Query Processing in Distributed Data Warehouse using Proposed Dynamic Task Dependency Scheduling Algorithm

S. Krishnaveni and M. Hemalatha
Department of Computer Science
Karpagam University, Coimbatore
Tamilnadu, India

ABSTRACT
A data warehouse is an electronic storage of huge amounts of data. It is also a system for retrieving and managing a data. In distributed data warehouse, data can be shared across multiple data repositories. Each may belong to one or more organizations. Query sorting is the problem of formatting the number of queries to be selected together. Reducing the usual completion period of a random order is a common concern. In this paper we propose Dynamic Task Dependency Scheduling (DTDS) Algorithm for query scheduling. Here our proposed algorithm takes the arrival time, size and also it considers the query dependency from the given query. It is also adaptable for all distributed data warehouse systems. Performance results show that the proposed algorithm gives less processing time and minimum query cost compared to others.

Keywords
Data Warehouse, Random Scheduling (RS), Optimal Resource Constraints (ORC), Grouping based Fine-grained Job Scheduling (GFJS), Heuristic Algorithm (HA), Dynamic Task Dependency Scheduling (DTDS)

1. INTRODUCTION
Stored data are uploaded from the operational systems in the data warehouse, where the data can go through an operational data for supplementary operations before it is used for reporting. The crucial mechanisms of a data warehousing system are,

- To retrieve and analyze the data
- To extract, transform and load data
- To manage data dictionary

Main data warehousing applications are personal productivity, data query and reporting and planning and analysis. Statistical packages, spreadsheets and graphics tools are personal productivity applications that are used in individual computers for manipulating and presenting data. Small amount of warehouse data required to develop a standalone environment. Distributive warehouse data are accessed by data query and reporting applications which are list oriented queries and they provide an overview of historical data. The planning analysis applications share a set of user requirements. They cannot be met by applying query tools against the historical data maintained in the warehouse repository.

Modern trends in distributed data warehouse have improved the significance of query selecting. In distribution logistics some of the large quantity queries are being exchanged by a number of small queries, which have to be practiced in extremely rigid time gaps. Distributed data warehouse containing more than two local data warehouses at each collection point and coordinator site. If the client gives queries in distributed system, query processing performed at local sites [1]. Skalla system is designed for distributed data warehouse to evaluate OLAP queries. Skalla translates OLAP queries to reduce the amount of data that needs to be shipped among sites.

Query selecting is the process of salvage items from their storage locations to fill customer orders, is known as the most time consuming and laborious component of the warehousing activities [2]. So the query selecting operation is a strong candidate for productivity improvement studies. Performance and competence of the query selecting operations are inclined by four vital factors, like warehouse layout, map-reading and sorting procedure, storage policy and grouping method [3].

Scheduling is used to form a resource's start and end times of activity and satisfy the execution of resource capacity constraints, and optimize few sets of performance objectives to the extent possible. Task scheduling is the designing of tasks or queries to specific physical resources to reduce the cost function processed by the client. This is an NP-complete problem and different heuristics may be used to reach an optimal or near optimal solution [4]. Effective computation and task scheduling are rapidly becoming one of the main challenges in various computing systems and is seen as being vital for its success.

2. LITERATURE REVIEW
We survey various task scheduling algorithms that focuses task scheduling, query selecting and scheduling approaches, time and cost minimizations, etc. The simple allocation schemes such as First Fit back fills (FF) are used in practice [5]. In any transactions First In First Out (FIFO) algorithm does not prioritize and transactions are performed based on their arrival time. Scheduling procedures are based on First-Come-First-Serve (FCFS) algorithm [6] which allocates the resources for tasks based on their arrival time. The benefit of FCFS grants the level of determinism on the waiting time of each task [7]. Demerit of FCFS shows, the tasks in the ready queue cannot be scheduled immediately due to lacking of resources but the tasks in the queue would be able to execute given the currently accessible resources. These latter tasks are blocked from executing while the system resources are remaining idle [8].

Hierarchical Job Scheduling (HJS) model [9] is based on a hierarchical approach using global and local level scheduler. The global scheduler uses a separate queue for different type of tasks and local scheduler uses a single queue for all tasks for scheduling with the FCFS, SJF or FF policy. The global scheduler has more functions, from that anyone is identical to

the resources are requested for participating clusters by a task or queries. Others are the best utilization of the available clusters.

In Scheduling Framework for Bandwidth-Aware Job Grouping-Based scheduling (SFBAJG) algorithm [10], tasks are scheduled in system by the use of bandwidth-aware scheduling and also consider their computational and communication capabilities of the resources. It uses network bandwidth of resources for priority determination of every resource. At the time of information retrieval, task grouping method was used and maximize the resource utilization. After that the grouped tasks are sent to earliest finished resources.

A task scheduling model based on Maximum Processor Utilization and Throughput (MPUT) scheduling algorithm [11] that exploits the CPU utilization, throughput and reduces turnaround time. This Job Schedule Model Based (JSMB) algorithm provides reliability with sensible load balance but it does not contemplate any constraints of tasks and resources.

In Highest Response Next (HRN) Scheduling [12] tasks are allotted to the processors based on their priority as well as processor's capability. It provides high responses with memory, CPU requirement and time. This has effectively completed all the tasks quickly than First Come First Serve (FCFS) and Shortest Job First (SJF). But it is not suitable for huge number of task allocation as there are considerable amount of CPU and memory wastage is there. HRN's turnaround time is also high.

In Resource Co-Allocation for Scheduling Tasks with Dependencies (RCSTD) algorithm [13], each step combines the clusters based on the dependencies between the combined clusters. Therefore these clusters are combined if any dependencies exist between current and former clusters. The aim of this algorithm is to enhance the load balancing efficiency and minimum time for the execution of tasks. This algorithm minimizes the task execution time and it has a dynamic nature as a result of within a cluster the tasks are allocated to the appropriate resource on that it can be scheduled at the earliest time. This RCSTD algorithm found a sensible load balancing for all the resources for a set of tasks scheduled for each resource in the system. Cluster communication overhead and unspecified task requirements are the main demerit for this algorithm.

Optimal Resource Constraint (ORC) Scheduling [14] allocates tasks according to their processor's capabilities. ORC applies Round Robin (RR) scheduling and a Best Fit algorithm to distribute the tasks for available processors. It gives better performance than FCFS, SJF and RR. It reduces the average waiting time, turnaround time and minimizes the process allocation complexity. High communication overhead is the difficulty for this algorithm.

Grouping-based Fine-grained Job Scheduling (GFJS) algorithm [15,16] is based on resource characteristics. This algorithm integrated with Greedy and FCFS algorithms improve the Fine-grained jobs. Then the coarse-grained tasks are formed by the grouping of fine-grained jobs. These coarse-grained are allocated to the available resources according to their capability (MIPS) and bandwidth (Mb/s). GFJS maximizes the resource utilization, reduces the task execution time, processing time and network latency. High preprocessing time & memory size constraint inconsiderable are the main drawbacks for this algorithm. Grouping strategy considers the processing power, memory size and bandwidth requirements of each task realize the real grid system. Provides a real grid computing environment and reduces the waiting time of the grouped tasks [17].

Heuristic algorithm [18] is used in experience based learning (EBL) for calculating the processing time. Heuristic algorithm is not considering all the possible schedules. It selects some possible schedules that are having the shortest sum of completion time and this set contains the optimal one. Patient scheduling is done by the use of this algorithm. This shows the patient's minimal waiting time in hospital and minimizes the total completion time.

In distribution logistics few-but-large quantity orders are being replaced by many-but-small orders. The optimal routing policies for a warehouse with multiple cross aisles that can be found by using dynamic programming [19]. There is a model to determine the optimal layout for minimizing the throughput time of a data warehouse and the yields are randomly stored [20]. Construction Management in Decision Support System (CMDSS) can provide exact and timely information to support project managers in construction decision-making [21].

Grouping queries to reduce the entire travel time for a multiple-aisle selector-to-part warehouse considered and the problem is still NP-hard in strong sense when the quantity of orders per batch is better than two [22]. A branch-and-price algorithm is intended to solve instances of modest size to optimality. For larger instances, it is instructed to use an iterated descent approximation algorithm.

In [23] focus on finding an approach for determining the optimal selecting batch size to order-pickers in a typical 2-block warehouse that is a simple but efficient approach also supports the average waiting time of a random order is a convex perform of the group size. It is difficult to capture the impact of aisle blockage, composite-Poisson arrivals or other storage methods and various layouts.

More than 300 papers surveyed in [24] and classified the literature on setup time consistent with store environments, group and non-group setup times, sequence-dependent and independent setup times (costs), and task and group availability models. Also they mentioned the issues of resource-dependent task and setup constraints, task and set-up corrosion, and task or group transportation. They suggest that future researches have to concentrate a specific solution method.

Every client order contains a set of tasks that must be shipped as one group at the same time. They proposed a new Minimum Flow Time Variation (MFV) dispatching rule for client order scheduling in a normal task shop to minimize the total completion time of all tasks within the same order [25]. This rule will efficiently minimize the finished goods' storage level and controls the waiting time before they can be shipped but it does not concentrate the finished time.

The query grouping issue is essential for operating manual picker-to-parts query or task selecting systems in distribution warehouses efficiently. The proposed meta-heuristics [26] are related to the different capacities of selecting devices, antithetic routing policies and required scenarios. They suggest the researchers to focus the minimization of overall query or task selecting time for issues involving due dates.

Depending upon this literature survey we have concluded that existing task scheduling algorithms, which are used in grid computing, are more efficient than other data warehouse related algorithms. In this paper, we are implementing four

existing algorithms and also proposed a Dynamic Task Dependency Scheduling algorithm which overcomes the demerits of existing task scheduling algorithms and it outperforms in terms of time and query cost.

3. OVERVIEW OF EXISTING SCHEDULING ALGORITHMS

Task is the smallest identifiable and an essential piece of work to be done or to be undertaken. Task scheduling is the designing of tasks or queries to specific physical resources to minimize the cost function and reduce the overall completion time processed by the client. It is one of the main challenges in distributed data warehouse system. Some of the existing task scheduling algorithms are taken for the comparison with proposed systems. They are Random Scheduling, Optimal Resource Constraints, Grouping-Based Fine-Grained Job scheduling and Heuristic Scheduling algorithms.

3.1 Random Scheduling (RS) Algorithm

Random scheduling is a randomized version of shared queries. It assigned resources for all queries randomly.

Algorithm steps

1. Initially available queries are assigned to resources randomly
2. Find the execution time of queries from the current random solution
3. Again generate another random solution and find the execution time of queries
4. If new execution time better than the previous execution time then continues with step2
5. If there are no solutions available then previous best solutions then terminate

3.2 Optimal Resource Constraint (ORC) Scheduling algorithm

ORC allocates tasks according to their processor's capabilities. It includes the mixture of Best fit algorithm and Round Robin scheduling to allocate the tasks in queue pool.

Algorithm Steps

1. Put all incoming queries into the queue pool
2. Searches the all queries in the queue and check the resources and its capability
3. The scheduler will allocate the queries to the resources based on its capability and tasks size
4. The tasks put it in Round Robin queue and again search the resource's capability to process the queries
5. After the completion of allotted queries by the resource, the queries are allotted to the best fit free resources
6. This process will be continued until all the queries are completed

3.3 Grouping-Based Fine-Grained Job Scheduling (GFJS) Algorithm

Based on the resource status, lightweight tasks are grouped as coarse-grained tasks consistent with processing capabilities (in MIPS) and also the bandwidth (in Mb/s) of the available resources. The processing capability and bandwidth are used to constrain the sizes of coarse-grained tasks.

Algorithm Steps

1. The scheduler receives the queries and gets resources status
2. According to the size of queries, query_list is sorted in descending order
3. Based on the resource status, small queries can be grouped as coarse-grained tasks according to processing capabilities and the bandwidth of the available resources
4. Processing time of the coarse-grained task should not exceed the expected time
5. Here only the processing capacity and bandwidth are used to constrain the sizes of coarse-grained
6. If any new tasks come, it will be allocated to an appropriate resource without grouping, if a coarse-grained task running in that resource.
7. Then the fine-grained task can be grouped as several new tasks and this group size should be less than the capacity of available resources

This method could sufficiently utilize the capability of the resources since the grouped tasks match the capacity of resource in a more proper way. When there is no more resource left, FCFS algorithm is used, once a resource node finishes its task, it will be assigned a new grouped task.

3.4 Heuristic Algorithm (HA)

By using this heuristic method, only some schedules need to be considered that will contain the optimal schedule. The heuristic function is selected such that the schedules that are created by this function should contain the optimal schedule. The query scheduling problem can be effectively solved using this heuristic method. Thus the optimal schedule having the minimum total weighted completion time and total tardiness is obtained.

Algorithm Steps

1. Scheduler collects the details of the number of queries, number of resources and the number of queries already assigned to each resource
2. Calculate the weighted sum of total completion time and total delay of each resource then tries to minimize the weighted sum by rearranging list of queries
3. Generate all possibilities for rearranging queries and resources to minimize the overall weighted sum
4. Among these generated the correct possible scheduling sequences select one with the minimum weighted sum of total completion time and total delay
5. Every iteration the weighted sum is calculated based on previous total completion time of query and delay of each resource

Though these existing algorithms work fine there are few demerits, to overcome these issues we propose a Dynamic Task Dependency Scheduling Algorithm.

4. PROPOSED DYNAMIC TASK DEPENDENCY SCHEDULING (DTDS) ALGORITHM

Proposed Dynamic Task Dependency Scheduling algorithm schedules tasks in distributed data warehouse systems by considering their query dependency and resource status. The main aim of this algorithm is to minimize processing time of the tasks and reduce the memory size.

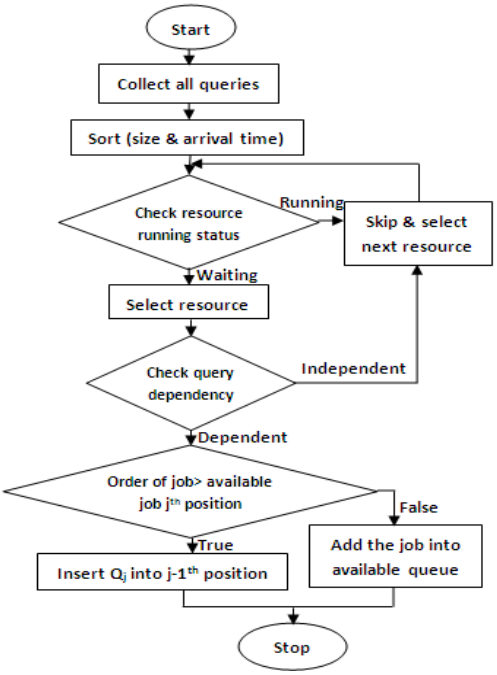

Fig 1: Framework for Proposed DTDS Algorithm

To start with, collect all the queries from client and sorting them by size and arrival time. Then check resource or processor status. If the resource is in running state just skip and search the other resource. When find the resource is in waiting state, queries are scheduled and mapped into that particular resource based on its dependency. Again resources are mapped to queries based on earliest free time of the resource and earliest start time of the query. If the resource is free check out all dependent queries are executed. Then insert the independent queries based on sorting order until all the queries are executed which is shown in Fig.1.

Proposed DTDS Algorithm Steps

1. Collect all the queries and sort them according to the size and arrival time
2. Initially queries are first scheduled and mapped to resources by its dependency
3. The resources are mapped to the queries based on earliest free time of the resource r_i and the earliest start time of the query q_i on the resource r_i
4. For each resources check if the resource is in running state, if it is skipping that resource then select next one
5. If the resource is having any available slot check whether the dependent queries are executed
6. If depended queries are executed insert this query based on sorting order
7. Repeat steps from 4 until all queries are executed

5. IMPLEMENTATION & RESULTS

Queries are randomly generated by the client in a distributed data warehouse environment. Submitted queries are allocated into different inter-processors according to scheduling algorithms. After query processing is completed the results are collected from inter-processors by the server and sent to the corresponding client. Results are compared based on processing time (in seconds) and memory size (in bytes/1000). While comparing various scheduling algorithms our proposed Dynamic Task Dependency Scheduling Algorithm performs well. So the proposed DTDS algorithm yield results in minimal query cost.

5.1 Data Set Description

In this work food mart data set has been used. It contains twenty four relevant tables that are stored in MS Access database. The tables are randomly distributed into different sites.

5.2 Performance Analysis

5.2.1 Performance of various scheduling algorithms with Time

Table 1 shows the processing time (seconds) for different number of queries for Random Scheduling (RS), Optimal Resource Constraints (ORC), Grouping-Based Fine-Grained Job Scheduling (GFJS), Heuristic Algorithm (HA) and proposed Dynamic Task Dependency Scheduling (DTDS) algorithms.

Table 1. Process Time for number of queries

Number of Queries	Time (seconds)				
	RS	ORC	GFJS	HA	DTDS
10	25	15	10	6	4
20	25	20	18	12	10
30	35	25	22	15	10
40	35	28	25	19	14
50	45	32	25	24	18
60	70	40	28	24	23
70	95	40	30	32	29
80	105	48	32	38	35
90	120	63	40	38	36

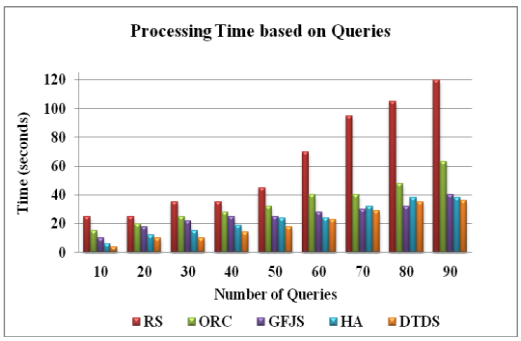

Fig 2: Performance of Scheduling Algorithms with Processing Time

Fig. 2 shows the processing time of RS, ORC, GFJS, HA and proposed DTDS algorithms with a number of queries and it proves that the proposed DTDS algorithm outperforms than other four existing algorithms. While the total number of queries increases the processing time is less in the proposed DTDS algorithm when compared with other existing algorithms.

5.2.2 Performance of various algorithms with Memory Size

The Table 2 shows the memory size (in bytes/1000) for different number of queries for different algorithms (RS, ORC, GFJS, HA and proposed DTDS).

Table 2. Memory size for number of queries

Number of Queries	Memory Size (bytes/1000)				
	RS	ORC	GFJS	HA	DTDS
10	180	150	110	70	25
20	340	280	190	150	55
30	500	430	280	200	125
40	660	600	370	220	125
50	820	750	460	280	165
60	1020	800	570	375	165
70	1200	880	680	420	205
80	1200	1050	770	500	275
90	1400	1180	870	560	315

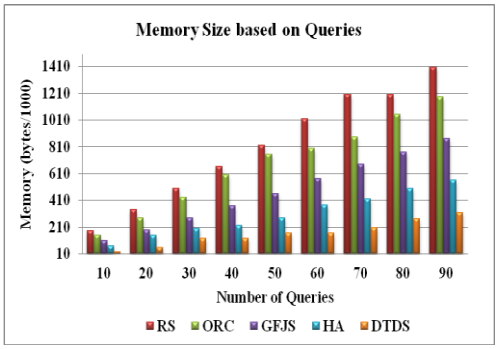

Fig 3: Performance of Scheduling Algorithms with Memory Size

Fig. 3 Despites the memory size of RS, ORC, GFJS, HA and proposed DTDS algorithms with a number of queries. It shows that the proposed DTDS algorithm outperforms than other existing algorithms. While the total number of queries increases the memory size is less in the proposed DTDS algorithm when compared with other existing algorithms.

6. CONCLUSION AND FUTURE WORK

In this paper, we surveyed various scheduling algorithms which are used in data warehouse as well as grid computing. From that we took four existing scheduling algorithms for comparison. Food mart data set was used and it contains twenty four relevant tables which are randomly distributed in various sites (inter-processors). Implementation result has shown the processing time (seconds) and memory size (bytes/1000) with respect to the number of queries. RS algorithm assigned resources for all queries randomly and gave the result to the client. ORC algorithm schedules the queries according to their processor's capabilities. GFJS algorithm schedules the queries based on the processor's processing capability and bandwidth. Heuristic algorithm (HA) selects few possible schedules that are contained optimality and shows the result in short period. Proposed Dynamic Task Dependency Scheduling algorithm schedules the queries based on its dependency as well as resource status. In the above performance analysis we found that the proposed DTDS algorithm outperforms than other existing algorithms. Our future work will be focused on resource management in the distributed data warehouse for avoiding the interruptions at the time of system failure.

7. ACKNOWLEDGEMENT

We thank the Karpagam University for the Motivation and Encouragement to make this work as successful one.

8. REFERENCES

[1] Akinde, M.O., Bhlen, M.H., Johnson, T., Lakshmanan, L.V.S., Srivastava, D., 2003. Efficient OLAP query processing in distributed data warehouses. Information Systems 28, 111-135.

[2] Tompkins, J.A., White, J.A., Bozer, Y.A., Tanchoco, J.M.A.T., 2003. Facilities Planning. John Wiley & Sons, New York, chap. 7, 432-444.

[3] Petersen, C.G., 1997. An Evaluation of Order Picking Routing Policies. International Journal of Operations & Production Management 17 (11), 1098–1111.

[4] Raksha Sharma, Vishnu Kant Soni, Manoj Kumar Mishra, Prachet Bhuyan, 2010. A Survey of Job Scheduling and Resource Management in Grid Computing. World Academy of Science, Engineering and Technology 64, 461-466.

[5] Vijay Subramani, Rajkumar Kettimuthu, Srividya Srinivasan, Sadayappan, P., 2002. Distributed Job Scheduling on Computational Grids using Multiple Simultaneous Requests. 11th IEEE International Symposium on High Performance Distributed Computing, 359-366.

[6] Claus Bitten, Joern Gehring, Uwe Schwiegelshohn, Ramin Yahyapour, 2000. The NRW-Metacomputer-Building Block for a Worldwide Computational Grid. 9th Heterogeneous Computing Workshop, 31-40.

[7] Carsten Ernemann, Volker Hamscher, Uwe Schwiegelshohn, Ramin Yahyapour, 2002. On Advantageous of Grid Computing for Parallel Job Scheduling. 2nd IEEE/ACM International Symposium on Cluster Computing and the Grid, 39-46.

[8] Hongzhang Shan, Leonid Oliker, Rupak Biswas, 2003. Job Superscheduler Architecture and Performance in Computational Grid Environments. ACM/IEEE Conference on Supercomputing, 44-58.

[9] Santoso, J., van Albada, G.D., Nazief, B.A.A., Sloot, P.M.A., 2000. Hierarchical Job Scheduling for Clusters of Workstations. 6th Annual Conference of the Advanced School for Computing and Imaging, 99-105.

[10] Ng Wai Keat, Ang Tan Fong, Ling Teck Chaw, Liew Chee Sun, 2006. Scheduling Framework for Bandwidth-Aware Job Grouping-Based Scheduling in Grid

Computing. Malaysian Journal of Computer Science 19 (2), 117-126.

[11] Homer Wu, Chong Yen Lee, Wuu Yee chen, Tsang Lee, 2007. A Job schedule Model Based on Grid Environment. 1st IEEE International Conference on Complex, Intelligent and Software Intensive System, 43-52.

[12] Somasundaram, K., Radhakrishnan, S., Gomathynayagam, M., 2007. Efficient Utilization of Computing Resources using Highest Response Next Scheduling in Grid. Asian Journal of Information Technology 6 (5), 544-547.

[13] Diana Moise, Izabela Moise, Florin Pop, Valentin Cristea, 2008. Resource CoAllocation for Scheduling Tasks with Dependencies in Grid. International Workshop on High Performance in Grid Middleware, 41-48.

[14] Somasundaram, K., Radhakrishnan, S., 2008. Node Allocation in Grid Computing using Optimal Resource Constraint (ORC) Scheduling. International Journal of Computer Science and Network Security 8 (6), 309-313.

[15] Quan Liu, Yeqing Liao, 2009. Grouping-Based Fine-grained Job Scheduling in Grid Computing. 1st IEEE International Workshop on Education Technology and Computer Science, 556-559.

[16] Yeqing Liao, Quan Liu, 2009. Research on Fine-grained Job Scheduling in Grid Computing. International Journal of Information Engineering and Electronic Business, 9-16.

[17] Vishnu Kant Soni, Raksha Sharma, Manoj Kumar Mishra, 2010. Grouping-Based Job Scheduling Model in Grid Computing. World Academy of Science, Engineering and Technology 65, 781-784.

[18] Grace Mary Kanaga, E., Valarmathi, M.L., Juliet A Murali, 2010. Agent Based Patient Scheduling Using Heuristic Algorithm. International Journal on Computer Science and Engineering 2, 69-75.

[19] Roodbergen, K.J., De Koster, R., 2001. Routing Methods for Warehouses with Multiple Cross Aisles. International Journal of Production Research 39 (9), 1865–1883.

[20] Roodbergen, K.J., 2001. Layout and Routing Methods for Warehouses. Ph.D. Thesis. Erasmus Research Institute of Management (ERIM), Erasmus University Rotterdam, The Netherlands.

[21] Chau, K.W., Ying Cao, Anson, M., Jianping Zhang, 2002. Application of Data Warehouse and Decision Support System in Construction Management. Automation in Construction 12 (2), 213-224.

[22] Gademann, N., van de Velde, S., 2005. Order Batching to Minimize Total Travel Time in a Parallel-Aisle Warehouse. IIE Transactions 37, 63-75.

[23] Tho Le-Duc, Rene´ M.B.M. de Koster, 2007. Travel Time Estimation and Order Batching in a 2-block Warehouse. European Journal of Operational Research 176, 374–388.

[24] Ali Allahverdi, Ng, C.T., Cheng, T.C.E., Mikhail Y. Kovalyov, 2008. A Survey of Scheduling Problems with Setup Times or Costs. European Journal of Operational Research 187, 985–1032.

[25] Sheng Yuan Hsu, Liu, C.H., 2009. Improving the Delivery Efficiency of the Customer Order Scheduling Problem in a Job Shop. Computers & Industrial Engineering 57, 856–866.

[26] Sebastian Henn, Gerhard Wäscher, 2012. Tabu Search Heuristics for the Order Batching Problem in Manual Order Picking Systems. European Journal of Operational Research, Accepted manuscript, 1-31.

9. AUTHORS PROFILE

S. Krishnaveni completed M.C.A., M. Phil. and currently pursuing Ph.D in computer science at Karpagam University under the guidance of Dr. M. Hemalatha, Professor and Head, Dept. of Software System, Karpagam University, Coimbatore. Published five papers in International Journals and presented one paper in National Conference and two papers in International Conferences. Area of research: Data Mining, Data Warehouse and Grid Computing.

Dr. M. Hemalatha completed M.Sc., M.C.A., M. Phil., Ph.D (Ph.D, Mother Terasa women's University, Kodaikanal). She is Professor & Head and guiding Ph.D Scholars in Department of Computer Science at Karpagam University, Coimbatore. Twelve years of experience in teaching and published more than hundred papers in International Journals and also presented more than eighty papers in various National and International Conferences. She received best researcher award in the year 2012 from Karpagam University. Her research areas include Data Mining, Image Processing, Computer Networks, Cloud Computing, Software Engineering, Bioinformatics and Neural Network. She is a reviewer in several National and International Journals.