Parallel Simulated Annealing for Materialized View Selection in Data Warehousing Environments

Parallel Simulated Annealing for Materialized View Selection in Data Warehousing Environments

Roozbeh Derakhshan [1], Bela Stantic [2], Othmar Korn [2], and Frank Dehne [3]

[1] ETH Zurich, Switzerland
[2] Institute for Integrated and Intelligent Systems
Griffith University, Brisbane, Australia
[3] School of Computer Science, Carleton University, Canada

Abstract. *In order to facilitate efficient query processing, the information contained in data warehouses is typically stored as a set of materialized views. Deciding which views to materialize represent a challenge in order to minimize view maintenance and query processing costs. Some existing approaches are applicable only for small problems, which are far from reality. In this paper we introduce a new approach for materialized view selection using Parallel Simulated Annealing (PSA) that selects views from an input Multiple View Processing Plan (MVPP). With PSA, we are able to perform view selection on MVPPs having hundreds of queries and thousands of views. Also, in our experimental study we show that our method provides a significant improvement in the quality of the obtained set of materialized views over existing heuristic and sequential simulated annealing algorithms.*

Key words: Parallel Simulated Annealing, Data Warehousing, Materialized view selection

1 Introduction

Data warehouses integrate data from multiple heterogeneous databases and other information sources. A data warehouse(DW) is a repository of historical information available for querying and analysis. To avoid accessing the original data sources and increase the efficiency of the warehousing queries, information within a data warehouse is organized as a set of views from different production databases. These views are often referred to as materialized views. The large computation and space required for view materialization implies that it is impractical to materialize all possible views. Hence, there is a need for selecting an appropriate set of views to materialize which increases the query performance, commonly referred to as the *view selection problem* [9].

Because materialized views have to be in synchronization with source data, any change to the source should be reflected to the views as well. Therefore, in the data warehousing view maintenance cost also has to be considered not just the query processing cost. The trade-off between query performance and view

maintenance cost makes materialized view selection one of the most challenging problems in data warehousing [13]. Based on a set of frequently asked DW queries, the task is to select a set of views to materialize so that the total query processing and view maintenance cost is minimized.

The materialized view selection problem is NP-hard[9]. Several heuristic algorithms have been proposed in the literature to address the view selection problem. We classified them into four major groups according to [1]:

Deterministic algorithms: The classic solution for this problem uses heuristics which usually construct or search a solution in a deterministic manner and apply some kind of heuristics(e.g greedy algorithm) to decrease the solution space [10, 9, 2]. In [11] an extension is proposed, which improved the quality by using index on the selected views. In [3] a "chunk" based precomputation method was introduced. This method precomputes a subset of chunk aggregates which provide better but not near optimal results over the heuristic approaches.

Genetic algorithms (GA): The above methods are effective when the number of views is relatively small. In order to obtain better solutions for a bigger number of views with respect to view maintenance and query processing costs genetic algorithms have been introduiced [16, 4]. The basic idea is to start with a random initial population and generate offspring by random variations (e.g., crossover and mutation). The "fittest" members of the population survive the subsequent selection. The algorithm terminates as soon as there is no further improvement over a period or after a predetermined number of generations. The fittest individual found is the solution. However, the possibility of infeasible solutions creates some problems. In fact, the approach proposed in [4] does not contain a "penalty" method to discourage infeasible solutions. This deficiency has subsequently been addressed in [16].

Randomize algorithms: Algorithms in this class pursue a completely different approach: a set of moves is defined. These moves constitute edges between the different solutions of the solution space; two solutions are connected by an edge if (and only if) they can be transformed into one another by exactly one move. Simulated Annealing(SA) as a type of randomize algorithm performs a random walk along the edges according to a cooling schedule, and terminates as soon as no applicable ones exist or lose all the energy in the system(frozen state). In [19, 17], SA has been applied to the view selection problem. [19], showed that by using SA the cost of a selected set of materialized views is up to 70% less than the genetic [4] and heuristic algorithms [15].

Hybrid algorithms: Hybrid algorithms combine the strategies of pure deterministic and pure randomized algorithms. Solutions obtained by deterministic algorithms are used as starting points for randomized algorithms or as initial population members for genetic algorithms. In [5], hybrid approach has been applied for the view selection problem, which combines the power of genetic algorithms in global search with heuristic's ability in fine-grained local search, to find a good set of materialized views.

In[4, 15, 19, 5], GA and SA tries to find the best set of intermediate results(views) in the Multiple View Processing Plan(MVPP) graph [15] so that

the cost of query processing and view maintenance is minimized. However, the number of views in the MVPP graph is relatively small(e.g: 60 queries and 250 views). In [16, 10] genetic algorithms and heuristics have been proposed to select the best set of views to materialize from an AND/OR view graph [9]. The number of nodes in their AND/OR view graph is not going further than 250 either.

In this paper we introduce a new approach for materialized view selection using Parallel Simulated Annealing (PSA) to select views from an input Multiple View Processing Plan (MVPP). With PSA, we are able to perform view selection on MVPPs having a much larger number of queries and views, which reflects the real data warehousing environment. As solution quality is affected by the number of times that the initial solution is perturbed, by performing simulated annealing with multiple inputs over multiple compute nodes concurrently, PSA is able to increase the quality of obtained sets of materialized views. In experimental study, conducted on real production data with more than 250 queries and thousand of views (intermediate nodes), we showed that our approach using PSA in conjunction with MVPP outperforms heuristic method [15] and sequential SA [19] to the extent of factor five considering the cost of obtained set of views.

The rest of this paper is organized as follows: Section 2 gives an overview on our framework for the materialize view selection problem, followed by our running example and some preliminaries for the Multiple View Processing Plan (MVPP) and its cost model. Section 4 discusses our Parallel Simulated Annealing(PSA) approach and how we apply PSA to solve materialized view selection. Section 5 present and analyse our experimental results. Section 6 concludes the paper.

2 Materialized view selection

Materialized view selection is an important design decision in data warehouse construction. Here we present our framework to select a set of views to materialize based on the given frequently used set of queries in the data warehouse environment. As figure 1 shows, the input is a list of frequently used queries. This list will then be an input to our XML convertor box, which translates the text queries to XML format. We found that the MVPP builder works better with XML format than with plain text. The out-put from the XML convertor will go to the MVPP builder which creates the MVPP graph. The MVPP graph will be an input to our parallel simulated annealing algorithm. The output from the simulated annealing algorithm would be an appropriate set of nodes to materialize in order to minimize the query processing and view maintenance cost.

2.1 Running example

In this section, we present an example to motivate the discussion of materialized view selection in data warehouses. Our example is taken from a sample data

Fig. 1. A framework for materialized view selection in data warehousing environments

warehouse application that analyzes trends in sales, and which was used in [14]. We used this running example just for explanation, however the data and query sets which we used for our experiments are explained in section 5. The relations and the attributes of the running example's schema are:

```
Product  (Pid, name, Did)
Division (Did, name, city)
Order (Pid,Cid, quantity, date)
Customer (Cid, name, city)
Part (Tid, name, Pid, supplier)
```

We use `Pd, Div, Ord, Cust` and `Pt` to refer to the above relations. Furthermore, we assume that all of these relations are stored at the same site and we do not need to consider data communication costs in our cost calculation. Suppose that we have the four following frequently used queries:

```
Query 1: Select Pd.name
         From Pd, Div
         Where Div.city= "LA" and
         Pd.Did=Div.Did
```

```
Query 2: Select Pt.name
         From Pd, Pt, Div
         where ere Div.city="LA"
         and Pd.Did=Div.Did
         and Pt.Pid=Pd.Pid
```

```
Query 3: Select Cust.name,
         Pd.name, quantity
         From Pd, Div, Ord, Cust
         Where Div.city= "LA" and
         Pd.Did=Div.Did  and
         Pd.Pid=Ord.Pid and
         Ord.Cid=Cust.Cid and
         Date > 7/1/96
```

```
Query 4: Select Cust.city,date
         From Ord, Cust
         Where quantity>100 and
         Ord.Cid=Cust.Cid
```

In Figure 2 we show a global query access plan for the above four queries. This plan is referred to as the Multiple View Processing Plan (MVPP)[15]. The query access frequencies are indicated above each query node. For simplicity, we assumed that the base relations `Pd, Div, Ord, Cust`, and `Pt` are updated once during the process of materialized view selection. There are different options for selection of a set of views to be materialized: (1) materialize all of the nodes in the MVPP; (2) materialize some of the intermediate nodes (e.g. tmp2, tmp3, tmp7, etc.); (3) do not materialize any of the nodes in MVPP. Option (1) and

(3) are not realistic because for option (1), we do not have enough time and space to materialize all of the nodes in MVPP. Option (3) implies that we have to execute all queries on the raw data set which will result in excessive query processing times. The best option is to materialize an appropriate subset of views that minimizes view maintenance and query processing costs.

Suppose there are some materialized intermediate nodes in the MVPP. For each query, the cost of query processing is its query frequencies multiplied by the cost of the query accesses to the materialized nodes. The maintenance cost for materialized view is the cost used for construction of the view (here we assume that rebuilding is used whenever an update of an involved base relation occurs) [15]. For example, if tmp2 is materialized, the query processing cost for `Q1` is $10 * 35.25$. The view maintenance cost is $2 * (35.25 + 0.25)$. The total cost for an MVPP is the sum of all query processing and view maintenance costs. What follows is a specification and the definition of the cost model for an MVPP.

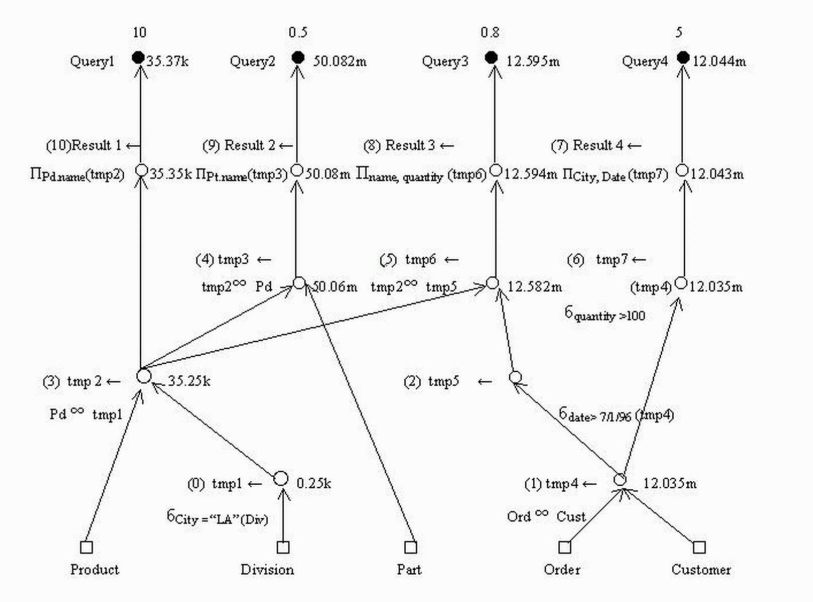

Fig. 2. A MVPP for running example queries

3 Multiple View Processing Plan (MVPP)

We are using an MVPP [15] together with parallel simulated annealing for selecting the best set of views to materialize. As shown in Figure 2, the MVPP is a directed acyclic graph (DAG) that represents a query processing plan. The leaf nodes in this graph represent the base relations, and the root nodes represent

the queries. Analogous to query execution plans there can be more than one MVPP for the same set of views. This depends upon the access characteristics of the applications and physical data warehouse parameters. We choose one of the possible optimal MVPPs. Note that the quality of the selected MVPP can effect on our result. An MVPP is a DAG $M = (V, A, C_q^a, C_m^r, f_q, f_u)$ where V is a set of vertices, A is a set of arcs over V defined as follows:

- For every relational algebra operation in the query tree, for every base relation, and for every distinct query, create a vertex;
- For $v \in V, T(v)$ is the relation generated by the corresponding vertex v. $T(v)$ can be a base relation, intermediate node while processing a query, or the final result for a query;
- For any leaf vertex v, (that is one which has no edges pointing to the vertex), $T(v)$ corresponds to a base relation. Let L be a set of leaf nodes;
- For any root vertex v (that is one which has no edges going out of the vertex), $T(v)$ corresponds to a global query. Let R be a set of root nodes;
- If the base relation or intermediate result relation $T(u)$ corresponding to vertex u is needed for further processing at a node v, introduce an arc $u \longrightarrow v$;
- For every vertex v, let $S(v)$ denote the source nodes which have edges pointed to v; for any $v \in L$,$S(v) = \phi$, $S^*(v)$ be the set of descendants of v;
- For every vertex v let $D(v)$ denote the destination nodes to which v is is pointed; for any $v \in R$, $D(v) = \phi$;
- For $v \in V$,C_q^a is the cost of query processing q accessing $T(v)$; $C_m^r(v)$ is the cost of maintaining T (v) based on changes to the base relation $S^*(v) \cap R$, if $T(v)$ is materialized.
- f_q, f_u denote query frequency and base relation maintenance frequency respectively.

3.1 Cost Model

We can now define the cost function for our problem, similar to the cost function in [15]. The cost function has two parts. One is the query processing cost:

$$C_{queryprocessing}(v) = \Sigma_{q \in R} f_q C_a^q(v)$$

the second part is the materialized view maintenance cost:

$$C_{maintenance}(v) = \Sigma_{r \in R} f_u C_m^r(v)$$

the total cost is the sum of the query processing and maintenance costs:

$$C_{total}(v) = C_{queryprocessing}(v) + C_{maintenance}(v)$$

Our goal is to find the set of views so that if the members of the set are materialized then the value of C_{total} will be smallest among all possible feasible sets of materialized views.

4 Parallel simulated annealing for materialized view selection

The motivation to use a Parallel Simulated Annealing (PSA) algorithm in solving the materialized view selection problem was based on observing that the data warehouse has a huge number of views and queries. Therefore in the view selection problem the solution space has many local minimas. A simple local search algorithm proceeds by choosing a random initial solution and generating a neighbor from that solution. The neighboring solution is accepted if it is a cost decreasing transition. Such a simple algorithm has the drawback of often being trapped to a local minimum. The simulated annealing algorithm, though by itself it is a local search algorithm, avoids getting trapped in a local minimum by also accepting cost increasing neighbors with some probability. In sequential SA according to [20]: first an initial solution is randomly generated, and a neighbor is found and is accepted with a probability of min(1,exp(-δ/T), where δ is is the cost difference and T is the control parameter corresponding to the temperature of the physical analogy and will be called temperature. On the slow reduction of temperature, the algorithm converges to the global minimum, but the time taken increases drastically.

Simulated annealing is inherently sequential and hence very slow for problems with large search space. Therefore, to speed up the computation a parallelization of SA is very desirable. Also, since solution quality in the SA algorithm is affected by the number of times that we perturb an initial random solution, the parallelization of SA with multiple inputs over multiple compute nodes concurrently will lead us to the better quality of solution.

In the following subsections, we describe how to apply PSA to design a solution for the materialized view selection problem. More precisely, we provide a suitable representation of solution space, followed by PSA's parameters and their desirable values.

4.1 Parallel simulated annealing framework

There have been many attempts toward parallelizing simulated annealing. Each of these methods classified parallel simulated annealing differently. Classification in [6][12] distinguished between *single* and *multiple-walks* (Figure 3). This is the first distinguishing criterion: the number of paths which are evaluated in the search space of the optimization problem. In a *single-walk* algorithm only a single path in the search space is traversed, whereas in a *multiple-walk* approach several different paths are evaluated simultaneously. In *single-walk* algorithms after evaluating a part of the neighborhood of the current solution either only one step is traversed (*single-step parallelism*) or a sequence of steps is made from the current solution (*multiple-step parallelism*). In *multiple-walk* algorithms the parallel walks can be independent or may interact according to a communication pattern.

In this paper, we are using the independent walks parallelization which is called the Multiple Independent Runs(MIR) [7]. In this parallelization strategy

no communication of moves or solutions is required. Independent runs of sequential SA are executed in each processor and the best found solution is chosen as the end result. Therefore, there is no need to add any communication cost to the total cost of the obtained set of materialized views.

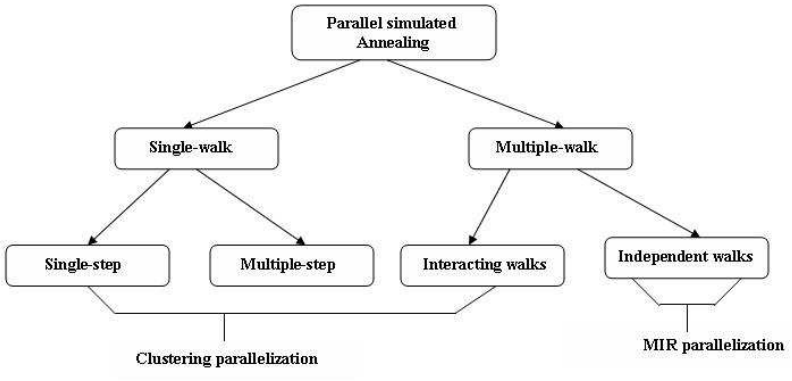

Fig. 3. Classification of parallel approaches for simulated annealing

4.2 Solution Representation

The problem to be solved can be stated as follows: given a MVPP graph (see figure 2) we attempt to find the best set of intermediate nodes (views) that can answer all queries with minimal cost. We do not use an MVPP directly as input into our PSA algorithm. We first convert the set of views to a binary string of 1s and 0s to represent views that will and will not be materialized, respectively. Our mapping strategy differs from [4],[5] and [16]. We number our nodes starting at the base relations moving left to right, and we continue up to the right-most node at the top of the graph. Nodes are numbered 0 to m-1 (where m is the number of intermediate nodes). We use a mapping array of size m-1 where each index in the array corresponds to a graph node. In our mapping array a '1' denotes that the corresponding node in the graph should be materialized and a '0' that the node is not materialized. For example in the binary string (0,0,0,0,1,1,0,0,1,1,0) we will materialize nodes 4,5,8 and 9.

4.3 Parallel Simulated Annealing Parameters

The success and quality of the SA algorithms either sequential or parallel relies on choosing the right parameters. Generally, we can categorize SA parameters into two separate classes: *generic parameters* and *problem specific parameters*.

Generic parameters such as: initial temperature, cooling schedule and run factor are concerned with parameters of the SA algorithm itself . The *problem specific parameters* such as: initial configuration of our solution space, perturbing the configuration and cost function are dependent on the specific problems.

Here we first explain each of the *generic parameters*:

Initial temperature: The temperature T can affect the number and ratio of acceptance of each move. This value has traditionally been chosen so that nearly all moves are accepted. We set our starting temperature large enough to allow an acceptance value of 90. If the starting temperature is larger, the run length may increase with no improvement in cost. Too low temperature may lead to premature levelling off of the algorithm.

Cooling schedule: The temperature decrement factor for the exponential cooling is set to a constant value of 0.7. This value performed sufficiently well on our problem, although the algorithm is not particularly sensitive to this parameter.

Run factor: In this paper we use MIR which provides a better quality solution than the solution of a sequential run with the same length. We have another important parameter which we call *run factor*. a bigger value of run factor means more iterations for each run and we will gain the better quality solution. However, this increasing run length will increase the time length for each run and the complete annealing process. So we have to choose a run factor big enough to gain the high quality of answer in a reasonable amount of time. In our experiments, we found that the quality of answer is heavily depending on the value of run factor. Thus, we set the appropriate value for run factor after many test runs.

The *problem specific parameters* are:

Initial configuration: In our initial configuration we map array with a randomized set of zeros and ones. We do not employ a penalty function to discourage infeasible solutions, instead we use a simple verification method. We check the feasibility of each initial configuration against the number of queries that the solution can answer. If the solution is not feasible, we simply bypass it. For example the sequence (0,0,0,1,0,0,1,0,0,0,0) is a feasible solution for our sample problem. In figure 2 the materialized node 3 can answer queries 1 to 3 and materialized node 6 can answer the 4th query.

Perturbing the configuration: In the spirit of the physical annealing process, neighboring configurations must be similar in the sense that they represent only a slight perturbation in the system's state[18]. We define the neighborhood of a configuration to contain all configurations that differ from it by giving a 50% chance to each randomly chosen node to be materialized or unmaterialized. For example, for solution (0,0,0,1,1,0,1) we randomly pick node number 4 whose value is 1, then we just simply change the value to 0. So our solution after perturbing would be (0,0,0,0,1,0,1). Our experiments showed that this simple method ensures that our annealing algorithm will not get trapped in local minima in early stages.

Cost function: We use the cost function described in section 3.1. For example, to calculate the overall cost for the solution (0,0,0,1,0,0,1,0,0,0,0) we add C_{total} for nodes 3 and 4.

5 Experimental Evaluation

To show the practical relevance of our approach to materialized views selection problem, we performed an extensive experimental evaluation and compared it with heuristic method [15] and sequential SA because in previous study it was shown that the sequential SA is better than the GA [19]. The experiment involves execution of our PSA application over an optimized MVPP for a set of queries. The PSA application is a C++ program, which uses a robust PSA library (parSA 2.1) implementation [8] with the addition of materialized view selection component. Tests are performed on SUN Microsystems V20z dual AMD Opteron 2.6 GHz with 4GB RAM. The MVPP input is provided by our custom C++ MVPP builder, which creates an optimized MVPP for testing set of SQL queries, their frequency of usage and number of rows in tables as a input. The number of nodes in our MVPP inputs exceeds one thousand. For the testing, we used the real data from production database. This database is used for generation of data warehousing database, which analyzes the trend in using university resources. The source database consist of more than 100 relations. The number of rows in particular tables is up to 10 million. We have chosen 250 frequently used queries and assigned frequency according to usage.

5.1 Results

In Figure 4 we show results for our PSA algorithm (for 4, 8 and 16 compute nodes) against the heuristic and sequential SA algorithms. The heuristic algorithm provides a benchmark for our normalized results. The graph shows that our PSA algorithm approach generates solutions with costs more than 4 times less than the heuristic algorithm. For a smaller number of queries the results for sequential SA and PSA are similar. For a larger number of queries (more than 150) the PSA algorithm outperforms sequential SA, particularly for 16 compute nodes. The PSA results are consistently better than both the sequential SA and heuristic algorithms.

6 Conclusion and Future work

In this paper we have described a new approach that is demonstrably better than the existing approaches for materialized view selection based on Parallel Simulated Annealing. In experimental study we show that our approach provides a significant improvement in the quality of the obtained set of materialized views compared to previously used methods for materialized view selection (Heuristic method and sequential SA). Additionally we show that our method can be

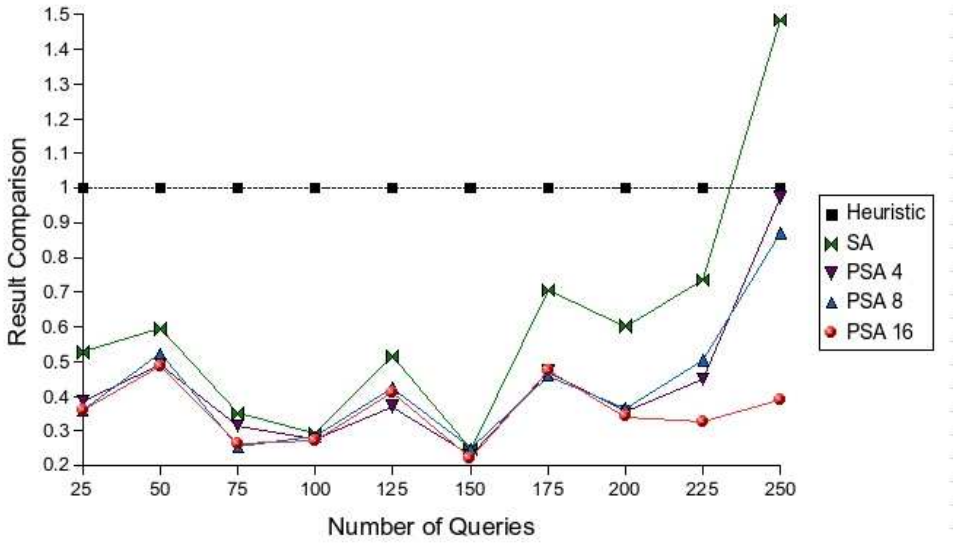

Fig. 4. Comparison of the solution quality (view maintenance and query processing costs) between our PSA, sequential SA algorithm, and heuristic method (normalized to "1")

efficiently applied to the large data warehousing systems, this leads to a significant improvement in query processing time and view maintenance costs. More specifically, in this study, we:

- Classified the existing methods for materialized view selection problem in order to identify their advantages and disadvantages,
- Proposed Parallel simulated annealing (PSA) framework which uses as input Multiple View Processing Plan (MVPP),
- showed that PSA can be efficiently applied to larger number of queries. Larger number of queries is more representative of real data warehousing systems,
- Experimentally evaluated the PSA by comparing its performance with heuristic method and sequential SA, and demonstrated its overall superior performance. The PSA algorithm approach generates solutions with costs up to 4 times less than the heuristic algorithm,
- We showed that PSA scaled with the increasing number of compute nodes.

As a future work we intend to do the testing with larger number of compute nodes and to use a more sophisticated parallel approach to achieve further improvement in the quality of results.

Acknowledgements

This research is partly sponsored by ARC (Australian Research Council) Discovery grant - Coarse Grained Parallel Algorithms, nr. DP0557303.

References

1. M. Steinbrunn and J. Moerkotte and A. Kemper . Heuristic and Randomized Optimization for the Join Ordering Problem . *Very Large Data Base*, 6:191–208, 1997.
2. V. Harinarayan and A. Rajaraman and J.D. Ullman. Implementing Data Cubes Efficiently. *ACM SIGMOD*, pages 205–216, 1996.
3. A. Shukla and P.Deshpande and J. Naughton. Materialized View Selection of Multidimensional Datasets. *Proceeding of the 24th VLDB Conference*, pages 488–499, 1998.
4. C. Zhang and J. Yang . Genetic Algorithm for Materialized View Selection in Data Warehouse Environments. *Proc. Intl Conf. Data Warehousing and Knowledge Discovery (DaWaK)*, pages 116–125, 1999.
5. C. Zhang and X. Yao and J. Yang. An Evolutionary Approach to Materialized Views Selection in a Data Warehouse Environment. *IEEE Transactions on Systems and Cybernetics Part C: Applications and Reviews*, 31(3):282–294, 2001.
6. E. Aarts and K.J. Lenstra. Local Search in Combinatorial Optimization. *John Wiley*, 1997.
7. G. Kliewer and S. Tschoke. A General Parallel Simulated Annealing Library and its Application in Airline Industry. *Proceedings of the 14th International Parallel and Distributed Processing Symposium (IPDPS)*, pages 55–61, 2000.
8. G. Kliewer and S. Tschoke. Parallel Simulated Annealing Library (parSA). *http://www.uni-paderborn.de/~parsa*, *University of Paderborn*, 2007.
9. H. Gupta and S. Mumick. Selection of Views to Materialize Under a Maintenance Cost Constraint. *Lecture Notes In Computer Science - International Conference on Database Theory*, 1540:453–470, 1998.
10. H. Gupta and S. Mumick. Selection of Views to Materialize in a Data Warehouse. *IEEE Transactions on Knowledge and Data Engineering*, 17(11):24–43, 2005.
11. H. Gupta and V. Harinarayan and A. Rajaraman and J.D. Ullman. Index Selection for OLAP. *Proc. Int'l Conf. on Data Engineering*, pages 208–219, 1997.
12. I.R. Azencott. Simulated Annealing: Parallelization Techniques. *Wiley*, 1992.
13. J. Widom . Research Problems in Data Warehouse. *4th International Conferance on Information, Knowledge and Managment*, pages 25–30, 1995.
14. J. Yang and K. Karlapalem and Q. Li. A Framework for Designing Materialized Views in Data Warehousing Environment. *Technical Report HKUST-CS96-35*, 1996.
15. J. Yang and K. Karlapalem and Q. Li. Algorithm for Materialized View Design in Data Warehousing Environment. *VLDB'97*, pages 136–145, 1997.
16. M. Lee and J. Hammer. Speeding up Materialized View Selection in Data Warehouses Using a Randomized Algorithm. *Int. J. Cooperative Inform. Syst.*, 10:327–353, 2001.
17. P. Kalnis and N. Mamoulis and D. Papadias. View Selection Using Randomized Search. *Data and Knowledge Engineering*, 42(1):89–111, 2002.
18. R. Davidson and D. Harel. Drawing Graphs Nicely using Simulated Annealing. *ACM Transactions on Graphics*, 15:301–331, 1996.
19. R. Derakhshan and F. Dehne and O. Korn and B. Stantic. Simulated Annealing for Materialized View Selection in Data Warehousing Environment. *Proceedings of the 24th IASTED international conference on Database and applications*, pages 89–94, 2006.
20. R. Janaki and T.H. Sreenivas and G.K. Subramaniam . Parallel Simulated Annealing Algorithms. *Journal of parallel and distributed computing*, 37:207–212, 1996.