the source node to a hub (spoke-to-hub flow).
- The *Transfer Cost* α, is the cost of flow from one hub to another (hub-to-hub flow).
- The *Distribution Cost*$\delta$is the cost of flow from a hub to the destination node (hub-to-spoke flow).

The cost types are interpreted as cost of flow volume between nodes per unit distance. For example, in Figure 1, letW_{ij} be the amount of a commodity that node $i$sends to node j. The package$W_{ij}$is first sent to hub k, subsequently to hub l, and eventually from hub l to the destination node j. The total transportation cost C_{ijkl} is given by,

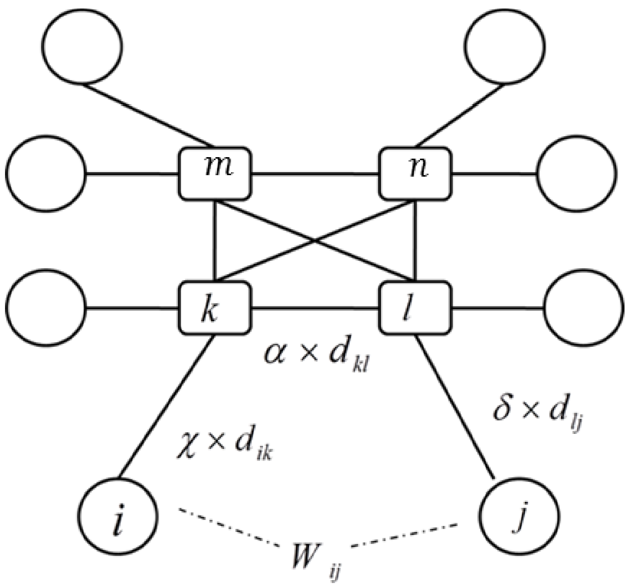


Figure 1 A single allocation hub-spoke network. The nodes $k, l, m,$ and $n$denote hubs and the remaining nodes are spokes. Communication between spokes occur via hubs.

$$C_{ijkl} = W_{ij}(\chi \times d_{ik} + \alpha \times d_{kl} + \delta \times d_{lj})$$

Where d_{ik}denotes the distance between spoke$i$ and hub k, $d_{kl}$is inter-hub distance between k and$l$, and d_{lj} is the distance between hub l and spoke$j$. To calculate the transportation cost for the entire network,C_{ijkl} is aggregated for all pairs of nodes in the network. To the transportation costis also added the cost of establishing the hubs.

A formulation for the USAHLP used in [1][7] is given in Figure 2 below.

$$\text{min}imize \sum_{i=1}^{N}\sum_{k=1}^{N} \sum_{l=1}^{N}\sum_{j=1}^{N} W_{ij}(\chi \times d_{ik} + \alpha \times d_{kl} + \delta \times d_{lj}) + \sum_{k}^{N} F_k Z_{kk}$$

subject to:

$$\sum_{i=1}^{N}\sum_{j=1}^{N} X_{ijkl} = 1 \qquad \forall i,j \in N \ (1)$$

$$Z_{ik} \leq Z_{kk} \qquad \forall i,k \in N \ (2)$$

$$\sum_{i=1}^{N}\sum_{j=1}^{N}\left(W_{ij}X_{ijkl} + W_{ji}X_{jilk}\right) = (O_i + D_i)Z_{ik} \quad \forall i,k \in N \ (3)$$

$$Z_{ik} \in \{0,1\} \qquad \forall i,k \in N \ (4)$$

$$0 \leq X_{ijkl} \leq 1 \qquad \forall i,j,k,l \in N \ (5)$$

where:
$O_i = \sum_{j=1}^{N} W_{ij}$is the flow i sends to other nodes.
$D_i = \sum_{j=1}^{N} W_{ji}$is the flow node i receives from other nodes.
$N = 0, 1, 2, \dots, N-1,$ is the number of network nodes.
F_k is the cost of hub $k, k \in N$
$Z_{kk}$is the decision variable for designating hubs.
$X_{ijkl}$is the fraction of flow between $i$and j routed through hubs k and l.

Figure 2 A formulation for the un-capacitated single allocation hub location problem(USAHLP)

UN-CAPACITATED FACILITY LOCATION PROBLEM

In the single source un-capacitated facility location problem (UFLP), an un-determined number of *facilities* have to be located such that the fixed set-up cost for the facilities and the variable cost of serving the *market demand* is minimized. The demand nodes are assigned to the facilities on a *single-allocation* basis. Like the USAHLP, the facilities have unlimited capacity and establishing a facility incurs a cost. A formulation for the facility location problem is given in Figure 3.

$$\text{minimize} \sum_{i=1}^{N}\sum_{j=1}^{N} c_{ij}x_{ij} + \sum_{i=1}^{N} f_i y_i, x_{ij} \geq 0, y_i \epsilon \{0,1\}$$

$$\sum_{j=1}^{N} x_{ij} \leq S_i y_i \quad 1 \leq i \leq N$$

$$\sum_{i=1}^{M} x_{ij} = D_j \quad 1 \leq j \leq M$$

where:x_{ij} denotes the amount provided from facility I to customer j
y_i givesthe status of a facility(established or not)

Figure 3 Formulation for the un-capacitated facility location problem (UFLP)

GENETIC ALGORITHM

GENETIC Algorithm (GA), first proposed by James Holland in the pioneering work, "Adaptation in natural and Artificial Systems", is a population-based stochastic method applied to search and optimization problems [5][10]. A general structure of a GA is given in Figure 4. The GA mimics the way biological evolution works in nature. Initially, a population of solutions to the given problem is randomly created. Then the solutions are progressively improved by calculating the fitness of the solutions and selecting a pair of *parent solutions* from the population. A pair of *offspring solutions* is produced from the parent solutions through the operations of crossover/recombination and mutation. This is repeated until a new of pool of population called the *offspring population* is generated. The offspring population is again subjected to fitness evaluation and becomes the parent population for the next generation offspring population through the reproduction process (selection, crossover, and mutation) as the predecessor parent population. The process continues for many generations until some termination criteria are met and a population of finally improved solutions is obtained [5]. The best solution in the final population is chosen as the required solution. The algorithm is run many times to discover the best solution. Many NP-hard combinatorial optimization problems have been solved with good degree of success.

Genetic_Algorithm

```
Begin
  Create Initial_Population
  Repeat
Evaluate fitness of solutions in the population
 Select solutions from population for reproduction
Apply crossover and mutation to create offspring
  Until terminating_condition
End
```

Figure 4 A general design of genetic algorithm(GA)

A-PRIORI ALGORITHM

A-priori is a data mining algorithm that is extensively used in market basket analysis. Primarily, it is used to determine frequent itemsets in a set of market transactions. These item sets are then used to discover Boolean association rules that indicate the items occurring together more frequently in the given transactions [11]. Figure 5 gives a description of the algorithm.

To construct itemsets, the a-priori relies on two operations; join and prune. Initially, itemsets of size 1 are constructed from given transactions. These item-sets are called candidate itemsets. Then an iterative process using the join and prune operations is applied to enlarge the itemsets until further enlargement is not possible. In the join operation in each iteration, an item set L_k is joined with itself i.e. $L_k join\ L_k$to produce the candidate itemset C_k. The itemsets in C_k are then pruned according to the given level of support and confidence to obtain the L_{k+1}itemset. In each step, the a-priori condition requiring the subsets of the candidate item sets to be frequent is maintained [11]. The pseudo-code for the algorithm is given in Figure 4.

A-priori Algorithm

```
Join Step:$L_k$ is joined with itself to produce$C_{k+1}$
•Prune Step: Only a $k$-itemset that is frequent can be a subset
of a frequent $(k+1)$ -itemset
$C_k$: Candidate itemset with size $k$
$L_k$: Frequent itemset with size $k$
$L_1$= {frequent items};
While ($L_k$ <> $\emptyset$) Do Begin
  $C_{k+1}$= candidates produced from $L_k$
For each transaction t in database Do
            Increment the number of candidates in $C_{k+1}$
            contained in t
  $L_{k+1}$=    candidates in $C_{k+1}$ with min_support
Increment $k$ by 1
 End While
Return $\cup_k L_k$
```

Figure 5 The a-priori Algorithm

RELATED WORK

Much attention has been paid to the study of Single Allocation Hub Location Problem (USAHLP) in hub related research. In [12], a hybrid heuristic to USAHLP has been proposed based on GA and Tabu Search. The approach uses a GA to determine hubs and Tabu Search to assign spokes to hubs. An improvement over an earlier GA that employed distance-based spoke-assignment to hubs was reported was reported for the hybrid approach. A GA based approach to the solution of USAHLP was also adopted by Topcuoglo *et al.* [13]. The method was tested on the benchmark Civil Aviation Board (CAB) data set and the Australian Post (AP) data and improved solutions were found to some of the CAB problems. Finally, a hybrid approach has been reported in [14]. The method combines a GA with simulated annealing to solve the USAHLP. A PSO based approach has been used in [15] and spatial analysis of performed in [16].

Among the non-GA approaches to the USAHLP are included two hybrid methods by Chen [1] and Silva *et al.* [7]. In [1], a simulated annealing algorithm is combined with TabuList(TL). The Simulated Annealing procedure is used to obtain an upper bound for the number of hubs to be used followed by a restricted single location exchange procedure to locate the hubs. To allocate spokes to hubs, the spokes are first assigned to nearest hubs and then an allocation improvement procedure is applied to re-allocate nodes with lower flow to other hubs until further improvement is not possible. The method proposed by Sinha and Cunha is a hybrid heuristic that uses multi-start tabu search heuristic and a two-stage integrated search heuristic [7] to solve the USAHLP.

Likewise, a number of approaches have been proposed in literature to tackle the UFLP. These approaches include integer programming, approximation algorithms, and various heuristic. Some specific methods cited by [2] are LP rounding and primal dual methods. A genetic algorithm proposed to the UFLP is by Tohyama *et al.* [17]. Resende and Werneck developed a hybrid multi-start heuristic for the UFLP and reported to obtain high-quality solutions to five classes of UFLP instances [18].

Our method is a heuristic based method. Its main distinction from other method is the use of a-priori to mine the GA output to build improved solutions to UFLP. To the best of our knowledge, solving UFLP with GA and a-priori has not been explored before.

PROPOSED APPROACH

As mentioned in section 1, we propose a hybrid GA and A-priori solution to the USAHLP and UFLP. In the proposed approach, the GA produces solutions, which are saved to a disk file that we call dataset. In each generation, only a few best solutions in the child population are saved. Each solution represents a distinct hub combination suggested by the GA. The a-priori converts the dataset into possible hub combinations called itemsets. An itemset is fed back to the GA for further processing. The general layout of the process is given in Figure 6. An explanation of the problem is provided subsequently.

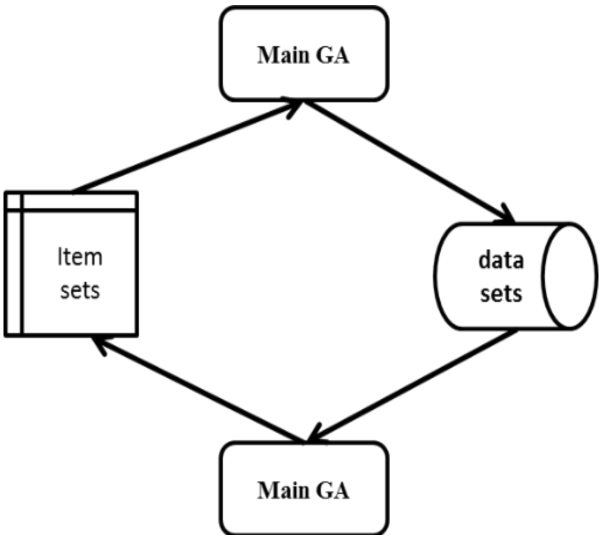


Figure 6 The proposed a-priori based approach

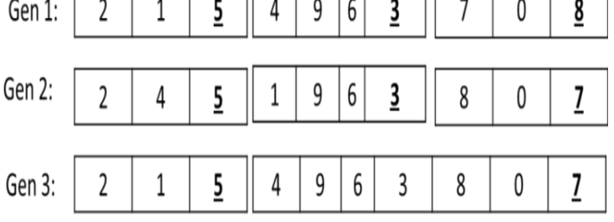
Consider a USAHLP problem of 10 nodes. For sake of brevity, consider solutions found in only three generations (or iterations) during a single run of the GA.The solutions (Figure 7) represent hub-spoke networks $(a), (b)$ and (c) respectively (Figure 7). The best hub combination in generation 1 is 5, 3, and 7. Likewise, best hub combinations in generations 2 and 3 are 5, 3, 8 and 5, 7 respectively. These combinations are saved to the transaction database and form initial transaction (called *hub-transactions*). The a-priori algorithm derives new combinations (itemsets) from this dataset of hub combinations.

As can be seen from Figure 7, the GA designates nodes 5 and 3 as hubs more frequently. Node 8 is designated as hub only once. Thus it may be the case that nodes 5, 3, 7, and 8 are better choices as hubs and may be re-combined with one another in a variety of ways to find hub combinations forproducingbetter solutions. The purposeof using a-priori with GA is to identify useful hub-combination patterns found by the GA in the first pass.

For example, some of the frequent itemsets generated by the a-priori from the transactions may be (5, 3), (5, 7), etc.

GENETIC ALGORITHM COMPONENT

The detail of the GA used in the experiment can be found in the Appendix. It was used in an earlier work by the author [9]. The GA incorporates a specialized crossover for the USAHLP named as Multi-Cluster Exchange Crossover (MCEC).


Gen 1: 2 1 5 4 9 6 3 7 0 8
Gen 2: 2 4 5 1 9 6 3 8 0 7
Gen 3: 2 1 5 4 9 6 3 8 0 7

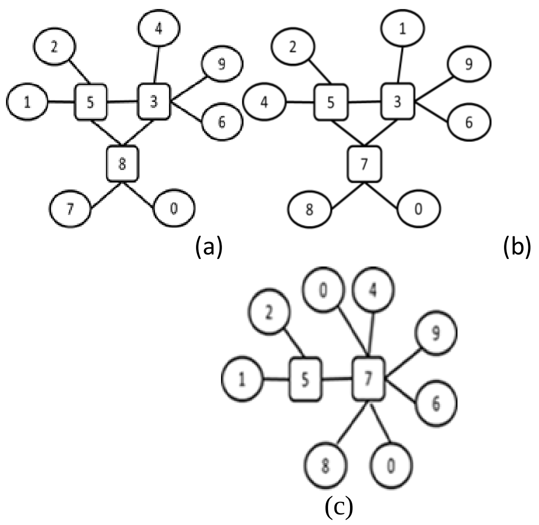


Figure 7 Solutions from generations Gen1, Gen2, Gen 3 and the corresponding hub-spoke networks in (a), (b), (c)

Additionally, there are three mutation operators used in the GA, i.e.,*shift*,*swap,* and *change-hub*,all designed to better adapt the mutation operation to the structure of the solutions in the hub location problems. The GA employs a two-stage 4-tournament selection scheme. Additionally, the GA uses elitism by transferring two best solutions in each generation to the child population directly without applying the operations of crossover and mutation to the elite solutions.

A-PRIORI METHOD

The A-priori used in our algorithm is based on the classical a-priori given in Figure 5.We have made the following modifications to adapt it to the present application.

- A support level of 1 is used to select itemsets from candidate sets.
- Each itemset has a score. This score is the sum of the frequencies of the hubs of an itemset in the transaction database.

To better illustrate this step, generation of itemsets for the example in Figure 7 is explained. T is the transaction database. It contains the hub combinations found by the GA in the first pass. The expressions, $C_k, L_k,$ and S_k, represent candidate itemsets, itemsetlist and scores of the itemsets for level-k.

Transaction database, T = {(5, 3, 7), (5, 3, 8), (5, 7)}

$$C_1 = \{ (5), (3), (7), (8)\}$$
$$L_1 = \{ (5), (3), (7), (8)\}$$
$$S_1 = \{3, 2, 2, 1\}$$

The items 5, 3, 7, and 8 in candidate itemset C_1occurs 3, 2,2, and 1 time respectively as shown in S_1. The frequencies as well as scores of 5, 3, 7, and 8 are 3, 2, 2, and 1. Because these frequencies are greater than the required support level 1 so all these items are included as itemsets inL_1.C_2 is constructed by merging itemsets in L_1. Next $C_2$is pruned by discarding (7,8) because 7 and 8 together appear 0 times in T, which is below the required support level of 1.The rest of the itemsets are included in L_2.

$$C_2 = \{(5,3), (5,7), (5,8), (3,7), (3,8), (7,8)\}$$
$$L_2 = \{(5,3), (5,7), (5,8), (3,7), (3,8)\}$$
$$S_2 = \{5, 5, 4, 4, 3\}$$

The score 5 for the itemset, (5, 3) in$L_2$is obtained by summing frequencies of 5 and 3 in the transaction database$T$, which are 3 and 2 respectively. The total score of itemset (5,3) is thus 5. Likewise, scores of itemsets (5,7), (5,8), (3,7), and (3,8) are 5, 4, 4, and 3 as given in S_2. Level 3 itemset list L_3 andthe corresponding scores are below.

$$C_3 = \{(5, 3, 7), (5, 3, 8)\}$$
$$L_3 = \{(5, 3, 7), (5, 3, 8)\}$$
$$S_3 = \{7,\ 6\}$$

The final itemsetC_4 below- although with zero support in the transaction database T- is generated to check whether better hub-combinations can result from the last list of itemsets (L_3in this case).

$$C_4 = \{5,3,7,8\}$$
$$L_4 = \{5,3,7,8\}$$
$$S_4 = \{8\}$$

Each itemset in an itemset list (i.e., L_k) is considered a hub-combination to be tested by the GA in pass II. The scores of the itemsets in a given itemset list $L_k$given by $S_k$define their priorities in the list. The GA processes the itemset (i.e., the hub combinations) in decreasing order of their scores, e.g., inL_3, the hub combination (5, 3, 7) has higher score, i.e., 7 compared to score 6 of combination (5, 3, 8), so (5, 3, 7) is processed first.

GA WORKING IN PASS II

The GA processes hub combinations produced by a-priori one at a time. To do this, the GA creates an initial population of solutions in which half of the non-hub nodes (i.e., spokes) are assigned to the hubs based on shortest-distance and half are assigned randomly. For example, to test the hub combination (5, 3, 7) in L_3, the GA creates an initial population of solutions in which each solution has 5, 3, and 7 as hubs and the remaining nodes as spokes half of which are assigned randomly and half according to shortest distance heuristic. So, in the second pass, hubs are fixed and the solutions are improved in terms of optimal assignment of spokes to hubs. Also, in the second pass, we keep the GA population small, so the processing is fast.

DISTRIBUTED IMPLEMENTATION

In the approach that we propose, the GA produces a large number of solutions. Consequently, the resulting data set is large and the a-priori takes a long time to convert it into itemsets. Furthermore, the number of itemsets is usually large and a non-distributed implementation of GA takes a long time to process it.

We are proposing distributed implementation of the proposed approach (Figure 8). In this implementation, there is a main GA that produces solutions, i.e., hub combinations and spokes allocated to hubs. The solutions are saved to a disk file as the dataset. The dataset is processed by a distributed system based on the client-server or master-slave model. In this model, there is a server, i e., GAP server (genetic algorithm with a-priori), which retrieves items from dataset and pass them to a-priori clients running on distinct machines.

The a-priori clients convert the datasets into itemsets, which are passed back to the GAP server. The GAP server in turn passes the itemsets to GA clients running on different machines one at a time. The GA clients produce solutions from the itemsets. The solutions are passed to the GAP server again, which selects the best solution from amongst the solutions it receives from the GA client.

For smooth operation of the distributed implementation, we handle the issue of non-active clients through exchange of sent-alive messages between server and clients. In this approach, the server keeps a list of clients that connect to it. It also maintains a record of work assigned to different clients. Clients send alive messages to server every 45 seconds and a low-level thread running on the server side updates the client information accordingly. When a sent-alive message from a client is not received, it is removed from the list of active clients and put into another list. The work assigned to the client is marked un-assigned and assigned to another client upon a work request from an active client. Later, if an inactive client comes alive, it is added to the list of active clients again.

DESIGN ASSUMPTIONS FOR THE PROPOSED APPROACH

The design of the proposed algorithm is based on the following assumption.

- The primary GA is efficient and yields good-quality solutions.
- The allocation of spokes to hubs is not sparse and most of the spokes clusters to few hubs. This assumption is important because sparse allocation of spokes to hubs involve considering large number of hubs by the a-priori, leading to combinatorial explosion.
- Communication between clients and server takes negligible time.

EXPERIMENTAL EVALUATION

The proposed algorithm was coded in Java and run on Pentium 4 Core 2.7 Ghz PC in Windows 10environment. To achieve the distribution Java RMI technology was used. To evaluate the computational performance, an experiment based on 5 trails of the algorithm for each problem instance was performed with the GA parameters as in Table I.

The values in Table I were determined empirically after extensive test runs of the algorithm with different values of the GA parameters.

DATA SET

To test the performance of the proposed algorithm, two sets of data sets were considered. For the USAHLP (Un-capicitated Single Allocation Hub Location Problem), the recently proposed [19] large-sized problems of 300 and 400 nodes were used. The USAHLP problems are chosen from Australian Post (AP) data set [1], which is a suit of standard bench-mark for the Single Allocation Hub Location Problem (SAHLP).

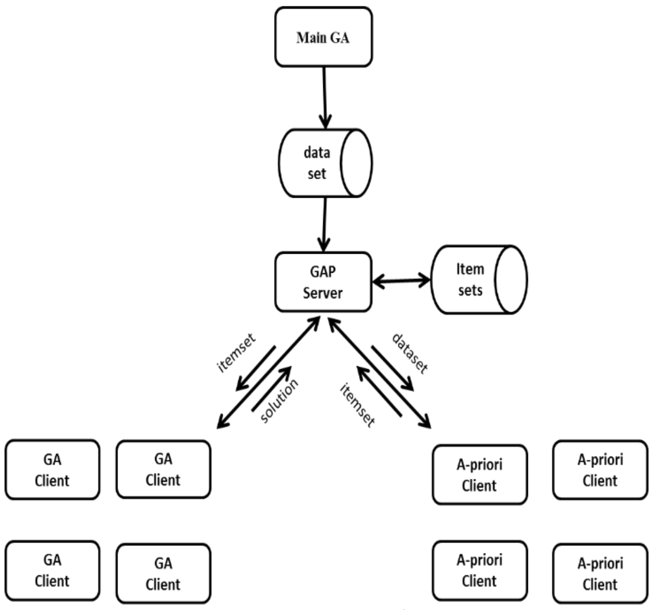


Figure 8 Distributed implementation of the proposed approach

Two varieties of these problems were used; the lose-cost (LL) 300LL and 400LL problems; and the high-cost(TL) 300TL and 400TL problems. The difference between the lose-cost and tight-cost problem is that, the problems with tight-cost have more variability in the hub-cost and, therefore, tend to be difficult to solve.

The set of problems considered for the Un-capacitated Facility Location Problem (UFLP) include 3 large-problem instances ofsize 1000×100 (i.e., 1000 demand nodes and 100 facilities) each. These problems,labelled capa, capb, and capc, are selected from the OR-library and have been used extensively in the facility location research [20].

TABLE I: Parameters of the GA

Parameter	Primary GA	Client GA
Population Size	500	200
Population	generational	generational
Initialization of the chromosome	random	Hubs fixed, spokes assigned randomly and based on shortest distance
Generational Span	500	1000
Probability of crossover	0.75	0.75
Probability of Mutation	0.4	0.4

COMPUTATIONAL RESULTS AND COMPARISON

Experimental results are presented in Table II and Table III. The first column of each table gives the names of the problems used in the experiments. For comparison with other results from literature [7][8], in the second column are included the current best solutions to the problems. Comparison is also given with the unpublished results of the non-distributed GA and *a-priori*. Last two columns of each table list results and computational time(secs) for the distributed GA and a-priori.

Following notation is used to describe the results.

- A value in bold represents the current best solution for a problem
- √ indicates a solution value that is the same as the current-best.

TABLE II: Non-distributed GA with *Apriori*

Problem Instance	Optimal Value [7][17][20]	Non-distributed GA with A-priori		
		Solution Cost	Gap	CPU Time
capa 1000-100	17156454.478	√	0.00	6600
capb 1000-100	12979071.582	√ (found once)	0.00	6500
capc 1000-100	11505594.329	11509361.659	0.03	7302
Hub 300LL	264837.88	**264531.906**	0.00	7128
Hub 300TL	276047.75	**276023.34**	0.00	6332
Hub 400LL	268164.13	269071.25	0.33	6791
Hub 400TL	284212.47	**284124.88**	0.00	6589

- An non-bold value indicates a solution value inferior to the current best

As shown, the proposed approach was able to obtain high quality solutions to the problems relatively more frequently. Except for the capc, to which both the non-distributed and distributed GA-Apriori fail to find optimal solution, capa and capc are solved to optimality by the distributed GA. However, the gap as indicated, is very small. Moreover, for both capa and capc, the distributed algorithm finds high quality values multiple times.

In the case of the USAHLP also, the distributed algorithm finds solutions of high-quality to all the problems with the exception of the problem instance 300LL. Although, for the 300LL problem, the proposed algorithm couldn't find known best solution but the gap is still low. Again the distributed algorithm finds the best solution multiple times. The computational time for both the USAHLP and UFLP instances is reasonable.

TABLE III: Distributed GA with A-priori

Problem Instance	Optimal Value [7][17][20]	Distributed GA with A-priori		
		Solution cost	Gap	CPU Time
capa 1000-100	17156454.478	√	0.00	3056
capb 1000-100	12979071.582	√ (2 times)	0.00	2456
capc 1000-100	11505594.329	11509361.65	0.03	2671
Hub 300LL	264837.88	**264526.02**	0.00	2315
Hub 300TL	276047.75	**276023.34**	0.00	2154
Hub 400LL	268164.13	269071.25	0.33	2635
Hub 400TL	284212.47	**284124.88**	0.00	2321

CONCLUSIONS AND FUTURE WORK

A distributed implementation of the a-priori algorithm was used with GA as explained on previous pages to solve the Un-capacitated Single Allocation Hub Location Problem (USAHLP). The assumption was that, the a-priori would construct better hub combinations from those already obtained by the GA, which could be used to find better solutions. The approach was implemented and tested on standard benchmark problems. The results were found to be promising. As anticipated, the distributed approach yielded high-quality solutions in relatively shorter time. Some of the known best values were found multiple times.

However, some issues in the proposed approach were also observed. For example, the a-priori depends on the effectiveness of the GA itself and as long as the GA produces good solutions in Pass I, the a-priori can improve the solutions. Secondly, the integration of the a-priori with the GA is also an issue. Currently the a-priori is working as a supporting routine. A tighter integration of the a-priori with the GA may result into better performance. Use of more suitable mining techniques for extracting hub-combination patterns from the solutions produced by the GA is another direction for future work. Finally, an effective spoke-assignment heuristic may take better advantage of the rule of the a-priori.

In future, the proposed method will be applied to more USAHLP and UFLP problem instances. Moreover, its performance on large-sized Capacitated Single Allocation

Hub Location Problem (CSAHLP) instances (300 and 400 nodes) will be evaluated. For better assessment of the effectiveness of the proposed approach, other NP-hard problems e.g., Bin-Packing Problem (BNP), Graph-Colouring Problem (GCP), etc. will also be considered. Combining a-priori with other heuristics for NP-hard problems such as tabu search and simulated annealing [3] as well as application of the approach to other optimization problems will be explored.

REFERENCES

1. Chen, J. F., (2007), "A hybrid heuristic for the single allocation hub location problem", *Omega*, vol. 35, no. 2, pp. 211-220.

2. Lazic, N., Frey, B. J., and Arabi, P. (2010), "Solving the uncapacitated facility location problem using message passing algorithm", *AISTATS*, vol. 9 pp. 429-436.

3. Edlecamp, S., (2011), *Heuristic Search: Theory and Applications*, Morgan Kaufman.

4. Lionel, D., (2008), "Branch and bound algorithm for a facility location problem with concave site dependent cost", *International Journal of Production Economics*, vol. 112, no. 1, pp. 245-254.

5. Darrel, W., (1994), "A Genetic Algorithm Tutorial", *Statistics and Computing*, vol. 4, no. 2, pp. 65-85.

6. Maqsood, S., Noor, S., Khan, M. K. and Wood, A., (2012). "Hybrid Genetic Algorithm (GA) for job shop scheduling problems and its sensitivity analysis." *International Journal of Intelligent Systems Technologies and Applications*, vol. 11, no. 1-2, pp. 49-62.

7. Silva, M. R., Cunha, B. C., (2002), "New simple and efficient heuristics for the uncapacitated single allocation hub location problem", *Computer and Operation Research*, vol. 36, no. 12, pp. 3152-3165.

8. OR-Library, Un-capacitated warehouse location problem,http://people.brunel.ac.uk/~mastjjb/jeb/orlib/uncapinfo.html.

9. Naeem, M., (2009), "Using Genetic Algorithm for the Single Allocation Hub Location Problem", M. Sc. Thesis, Brock University, unpublished.

10. Maqsood, S. (2014). "The scheduling of manufacturing systems using Artificial Intelligence (AI) techniques in order to find optimal/near-optimal solutions." Thesis, University of Bradford, UK

11. Markus, H., (2005), "The A-Priori Algorithm-a Tutorial", WSPC/Lecture Notes Series.

12. Sue, A. H., (1998), "A Hybrid Heuristic for the Un-capacitated Hub Location Problem", *European Journal of Operational Research*, vol. 106, no. 2-3, pp. 489-499.

13. Topcuoglo, C., Ermis, Y., (2005), "Solving the Un-capacitated Hub Location Problem Using Genetic Algorithm", *Computer and Operations Research*, vol. 32, no. 4, pp. 967-984.

14. Sibel, A., Bahar, K., (2008), "Network Hub Location Problem: State of the Art", *European Journal of Operational Research*, vol. 190, no. 1, pp. 1-21.

15. Bailey, A., Ombuki-Berman, B., Asobiela, S., (2013), "Discrete PSO for the single allocation hub location problem", *IEEE Workshop on Computational Intelligence In Production and Logistic Systems*, Singapore.

16. Peker, M., Kara, B.Y., Campbell, J.F. et al., (2016) "Spatial analysis of single allocation hub location problem, Springer Link, vol. 16, no. 4, pp. 1075-1101.

17. Tohoyama, H., Ida, K., Matsueda, J., (2011), "A genetic algorithm for the un-capacitated facility location problem", *Electronics and Communication in Japan*, vol. 94, no. 5, pp. 47-54.

18. Resendo, G. C. M., Werneck, F. R., (2006), "A hybrid multistart heuristic for the uncapacitated facility location problem", *European Journal of Operational Research*, vol. 174, no. 1, pp. 54-68.

19. Silva, M., Cunha, C. B. D., (2009), "New simple and efficient heuristic for the uncapacitated single allocation hub location problem", *Computers and Operation Research*, vol. 36, no. 12, pp. 3152-3165.

20. OR-Library, Hub location problem, http://people.brunel.ac.uk/~mastjjb/jeb/orlib/phubinfo.html.